

Hardware Reference Manual

Version 1.0

October 2018

Redpine Signals, Inc.

2107 North First Street, Suite #540, San Jose, California 95131,
United States of America.

Phone: +1-408-748-3385, Fax: +1-408-705-2019

Email: sales@redpinesignals.com

Website: www.redpinesignals.com

Disclaimer

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only.

Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

Copyright © 2018 Redpine Signals, Inc. All rights reserved.

Table of Contents

| | | |
|--------|---|-----|
| 1 | Cortex-M4 | 26 |
| 1.1 | Cortex-M4 Introduction | 26 |
| 1.1.1 | General Description | 26 |
| 1.1.2 | System level interface..... | 27 |
| 1.1.3 | Integrated configurable debug | 27 |
| 1.1.4 | Cortex-M4 processor features and benefits summary | 27 |
| 1.1.5 | Cortex-M4 core peripherals | 28 |
| 1.1.6 | More Information | 28 |
| 1.2 | Cortex-M4 Processor..... | 28 |
| 1.2.1 | Programmers Model | 28 |
| 1.2.2 | Memory model | 38 |
| 1.2.3 | Exception model | 46 |
| 1.2.4 | Fault handling | 52 |
| 1.2.5 | Power management | 54 |
| 1.3 | Cortex-M4 Instruction Set..... | 56 |
| 1.3.1 | CMSIS functions | 63 |
| 1.3.2 | About the instruction descriptions | 65 |
| 1.3.3 | Memory access instructions | 72 |
| 1.3.4 | General data processing instructions | 82 |
| 1.3.5 | Multiply and divide instructions..... | 103 |
| 1.3.6 | Saturating instructions..... | 115 |
| 1.3.7 | Packing and unpacking instructions | 123 |
| 1.3.8 | Bitfield instructions..... | 127 |
| 1.3.9 | Branch and control instructions | 129 |
| 1.3.10 | Floating-point instructions..... | 133 |
| 1.3.11 | Miscellaneous instructions | 148 |
| 1.4 | Cortex M4 Peripherals..... | 154 |
| 1.4.1 | Nested Vectored Interrupt Controller | 154 |
| 1.4.2 | System control block | 161 |
| 1.4.3 | System timer, SysTick | 180 |
| 1.4.4 | Memory protection unit..... | 182 |

| | | |
|-------|---|-----|
| 1.4.5 | MPU Type Register | 184 |
| 1.4.6 | MPU Region Number Register | 186 |
| 1.4.7 | Floating Point Unit (FPU) | 194 |
| 1.4.8 | Coprocessor Access Control Register | 195 |
| 1.5 | Instruction Cache Controller | 200 |
| 1.5.1 | General Description | 200 |
| 1.5.2 | Features | 200 |
| 1.5.3 | Functional Description | 200 |
| 1.5.4 | Register Summary | 201 |
| 1.5.5 | Register Description | 201 |
| 2 | MCU Bus Matrix | 205 |
| 2.1 | General Description | 205 |
| 2.2 | Features | 205 |
| 2.3 | Functional Description | 205 |
| 2.3.1 | Overview | 205 |
| 2.3.2 | Master and Slave Details | 205 |
| 2.3.3 | Address Mapping | 207 |
| 2.3.4 | Register Summary | 209 |
| 2.3.5 | Register Description | 209 |
| 3 | Memory Architecture | 210 |
| 3.1 | General Description | 210 |
| 3.2 | Features | 210 |
| 3.3 | Functional Description | 210 |
| 3.4 | Unified Memory Architecture and Multiport | 211 |
| 3.4.1 | LP SRAM | 211 |
| 3.4.2 | ROM | 211 |
| 3.4.3 | HP SRAM1 | 211 |
| 3.4.4 | HP ROM | 211 |
| 3.4.5 | HP SRAM2 | 211 |
| 3.4.6 | ULP Memory | 212 |
| 3.4.7 | ICache Memory | 212 |
| 3.4.8 | Flash Memory | 212 |
| 3.5 | Frequency and Latency | 212 |

| | | |
|--------|--|-----|
| 3.6 | Memory Protection | 212 |
| 3.6.1 | Features..... | 212 |
| 3.6.2 | Trap Generation and Handling..... | 213 |
| 3.7 | Programming Sequence..... | 213 |
| 3.8 | Register Summary..... | 213 |
| 3.9 | Register Description..... | 215 |
| 3.9.1 | ENABLE_TRAP_REG | 215 |
| 3.9.2 | IBUS_TRAP_ENABLE_REG_HP_SRAM1..... | 215 |
| 3.9.3 | DMA_WR_TRAP_ENABLE_REG_HP_SRAM1..... | 216 |
| 3.9.4 | DMA_RD_TRAP_ENABLE_REG_HP_SRAM1 | 216 |
| 3.9.5 | AHB_MASTER_NUM_MASK_REG_HP_SRAM1 | 216 |
| 3.9.6 | AHB_MASTER_NUM_MASK_REG_LP_SRAM | 217 |
| 3.9.7 | TRAP_DETECTED_REG_HP_SRAM1 | 217 |
| 3.9.8 | IBUS_TRAP_STATUS_REG_HP_SRAM1..... | 217 |
| 3.9.9 | IBUS_TRAP_STATUS_REG_LP_SRAM | 217 |
| 3.9.10 | DMA0_TRAP_STATUS_REG_HP_SRAM1 | 218 |
| 3.9.11 | DMA0_TRAP_STATUS_REG_LP_SRAM..... | 218 |
| 3.9.12 | DMA1_TRAP_STATUS_REG_HP_SRAM1 | 218 |
| 3.9.13 | DMA1_TRAP_STATUS_REG_LP_SRAM..... | 219 |
| 3.9.14 | DBUS_TRAP_STATUS_REG_HP_SRAM1 | 219 |
| 3.9.15 | DBUS_TRAP_STATUS_REG_LP_SRAM..... | 219 |
| 3.9.16 | IBUS_TRAP_ENABLE_REG_LP_SRAM | 220 |
| 3.9.17 | DMA_RD_TRAP_ENABLE_REG_LP_SRAM | 220 |
| 3.9.18 | DMA_WR_TRAP_ENABLE_REG_LP_SRAM | 220 |
| 3.9.19 | TRAP_STATUS_HP_SRAM1..... | 221 |
| 3.9.20 | TRAP_STATUS_LP_SRAM | 221 |
| 3.9.21 | TRAP_STATUS_HP_SRAM2..... | 221 |
| 3.9.22 | TRAP_CLEAR_HP_SRAM1 | 221 |
| 3.9.23 | TRAP_CLEAR_LP_SRAM | 222 |
| 3.9.24 | TRAP_CLEAR_HP_SRAM2 | 222 |
| 3.9.25 | TRAP_DETECTED_LP_SRAM..... | 222 |
| 3.9.26 | TRAP_DETECTED_HP_SRAM2 | 222 |
| 4 | Clock Architecture..... | 224 |
| 4.1 | General Description | 224 |

| | | |
|---------|---------------------------------|-----|
| 4.2 | Features | 224 |
| 4.3 | Functional Description | 224 |
| 4.3.1 | Overview..... | 224 |
| 4.4 | HF Crystal Clock | 225 |
| 4.5 | Naming Convention | 225 |
| 4.6 | Reference Clock | 227 |
| 4.6.1 | MCU HP..... | 227 |
| 4.6.2 | MCU ULP | 227 |
| 4.7 | Clocking Schemes | 228 |
| 4.8 | Register Summary..... | 228 |
| 4.9 | Register Description..... | 228 |
| 4.9.1 | MCU_REF_CLK_CONFIG..... | 228 |
| 4.10 | High-Frequency PLL..... | 229 |
| 4.10.1 | General Description | 229 |
| 4.10.2 | Features..... | 229 |
| 4.10.3 | Functional Description | 229 |
| 4.10.4 | Input Reference Clock..... | 230 |
| 4.10.5 | SoC-PLL | 230 |
| 4.10.6 | Interface-PLL | 231 |
| 4.10.7 | I2S-PLL..... | 232 |
| 4.10.8 | PLL Programming Baud Rate..... | 234 |
| 4.10.9 | Register Summary..... | 234 |
| 4.10.10 | Register Description..... | 235 |
| 4.11 | MCU HP Clock Architecture | 240 |
| 4.11.1 | General Description | 240 |
| 4.11.2 | Features..... | 240 |
| 4.11.3 | Functional Description | 240 |
| 4.11.4 | Low-Power Clock | 241 |
| 4.11.5 | Processor | 242 |
| 4.11.6 | SPI Flash Controller | 242 |
| 4.11.7 | SD-MEM (eMMC) | 243 |
| 4.11.8 | CCI | 244 |
| 4.11.9 | Ethernet RMII..... | 244 |

| | |
|--|------------|
| 4.11.10 UART1/UART2..... | 245 |
| 4.11.11 SPI / SSI Master | 246 |
| 4.11.12 I2S Controller | 247 |
| 4.11.13 CAN Controller..... | 248 |
| 4.11.14 Configurable Timers | 248 |
| 4.11.15 Generic SPI Master | 248 |
| 4.11.16 MCU-ULP SoC Clock | 249 |
| 4.11.17 External Clock | 250 |
| 4.11.18 Static Clock Gated Domains | 250 |
| 4.11.19 Register Summary..... | 251 |
| 4.11.20 Register Description..... | 252 |
| 4.12 MCU ULP Clock Architecture | 267 |
| 4.12.1 General Description | 267 |
| 4.12.2 Features..... | 267 |
| 4.12.3 Functional Description | 267 |
| 4.12.4 AHB Interface Clock | 268 |
| 4.12.5 UART | 269 |
| 4.12.6 SPI / SSI Master | 270 |
| 4.12.7 I2S Controller | 271 |
| 4.12.8 Timer..... | 272 |
| 4.12.9 Touch Sensor..... | 272 |
| 4.12.10 Aux-ADC/DAC Controller | 273 |
| 4.12.11 Voice-Activity Detection (VAD)..... | 273 |
| 4.12.12 UULP APB Clock | 274 |
| 4.12.13 Static Clock Gated Domains | 275 |
| 4.12.14 Register Summary..... | 275 |
| 4.12.15 Register Description..... | 275 |
| 4.13 MCU ULP VBAT Clock Architecture..... | 285 |
| 4.13.1 General Description | 285 |
| 4.13.2 Features..... | 285 |
| 4.13.3 Functional Description | 285 |
| 4.13.4 Register Summary..... | 286 |
| 4.13.5 Register Description..... | 286 |
| 4.14 ULP Clock Oscillators..... | 287 |

| | | |
|----------|--|------------|
| 4.14.1 | General Description | 287 |
| 4.14.2 | Features..... | 287 |
| 4.14.3 | Functional Description | 287 |
| 4.14.4 | Register Summary..... | 288 |
| 4.14.5 | Register Description..... | 288 |
| 5 | Resets | 290 |
| 5.1 | General Description | 290 |
| 5.2 | Features..... | 290 |
| 5.3 | Functional Description | 290 |
| 5.3.1 | RESET_N_PAD | 290 |
| 5.3.2 | POC_IN..... | 290 |
| 5.3.3 | Black Out Monitor | 291 |
| 5.3.4 | WatchDog Reset | 291 |
| 5.4 | Reset request from host or Processor..... | 291 |
| 5.5 | Register Summary..... | 291 |
| 5.6 | Register Description..... | 291 |
| 5.6.1 | BLACK_OUT_MON_EN_REG | 291 |
| 5.6.2 | BLACK_OUT_MON_TRIM_REG | 291 |
| 6 | Interrupts..... | 293 |
| 6.1 | General Description | 293 |
| 6.2 | Features | 293 |
| 6.3 | Functional Description | 293 |
| 6.3.1 | Flexible exception and interrupt management..... | 293 |
| 6.3.2 | Nested exception/interrupt support..... | 293 |
| 6.3.3 | Vectored exception/interrupt entry | 293 |
| 6.3.4 | Interrupt masking | 293 |
| 6.3.5 | Vector table | 294 |
| 6.3.6 | Vectored Interrupt Table (VIT)..... | 294 |
| 7 | Power Architecture | 307 |
| 7.1 | General Description | 307 |
| 7.2 | Features | 307 |
| 7.3 | Functional Description | 307 |
| 7.3.1 | Power Domains | 307 |

| | | |
|--------|---|-----|
| 7.3.2 | Programming Sequence | 309 |
| 7.4 | Voltage Domains | 310 |
| 7.5 | Power States | 311 |
| 7.5.1 | PS4 | 312 |
| 7.5.2 | PS3 | 313 |
| 7.5.3 | PS2 | 313 |
| 7.5.4 | PS1 | 313 |
| 7.5.5 | STANDBY | 313 |
| 7.5.6 | SLEEP | 314 |
| 7.5.7 | PS0 | 314 |
| 7.5.8 | Programming Sequence for transitions | 314 |
| 7.6 | Memory Retention in Sleep / Shutdown states | 318 |
| 7.7 | Wakeup Sources | 318 |
| 7.8 | Register Summary | 319 |
| 7.8.1 | High-Performance Power Domains | 319 |
| 7.8.2 | Low-Power Domains | 320 |
| 7.8.3 | Ultra Low-Power Domains | 320 |
| 7.8.4 | Analog Domains | 320 |
| 7.8.5 | SLEEP FSM | 320 |
| 7.8.6 | ULP Configuration | 321 |
| 7.9 | Register Description | 321 |
| 7.9.1 | Power_Control_SET | 321 |
| 7.9.2 | Power_Control_CLEAR | 322 |
| 7.9.3 | PLL_Power_Control | 323 |
| 7.9.4 | DLL_Power_Control | 324 |
| 7.9.5 | SRAM_Power_Control_REG1_SET | 324 |
| 7.9.6 | SRAM_Power_Control_REG1_CLEAR | 325 |
| 7.9.7 | SRAM_Power_Control_REG2_SET | 326 |
| 7.9.8 | SRAM_Power_Control_REG2_CLEAR | 328 |
| 7.9.9 | SRAM_Power_Control_REG3_SET | 329 |
| 7.9.10 | SRAM_Power_Control_REG3_CLEAR | 329 |
| 7.9.11 | SRAM_Power_Control_REG4_SET | 330 |
| 7.9.12 | SRAM_Power_Control_REG4_CLEAR | 332 |
| 7.9.13 | PMU_LDO_CTRL_REG_SET | 335 |

| | |
|--|------------|
| 7.9.14 PMU_LDO_CTRL_REG_CLEAR | 335 |
| 7.9.15 ULP_Peripheral_Power_Control_SET | 336 |
| 7.9.16 ULP_Peripheral_Power_Control_CLEAR | 336 |
| 7.9.17 ULP_SRAM_Power_Control_REG1_SET | 337 |
| 7.9.18 ULP_SRAM_Power_Control_REG1_CLEAR | 338 |
| 7.9.19 ULP_SRAM_Power_Control_REG2_SET | 338 |
| 7.9.20 ULP_SRAM_Power_Control_REG2_CLEAR | 339 |
| 7.9.21 ULP_SRAM_Power_Control_REG4_SET | 340 |
| 7.9.22 ULP_SRAM_Power_Control_REG4_CLEAR | 341 |
| 7.9.23 UULP_Peripheral_Power_Control_SET | 342 |
| 7.9.24 UULP_Peripheral_Power_Control_CLEAR | 342 |
| 7.9.25 FSM_SLEEP_CTRLS_AND_WAKEUP_MODE | 343 |
| 7.9.26 FSM_ONTIME_CONFIG_REG | 346 |
| 7.9.27 ULP_MODE_CONFIG | 347 |
| 7.9.28 FSM_POWER_CTRL_DELAY | 348 |
| 7.9.29 FSM_CTRL_POWER_DOMAINS | 349 |
| 7.9.30 ULP_ISOLATION_CTRL | 350 |
| 8 Power Management Unit..... | 351 |
| 8.1 General Description | 351 |
| 8.2 Features..... | 351 |
| 8.3 Functional Description | 351 |
| 8.3.1 Block Diagram | 351 |
| 8.3.2 Modes of Operation | 352 |
| 8.3.3 Bypass Options | 352 |
| 8.4 Register Summary..... | 356 |
| 8.5 Register Description..... | 357 |
| 8.5.1 PMU_IP3_CTRL_REG..... | 357 |
| 8.5.2 PMU_LDOSOC_REG..... | 358 |
| 8.6 PMU Good Time..... | 358 |
| 8.6.1 Direct Battery Connected PMU Good Time | 358 |
| 8.6.2 Cascaded Power Supply PMU Good Time | 359 |
| 9 ULP Regulators..... | 360 |
| 9.1 General Description | 360 |

| | | |
|--------|---|-----|
| 9.2 | Features..... | 360 |
| 9.3 | Functional Description | 360 |
| 9.3.1 | Block Diagram | 360 |
| 9.4 | Register Summary..... | 361 |
| 9.5 | Register Description..... | 361 |
| 9.5.1 | SCDC_CTRL_REG_0..... | 361 |
| 9.5.2 | BG_SCDC_PROG_REG_1..... | 362 |
| 9.5.3 | BG_SCDC_PROG_REG_2..... | 362 |
| 9.5.4 | BG_LDO_REG..... | 363 |
| 10 | Pad Configurations | 365 |
| 10.1 | General Description | 365 |
| 10.2 | Features..... | 365 |
| 10.3 | Functional Description | 365 |
| 10.3.1 | PAD Description | 366 |
| 10.3.2 | Programming Sequence..... | 368 |
| 10.3.3 | PAD Configuration and GPIO Mode Reset Values..... | 371 |
| 10.4 | DDR PADs..... | 373 |
| 10.4.1 | Overview | 373 |
| 10.5 | Register Summary..... | 373 |
| 10.5.1 | PAD Selection Registers..... | 373 |
| 10.5.2 | MCU HP GPIO PAD Configuration Registers..... | 373 |
| 10.5.3 | MCU ULP GPIO PAD Configuration Registers..... | 373 |
| 10.5.4 | MCU UULP Vbat GPIO PAD Configuration Registers..... | 374 |
| 10.6 | Register Description..... | 374 |
| 10.6.1 | MCUHP_PAD_SELECTION..... | 374 |
| 10.6.2 | MEM_GPIO_ACCESS_CTRL_SET..... | 374 |
| 10.6.3 | MEM_GPIO_ACCESS_CTRL_CLEAR | 375 |
| 10.6.4 | PAD_CONFIG_REG_n | 375 |
| 10.6.5 | ULP_PAD_CONFIG_REG0..... | 376 |
| 10.6.6 | ULP_PAD_CONFIG_REG1..... | 377 |
| 10.6.7 | ULP_PAD_CONFIG_REG2..... | 379 |
| 10.6.8 | UULP_VBAT_GPIOOn_CONFIG_REG..... | 379 |
| 10.6.9 | Register Summary..... | 380 |

| | |
|--|------------|
| 10.6.10 Register Description..... | 380 |
| 11 Pin MUX..... | 383 |
| 11.1 Overview..... | 383 |
| 11.2 Functional Description | 383 |
| 11.2.1 SoC GPIOs..... | 383 |
| 11.2.2 ULP GPIOs..... | 387 |
| 11.2.3 UULP VBAT GPIOs | 389 |
| 11.2.4 Description of Digital Peripheral Pins Multiplexed on GPIOs | 389 |
| 11.2.5 Description of Analog Peripheral Pins Multiplexed on GPIOs | 394 |
| 12 SPI Flash Controller | 396 |
| 12.1 General Description | 396 |
| 12.2 Features..... | 396 |
| 12.3 Functional Description | 397 |
| 12.3.1 Programming sequence | 398 |
| 13 GPDMA | 399 |
| 13.1 General Description | 399 |
| 13.2 Features..... | 399 |
| 13.3 Functional Description | 399 |
| 13.4 Programming sequence | 401 |
| 13.4.1 Interrupt configurations of GPDMA..... | 402 |
| 13.4.2 Transfer Size less than burst size error | 402 |
| 13.4.3 FIFO Re-Configuration | 402 |
| 13.5 Register Summary..... | 402 |
| 13.6 Register Description..... | 404 |
| 13.6.1 INTERRUPT_REG | 404 |
| 13.6.2 INTERRUPT_MASK_REG | 405 |
| 13.6.3 INTERRUPT_STAT_REG | 406 |
| 13.6.4 DMA_CHNL_ENABLE_REG | 409 |
| 13.6.5 DMA_CHNL_SQUASH_REG | 409 |
| 13.6.6 DMA_CHNL_LOCK_REG | 410 |
| 13.6.7 LINK_LIST_PTR_REG_CHNL_n..... | 410 |
| 13.6.8 SRC_ADDR_REG_CHNL_n..... | 410 |
| 13.6.9 DEST_ADDR_REG_CHNL_n..... | 410 |

| | |
|---|------------|
| 13.6.10 CHANNEL_CTRL_REG_CHNL_n..... | 410 |
| 13.6.11 MISC_CHANNEL_CTRL_REG_CHNL_n | 412 |
| 13.6.12 FIFO_CONFIG_REG_CHNL_n | 413 |
| 13.6.13 PRIORITY_LEVEL_REG_CHNL_n..... | 413 |
| 14 Micro DMA (uDMA) | 414 |
| 14.1 General Description | 414 |
| 14.2 Features | 414 |
| 14.3 Functional Description | 414 |
| 14.4 Programming Sequence..... | 415 |
| 14.5 Register Summary..... | 415 |
| 14.6 Register Description..... | 416 |
| 14.6.1 DMA_STATUS..... | 416 |
| 14.6.2 DMA_CFG | 417 |
| 14.6.3 CTRL_BASE_PTR | 418 |
| 14.6.4 ALT_CTRL_BASE_PTR..... | 418 |
| 14.6.5 DMA_WAITONREQUEST_STATUS..... | 418 |
| 14.6.6 CHNL_SW_REQUEST..... | 418 |
| 14.6.7 CHNL_USEBURST_SET | 419 |
| 14.6.8 CHNL_USEBURST_CLR | 419 |
| 14.6.9 CHNL_REQ_MASK_SET | 420 |
| 14.6.10 CHNL_REQ_MASK_CLR..... | 420 |
| 14.6.11 CHNL_ENABLE_SET | 421 |
| 14.6.12 CHNL_ENABLE_CLR | 421 |
| 14.6.13 CHNL_PRI_ALT_SET..... | 422 |
| 14.6.14 CHNL_PRI_ALT_CLR | 423 |
| 14.6.15 CHNL_PRIORITY_SET | 424 |
| 14.6.16 CHNL_PRIORITY_CLR..... | 424 |
| 14.6.17 UDMA_BUS_ERR_CLR_REG | 424 |
| 14.6.18 UDMA_SKIP_DESC_FETCH_REG | 425 |
| 14.6.19 UDMA_DONE_STATUS_REG | 425 |
| 14.6.20 UDMA_CHANNEL_STATUS_REG..... | 428 |
| 14.6.21 UDMA_CONFIG_CTRL_REG..... | 430 |
| 15 MCU AHB High Speed Peripherals..... | 431 |

| | | |
|--------|--------------------------------|-----|
| 15.1 | CCI Controller | 431 |
| 15.1.1 | General Description | 431 |
| 15.1.2 | Features..... | 431 |
| 15.1.3 | Functional Description | 432 |
| 15.1.4 | Register Summary..... | 437 |
| 15.1.5 | Register Description..... | 438 |
| 15.2 | Ethernet Controller | 442 |
| 15.2.1 | General Description | 442 |
| 15.2.2 | Features..... | 442 |
| 15.2.3 | Functional Description | 443 |
| 15.2.4 | Register Summary..... | 464 |
| 15.2.5 | DMA Register Description | 467 |
| 15.2.6 | MAC Register Description | 487 |
| 15.2.7 | MMC Register Description..... | 504 |
| 15.3 | High Speed SPI Slave | 507 |
| 15.3.1 | General Description | 507 |
| 15.3.2 | Features..... | 507 |
| 15.3.3 | Functional Description | 507 |
| 15.3.4 | Register Summary..... | 511 |
| 15.3.5 | Register Description..... | 512 |
| 15.4 | SDIO Slave | 518 |
| 15.4.1 | General Description | 518 |
| 15.4.2 | Features..... | 518 |
| 15.4.3 | Functional Description | 519 |
| 15.4.4 | Register Summary..... | 526 |
| 15.4.5 | Register Description..... | 528 |
| 15.5 | SMIHC | 552 |
| 15.5.1 | General Description | 552 |
| 15.5.2 | Features..... | 552 |
| 15.5.3 | Functional Description | 553 |
| 15.5.4 | Register Summary..... | 564 |
| 15.5.5 | Register Description..... | 565 |
| 15.6 | USB 2.0..... | 634 |
| 15.6.1 | General Description | 634 |

| | |
|---|------------|
| 15.6.2 Features | 634 |
| 15.6.3 Functional Description | 635 |
| 15.6.4 Managing Endpoints | 656 |
| 15.6.5 Operational Model For Packet Transfers | 658 |
| 15.6.6 Managing Queue Heads | 664 |
| 15.6.7 Managing Transfers with Transfer Descriptors | 665 |
| 15.6.8 Servicing Interrupts | 667 |
| 15.6.9 HOST OPERATIONAL MODEL | 668 |
| 15.6.10 Schedule Traversal Rules | 675 |
| 15.6.11 Host Controller Pause | 713 |
| 15.6.12 Port Test Modes | 714 |
| 15.6.13 Interrupts | 714 |
| 15.6.14 Register Summary | 718 |
| 15.6.15 Register Description | 720 |
| 16 MCU APB Peripherals | 765 |
| 16.1 General Description | 765 |
| 16.2 CAN Controller | 766 |
| 16.2.1 General Description | 766 |
| 16.2.2 Features | 767 |
| 16.2.3 Functional Description | 767 |
| 16.2.4 Register Summary | 771 |
| 16.2.5 Register Description | 772 |
| 16.3 Configurable Timers | 779 |
| 16.3.1 General Description | 779 |
| 16.3.2 Features | 779 |
| 16.3.3 Functional Description | 779 |
| 16.3.4 Register Summary | 783 |
| 16.3.5 Register Description | 786 |
| 16.4 CRC Accelerator | 820 |
| 16.4.1 General Description | 820 |
| 16.4.2 Features | 820 |
| 16.4.3 Functional Description | 820 |
| 16.4.4 Register Summary | 821 |
| 16.4.5 Register Description | 822 |

| | | |
|---------|--|-----|
| 16.5 | eFuse Controller..... | 829 |
| 16.5.1 | General Description | 829 |
| 16.5.2 | Features..... | 829 |
| 16.5.3 | Functional Description | 829 |
| 16.5.4 | Register Summary..... | 835 |
| 16.5.5 | Register Description..... | 836 |
| 16.6 | Enhanced GPIO (EGPIO)..... | 840 |
| 16.6.1 | General Description | 840 |
| 16.6.2 | Features..... | 840 |
| 16.6.3 | Functional Description | 840 |
| 16.6.4 | Register Summary..... | 842 |
| 16.6.5 | Register Description..... | 844 |
| 16.7 | Generic SPI Master | 849 |
| 16.7.1 | General Description | 849 |
| 16.7.2 | Features..... | 849 |
| 16.7.3 | Functional Description | 849 |
| 16.7.4 | Register Summary..... | 851 |
| 16.7.5 | Register Description..... | 852 |
| 16.8 | Hardware Random Number Generator | 860 |
| 16.8.1 | General Description | 860 |
| 16.8.2 | Features..... | 860 |
| 16.8.3 | Functional Description | 861 |
| 16.8.4 | Register Summary..... | 861 |
| 16.8.5 | Register Description..... | 861 |
| 16.9 | I2C Master and slave | 862 |
| 16.9.1 | General Description | 862 |
| 16.9.2 | Features..... | 862 |
| 16.9.3 | Functional Description | 862 |
| 16.9.4 | Operating Modes..... | 864 |
| 16.9.5 | Register Summary..... | 866 |
| 16.9.6 | Register Description..... | 868 |
| 16.10 | I2S/PCM Master and Slave | 904 |
| 16.10.1 | General Description | 904 |
| 16.10.2 | Features..... | 905 |

| | |
|--|-------------|
| 16.10.3 Functional Description | 905 |
| 16.10.4 Programming Sequence of I2S..... | 908 |
| 16.10.5 PCM | 913 |
| 16.10.6 Register Summary..... | 914 |
| 16.10.7 Register Description..... | 915 |
| 16.11 MCU Configuration Registers | 928 |
| 16.11.1 General Description | 928 |
| 16.11.2 Features..... | 928 |
| 16.11.3 Functional Description | 928 |
| 16.11.4 Register Summary..... | 929 |
| 16.11.5 Register Description..... | 931 |
| 16.11.6 MCU HP Peripheral Interrupt Handling | 956 |
| 16.11.7 Programming Sequence for P2P Interrupt Handling..... | 958 |
| 16.12 Motor Control PWM | 958 |
| 16.12.1 General Description | 958 |
| 16.12.2 Features..... | 958 |
| 16.12.3 Functional Description | 958 |
| 16.12.4 Programing sequence..... | 960 |
| 16.12.5 Register Summary..... | 961 |
| 16.12.6 Register Description..... | 963 |
| 16.13 Quadrature Encoder | 986 |
| 16.13.1 General Description | 986 |
| 16.13.2 Features..... | 986 |
| 16.13.3 Functional Description | 986 |
| 16.13.4 Register Summary..... | 988 |
| 16.13.5 Register Description..... | 989 |
| 16.14 Serial IO | 1001 |
| 16.14.1 General Description | 1001 |
| 16.14.2 Features..... | 1001 |
| 16.14.3 Functional Description | 1002 |
| 16.14.4 SIO programming..... | 1003 |
| 16.14.5 Register Summary..... | 1004 |
| 16.14.6 Register Description..... | 1010 |
| 16.15 SSI Master | 1024 |

| | |
|--|-------------|
| 16.15.1 General Description | 1024 |
| 16.15.2 Features | 1025 |
| 16.15.3 Functional Description | 1025 |
| 16.15.4 SSI Interrupts | 1026 |
| 16.15.5 Programming sequence | 1027 |
| 16.15.6 Register Summary | 1029 |
| 16.15.7 Register Description | 1030 |
| 16.16 SSI Slave | 1042 |
| 16.16.1 General Description | 1042 |
| 16.16.2 Features | 1043 |
| 16.16.3 Functional Description | 1043 |
| 16.16.4 SSI Interrupts | 1043 |
| 16.16.5 Programming Sequence for data transfer | 1044 |
| 16.16.6 Register Summary | 1046 |
| 16.16.7 Register Description | 1047 |
| 16.17 UART | 1055 |
| 16.17.1 General Description | 1055 |
| 16.17.2 Features | 1055 |
| 16.17.3 Functional Description | 1056 |
| 16.17.4 UART Transmission | 1057 |
| 16.17.5 UART Reception | 1058 |
| 16.17.6 Interrupts | 1059 |
| 16.17.7 Auto Flow Control | 1060 |
| 16.17.8 Register Summary | 1060 |
| 16.17.9 Register Description | 1062 |
| 16.18 USART | 1098 |
| 16.18.1 General Description | 1098 |
| 16.18.2 Features | 1098 |
| 16.18.3 Functional Description | 1098 |
| 16.18.4 Procedures to use USRT | 1099 |
| 16.18.5 Register Summary | 1099 |
| 16.18.6 Register Description | 1101 |
| 17 MCU ULP Peripherals | 1129 |
| 17.1 IR_Decoder | 1129 |

| | | |
|--------|-------------------------------------|------|
| 17.1.1 | General Description | 1129 |
| 17.1.2 | Features..... | 1129 |
| 17.1.3 | Functional Description | 1129 |
| 17.1.4 | Register Summary..... | 1132 |
| 17.1.5 | Register Description..... | 1133 |
| 17.2 | Sensor Data Collector (SDC) | 1135 |
| 17.2.1 | General Description | 1135 |
| 17.2.2 | Features..... | 1135 |
| 17.2.3 | Functional Description | 1135 |
| 17.2.4 | Register Summary..... | 1137 |
| 17.2.5 | Register Description..... | 1138 |
| 17.3 | AUX ADC/DAC Controller..... | 1153 |
| 17.3.1 | General Description | 1153 |
| 17.3.2 | AUX ADC Features | 1153 |
| 17.3.3 | AUX DAC Features | 1153 |
| 17.3.4 | Register Summary..... | 1158 |
| 17.3.5 | Register Description..... | 1160 |
| 17.4 | Capacitive Touch Controller..... | 1189 |
| 17.4.1 | General Description | 1189 |
| 17.4.2 | Features..... | 1189 |
| 17.4.3 | Functional Description | 1190 |
| 17.4.4 | Programming Sequence..... | 1191 |
| 17.4.5 | Register Summary..... | 1194 |
| 17.4.6 | Register Description..... | 1194 |
| 17.5 | VAD..... | 1198 |
| 17.5.1 | General Description | 1198 |
| 17.5.2 | Features..... | 1198 |
| 17.5.3 | Functional Description | 1199 |
| 17.5.4 | Programming Sequence..... | 1199 |
| 17.5.5 | Register Summary..... | 1200 |
| 17.5.6 | Register Description..... | 1200 |
| 17.6 | FIM..... | 1204 |
| 17.6.1 | General Description | 1204 |
| 17.6.2 | Features..... | 1204 |

| | | |
|---------|------------------------------|------|
| 17.6.3 | Functional Description | 1204 |
| 17.6.4 | Operations..... | 1205 |
| 17.6.5 | Programming Sequence..... | 1207 |
| 17.6.6 | Register Summary..... | 1207 |
| 17.6.7 | Register Description..... | 1208 |
| 17.7 | Enhanced-GPIO (EGPIO) | 1211 |
| 17.7.1 | General Description | 1211 |
| 17.8 | ULP Timers | 1211 |
| 17.8.1 | General Description | 1211 |
| 17.8.2 | Features..... | 1211 |
| 17.8.3 | Block Diagram | 1212 |
| 17.8.4 | Functional Description | 1212 |
| 17.8.5 | Register Summary..... | 1213 |
| 17.8.6 | Register Description..... | 1213 |
| 17.8.7 | Programming Sequence..... | 1216 |
| 17.9 | ULP I2C | 1217 |
| 17.9.1 | General Description | 1217 |
| 17.10 | ULP UART..... | 1217 |
| 17.10.1 | General Description | 1217 |
| 17.11 | ULP I2S..... | 1217 |
| 17.11.1 | General Description | 1217 |
| 18 | UULP VBAT Peripherals | 1218 |
| 18.1 | Bandgap Top | 1218 |
| 18.1.1 | Features..... | 1218 |
| 18.1.2 | Functional Description | 1218 |
| 18.1.3 | Register Summary..... | 1219 |
| 18.1.4 | Register Description..... | 1219 |
| 18.2 | BOD | 1223 |
| 18.2.1 | General Description | 1223 |
| 18.2.2 | Features..... | 1223 |
| 18.2.3 | Block Diagram | 1223 |
| 18.2.4 | GPIO Pins | 1223 |
| 18.2.5 | Functional Description | 1224 |
| 18.2.6 | Voltage Scaler..... | 1224 |

| | | |
|---------|------------------------------|------|
| 18.2.7 | Resistor Bank (BOD)..... | 1224 |
| 18.2.8 | Input Modes..... | 1226 |
| 18.2.9 | Comparison Modes | 1226 |
| 18.2.10 | Button wakeup..... | 1226 |
| 18.2.11 | Slotting | 1227 |
| 18.2.12 | Register Summary..... | 1227 |
| 18.2.13 | Register Description..... | 1227 |
| 18.3 | Calendar | 1231 |
| 18.3.1 | General Description | 1231 |
| 18.3.2 | Features..... | 1231 |
| 18.3.3 | Functional Description | 1231 |
| 18.3.4 | Register Summary..... | 1231 |
| 18.3.5 | Register Description..... | 1232 |
| 18.4 | GPIO Timestamp | 1236 |
| 18.4.1 | General Description | 1236 |
| 18.4.2 | Features..... | 1236 |
| 18.4.3 | Programming Sequence..... | 1236 |
| 18.4.4 | Register Summary..... | 1237 |
| 18.4.5 | Register Description..... | 1237 |
| 18.5 | POC | 1238 |
| 18.5.1 | General Description | 1238 |
| 18.5.2 | Features..... | 1238 |
| 18.5.3 | Functional Description | 1238 |
| 18.5.4 | Register Summary..... | 1239 |
| 18.5.5 | Register Description..... | 1239 |
| 18.6 | Secure Storage | 1239 |
| 18.6.1 | General Description | 1239 |
| 18.6.2 | Features..... | 1239 |
| 18.6.3 | Functional Description | 1239 |
| 18.6.4 | Programming Sequence..... | 1240 |
| 18.6.5 | Secure Mode..... | 1240 |
| 18.6.6 | Register Summary..... | 1241 |
| 18.6.7 | Register Description..... | 1241 |
| 18.7 | Sleep Clock Calibrator | 1243 |

| | | |
|--------|-----------------------------------|------|
| 18.7.1 | General Description | 1243 |
| 18.7.2 | Features..... | 1243 |
| 18.7.3 | Functional Description | 1244 |
| 18.7.4 | Register Summary..... | 1245 |
| 18.7.5 | Register Description..... | 1245 |
| 18.7.6 | Programming Sequence | 1249 |
| 18.8 | WatchDog Timer (WDT)..... | 1249 |
| 18.8.1 | General Description | 1249 |
| 18.8.2 | Features..... | 1250 |
| 18.8.3 | Functional Description | 1250 |
| 18.8.4 | Register Summary..... | 1252 |
| 18.8.5 | Register Description..... | 1252 |
| 19 | Analog Peripherals..... | 1255 |
| 19.1 | Analog Comparators..... | 1255 |
| 19.1.1 | General Description | 1255 |
| 19.1.2 | Features..... | 1255 |
| 19.1.3 | Block Diagram | 1256 |
| 19.1.4 | Functional Description | 1256 |
| 19.1.5 | Input Selection | 1257 |
| 19.1.6 | Voltage Scaler..... | 1257 |
| 19.1.7 | Resistor Bank (BOD)..... | 1258 |
| 19.1.8 | Register Summary..... | 1259 |
| 19.1.9 | Register Description..... | 1259 |
| 19.2 | Analog to Digital Converter | 1260 |
| 19.2.1 | General Description | 1260 |
| 19.2.2 | Features..... | 1260 |
| 19.2.3 | Functional Description | 1260 |
| 19.2.4 | ADC channel select mode: | 1262 |
| 19.2.5 | ADC calibration mode: | 1264 |
| 19.2.6 | Channel selection | 1270 |
| 19.2.7 | Register Summary..... | 1278 |
| 19.3 | AUX_LDO..... | 1281 |
| 19.3.1 | General Description | 1281 |
| 19.3.2 | Functional Description | 1281 |

| | | |
|---------|--|------|
| 19.3.3 | Output Programming | 1281 |
| 19.3.4 | Register Summary..... | 1282 |
| 19.3.5 | Register Description..... | 1282 |
| 19.4 | Digital to Analog Converter | 1282 |
| 19.4.1 | General Description | 1282 |
| 19.4.2 | Features..... | 1283 |
| 19.4.3 | Functional Description | 1283 |
| 19.4.4 | Register Summary..... | 1283 |
| 19.4.5 | Register Description..... | 1284 |
| 19.4.6 | AUXDAC OUTPUT MUX | 1284 |
| 19.5 | OPAMPS | 1284 |
| 19.5.1 | General Description | 1284 |
| 19.5.2 | Features..... | 1284 |
| 19.5.3 | Block Diagram | 1285 |
| 19.5.4 | Functional Description | 1285 |
| 19.5.5 | Input Selection | 1285 |
| 19.5.6 | Configuring the Opamps..... | 1286 |
| 19.5.7 | Standalone mode..... | 1287 |
| 19.5.8 | Built-in modes | 1287 |
| 19.5.9 | Resistor banks..... | 1295 |
| 19.5.10 | Opamp's output on GPIO | 1296 |
| 19.5.11 | Guidelines for 'Ton' Selection | 1296 |
| 19.5.12 | Register Summary..... | 1296 |
| 19.5.13 | Register Description..... | 1296 |
| 19.6 | Temperature Sensor: RO Based and BJT Based | 1298 |
| 19.6.1 | General Description | 1298 |
| 19.6.2 | BJT Temperature Sensor..... | 1299 |
| 19.6.3 | Ring Oscillator Based Temperature Sensor..... | 1299 |
| 19.7 | Touch Capacitance sensor | 1302 |
| 19.7.1 | General Description | 1302 |
| 19.7.2 | Block Diagram | 1304 |
| 19.7.3 | Functional Description | 1304 |
| 19.7.4 | Resistor Selection | 1305 |
| 19.7.5 | Vref Selection | 1305 |

| | | |
|--------|-------------------------------------|------|
| 19.7.6 | GPIO Selection | 1305 |
| 19.7.7 | Register Summary..... | 1306 |
| 19.7.8 | Register Description..... | 1306 |
| 20 | Security Architecture | 1312 |
| 20.1 | General Description | 1312 |
| 20.2 | Features..... | 1312 |
| 20.3 | Register Summary..... | 1312 |
| 20.4 | Register Description..... | 1312 |
| 21 | In-System Programming (ISP)..... | 1313 |
| 21.1 | General Description | 1313 |
| 21.2 | Features..... | 1313 |
| 21.3 | Functional Description | 1313 |
| 21.4 | Register Summary..... | 1313 |
| 21.5 | Register Description..... | 1313 |
| 22 | Boot Process and Bootloader | 1314 |
| 22.1 | General Description | 1314 |
| 22.2 | Features..... | 1314 |
| 22.3 | Functional Description | 1314 |
| 22.4 | Secure Bootup..... | 1314 |
| 22.5 | Secure Firmware Upgrade (ISP) | 1315 |
| 22.6 | Secure Key management..... | 1315 |
| 22.7 | Secure Zone..... | 1315 |
| 22.8 | Bootloader Flowchart..... | 1315 |
| 22.8.1 | RPS Format..... | 1317 |
| 22.8.2 | Non-RPS Format | 1319 |
| 22.9 | Register Summary..... | 1319 |
| 22.10 | Register Description..... | 1319 |
| 23 | HRM Revision History..... | 1320 |

About this Document

This document is the first version of Hardware Reference Manual provided to select customers under a Non-Disclosure Agreement (NDA).

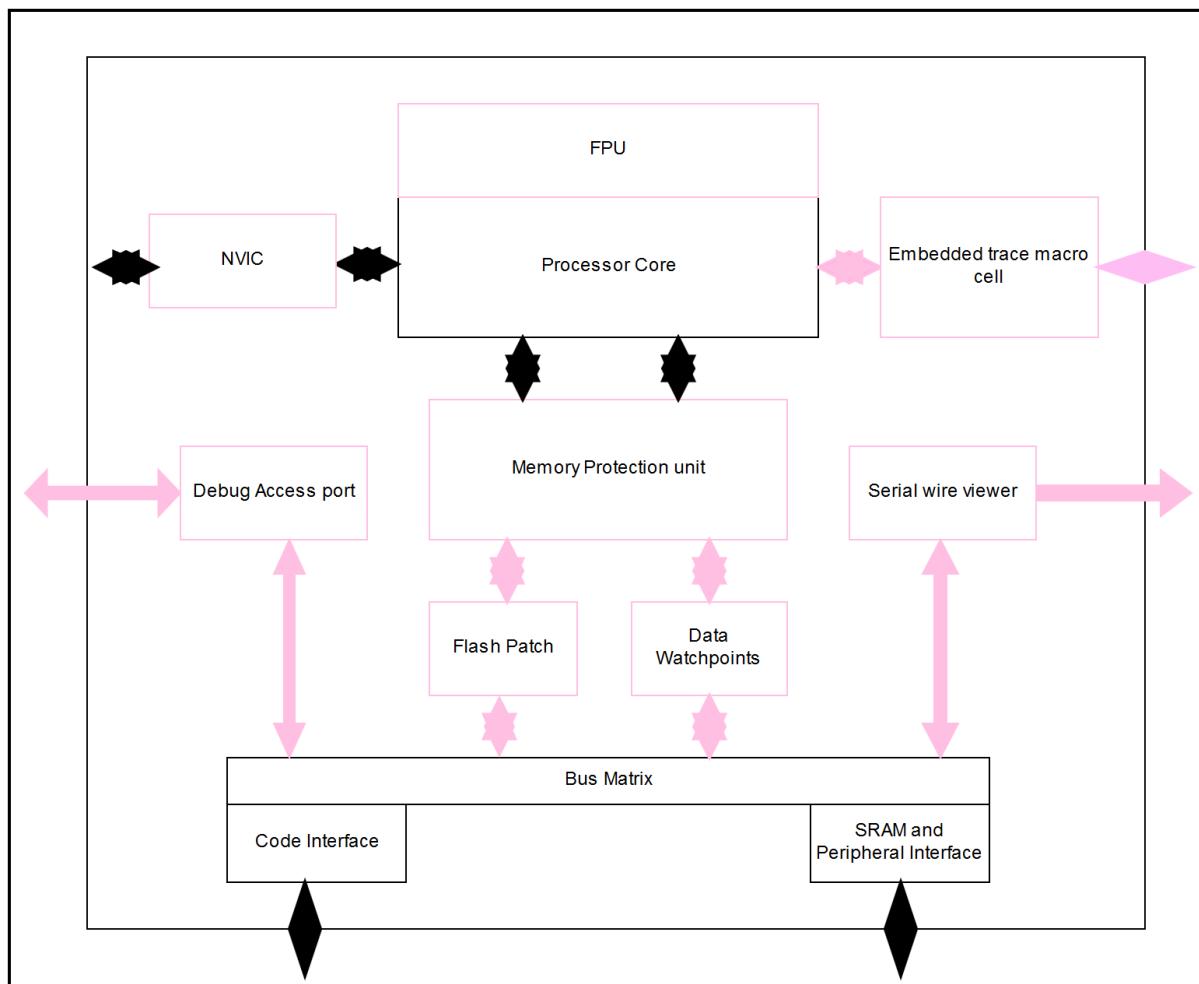
1 Cortex-M4

1.1 Cortex-M4 Introduction

1.1.1 General Description

The Redpine MCU and WiSeMCU™ families include ARM® Cortex®-M4 processor (revision r0p1) for user application. The Cortex-M4 processor is a high performance 32-bit processor designed for the microcontroller market. It offers significant benefits to developers, including:

- Outstanding processing performance combined with fast interrupt handling
- Enhanced system debug with extensive breakpoint and trace capabilities
- Efficient processor core, system and memories
- Ultra-low power consumption with integrated sleep modes
- Platform security robustness, with integrated memory protection unit (MPU).



Cortex-M4 Implementation

The Cortex-M4 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including IEEE754-compliant single-precision floating-point computation, a range of single-cycle and

SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M4 processor implements tightly-coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M4 processor implements a version of the Thumb[®] instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M4 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M4 processor closely integrates a configurable NVIC, to deliver industry-leading interrupt performance. The NVIC includes a *non-maskable interrupt* (NMI), and provides up to 64 interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of *interrupt service routines* (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to suspend load-multiple and store-multiple operations. Interrupt handlers do not require wrapping in assembler code, removing any code overhead from the ISRs. A tail-chain optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, that include a deep sleep function that enables the entire device to be rapidly powered down while still retaining program state.

1.1.2 System level interface

The Cortex-M4 processor provides multiple interfaces using AMBA[®] technology to provide high speed, low latency memory accesses. It supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks and thread-safe Boolean data handling.

The Cortex-M4 processor has a memory protection unit (MPU) that provides fine grain memory control, enabling applications to utilize multiple privilege levels, separating and protecting code, data and stack on a task-by-task basis. Such requirements are becoming critical in many embedded applications such as automotive.

1.1.3 Integrated configurable debug

The Cortex-M4 processor implements a complete hardware debug solution. This provides high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin *Serial Wire Debug* (SWD) port that is ideal for microcontrollers and other small package devices.

For system trace the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system events these generate, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.

The Flash Patch and Breakpoint Unit (FPB) provides up to 8 hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to 8 words in the program code in the CODE memory region. This enables applications stored on a non-erasable, ROM-based microcontroller to be patched if a small programmable memory, for example flash, is available in the device. During initialization, the application in ROM detects, from the programmable memory, whether a patch is required. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration, which means the program in the non-modifiable ROM can be patched.

1.1.4 Cortex-M4 processor features and benefits summary

- Thumb instruction set combines high code density with 32-bit performance
- IEEE754-compliant single-precision FPU
- Code-patch ability for ROM system updates
- Power control optimization of system components
- Integrated sleep modes for low power consumption
- Fast code execution permits slower processor clock or increases sleep mode time
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Optional *memory protection unit* (MPU) for safety-critical applications

- Extensive debug and trace capabilities:

— Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging, tracing, and code profiling.

1.1.5 Cortex-M4 core peripherals

The Cortex-M4 core includes following core peripherals:

Nested Vectored Interrupt Controller

The NVIC is an embedded interrupt controller that supports low latency interrupt processing.

System control block

The System control block (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

System timer

The system timer, SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter.

Memory protection unit

The Memory protection unit (MPU) improves system reliability by defining the memory attributes for different memory regions. It provides up to eight different regions, and an optional predefined background region.

Floating-point unit

The Floating-point unit (FPU) provides IEEE754-compliant operations on single-precision, 32-bit, floating-point values.

1.1.6 More Information

To learn about Cortex-M4 architecture, instruction set architecture (ISA) and core peripherals, visit ARM Information Center site for Cortex-M4 r0p1 revision (<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0439b/index.html>).

1.2 Cortex-M4 Processor

1.2.1 Programmers Model

This section describes the Cortex-M4 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and privilege levels for software execution and stacks.

Processor mode and privilege levels for software execution

The processor modes are:

| | |
|---------------------|--|
| Thread mode | Used to execute application software. The processor enters Thread mode when it comes out of reset. |
| Handler mode | Used to handle exceptions. The processor returns to Thread mode when it has finished all exception processing. |

The *privilege levels* for software execution are:

| | |
|---------------------|--|
| Unprivileged | The software: <ul style="list-style-type: none"> has limited access to the MSR and MRS instructions, and cannot use the CPS instruction cannot access the system timer, NVIC, or system control block might have restricted access to memory or peripherals. <i>Unprivileged software</i> executes at the unprivileged level. |
| Privileged | The software can use all the instructions and has access to all resources. |

Privileged software executes at the privileged level.

In Thread mode, the CONTROL register controls whether software execution is privileged or unprivileged, see [CONTROL register](#). In Handler mode, software execution is always privileged.

Only privileged software can write to the CONTROL register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a *supervisor call* to transfer control to privileged software.

Stacks

The processor uses a full descending stack. This means the stack pointer holds the address of the last stacked item in memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks, the *main stack* and the *process stack*, with a pointer for each held in independent registers, see [Stack Pointer](#).

In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack, see [CONTROL register](#). In Handler mode, the processor always uses the main stack. The options for processor operations are:

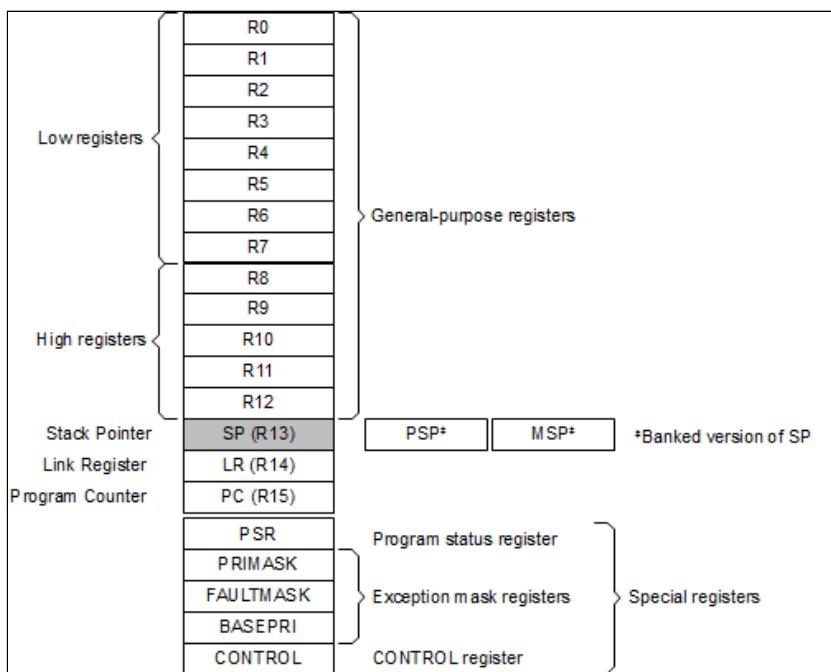
| Processor mode | Used to execute | Privilege level for software execution | Stack used |
|----------------|--------------------|---|--|
| Thread | Applications | Privileged or unprivileged ^a | Main stack or process stack ^a |
| Handler | Exception handlers | Always privileged | Main stack |

1 . Summary of processor mode, execution privilege level, and stack use options

See [CONTROL register](#).

Core registers

The processor core registers are:



1 .Core register set summary

| Name | Type ^a | Required privilege ^b | Reset value | Description |
|-------------|-------------------|---------------------------------|-----------------|-------------------------------------|
| R0-R12 | RW | Either | Unknown | General-purpose registers |
| MSP | RW | Privileged | See description | Stack Pointer |
| PSP | RW | Either | Unknown | Stack Pointer |
| LR | RW | Either | 0xFFFFFFFF | Link Register |
| PC | RW | Either | See description | Program Counter |
| PSR | RW | Privileged | 0x01000000 | Program Status Register |
| ASPR | RW | Either | Unknown | Application Program Status Register |
| IPSR | RO | Privileged | 0x00000000 | Interrupt Program Status Register |
| EPSR | RO | Privileged | 0x01000000 | Execution Program Status Register |
| PRIMASK | RW | Privileged | 0x00000000 | Priority Mask Register |
| FAULTMASKRW | Privileged | | 0x00000000 | Fault Mask Register |
| BASEPRI | RW | Privileged | 0x00000000 | Base Priority Mask Register |
| CONTROL | RW | Privileged | 0x00000000 | CONTROL register |

2 .Core register set summary

1. Describes access type during program execution in thread mode and Handler mode. Debug access can differ.

-
2. An entry of Either means privileged and unprivileged software can access the register.

General-purpose registers

R0-R12 are 32-bit general-purpose registers for data operations.

Stack Pointer

The *Stack Pointer* (SP) is register R13. In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

- 0 = *Main Stack Pointer* (MSP). This is the reset value.
- 1 = *Process Stack Pointer* (PSP).

On reset, the processor loads the MSP with the value from address 0x00000000.

Link Register

The *Link Register* (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the processor sets the LR value to 0xFFFFFFFF.

Program Counter

The *Program Counter* (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

Program Status Register

The *Program Status Register* (PSR) combines:

- *Application Program Status Register* (APSR)
- *Interrupt Program Status Register* (IPSR)
- *Execution Program Status Register* (EPSR)

These registers are mutually exclusive bitfields in the 32-bit PSR. The bit assignments are:

| | 31 | 30 | 29 | 28 | 27 | 26:25 | 24 | 23:20 | 19:16 | 15 | | 10 | 9 | 8 | | | | | | 0 |
|-------------|----------|----|----|----|--------|----------|----|----------|--------|----------|----------|----|---|------------|--|--|--|--|--|---|
| APSR | N | Z | C | V | Q | Reserved | | GE[3:0] | | Reserved | | | | | | | | | | |
| IPSR | Reserved | | | | | | | | | | | | | ISR_NUMBER | | | | | | |
| EPSR | Reserved | | | | ICI/IT | T | | Reserved | ICI/IT | | Reserved | | | | | | | | | |

Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- read all of the registers using PSR with the MRS instruction
- write to the APSR N, Z, C, V, and Q bits using APSR_nzcvq with the MSR instruction.

The PSR combinations and attributes are:

| Register | Type | Combination |
|----------|-----------------|----------------------|
| PSR | RWa, b | APSR, EPSR, and IPSR |
| IEPSR | RO | EPSR and IPSR |
| IAPSR | RW ^a | APSR and IPSR |
| EAPSR | RW ^b | APSR and EPSR |

3 . PSR register combinations

- a. The processor ignores writes to the IPSR bits.
- b. Reads of the EPSR bits return zero, and the processor ignores writes to these bits

See the instruction descriptions [MRS](#) and [MSR](#) for more information about how to access the program status registers.

Application Program Status Register

The APSR contains the current state of the condition flags from previous instruction executions. See the register summary in [Table 2-2](#) for its attributes. The bit assignments are:

| Bits | Name | Function |
|---------|---------|--|
| [31] | N | Negative flag |
| [30] | Z | Zero flag |
| [29] | C | Carry or borrow flag |
| [28] | V | Overflow flag |
| [27] | Q | DSP overflow and saturation flag |
| [26:20] | - | Reserved |
| [19:16] | GE[3:0] | Greater than or Equal flags. See SEL for more information. |
| [15:0] | - | Reserved |

4 . APSR bit assignments

Interrupt Program Status Register

The IPSR contains the exception type number of the current *Interrupt Service Routine* (ISR). See the register summary in [Table 2-2](#) for its attributes. The bit assignments are:

| Bits | Name | Function |
|--------|------|----------|
| [31:9] | - | Reserved |

| Bits | Name | Function |
|-------|------------|---|
| [8:0] | ISR_NUMBER | <p>This is the number of the current exception:</p> <ul style="list-style-type: none"> 0 = Thread mode 1 = Reserved 2 = NMI 3 = HardFault 4 = MemManage 5 = BusFault 6 = UsageFault 7-10 = Reserved 11 = SVCall 12 = Reserved for Debug 13 = Reserved 14 = PendSV 15 = SysTick 16 = IRQ0 . . . 98 = IRQ98 see Exception types for more information. |

5 . IPSR bit assignments

Execution Program Status Register

The EPSR contains the Thumb state bit, and the execution state bits for either the:

- *If-Then* (IT) instruction
- *Interruptible-Continuable Instruction* (ICI) field for an interrupted load multiple or store multiple instruction.

See the register summary in the below for the EPSR attributes. The bit assignments are:

| Bits | Name | Function |
|------------------|------|--|
| [31:27] | - | Reserved. |
| [26:25], [15:10] | ICI | Interruptible-continuable instruction bits, see Interruptible-continuable instructions . |
| [24] | T | Thumb state bit, see Thumb state . |
| [23:16] | - | Reserved. |
| [9:0] | - | Reserved. |

6 . EPSR bit assignments

[26:25], [15:10]IT Indicates the execution state bits of the IT instruction, see [IT](#).

Attempts to read the EPSR directly through application software using the MSR instruction always return zero.
Attempts to write the EPSR using the MSR instruction in application software are ignored.

Interruptible-continuable instructions

When an interrupt occurs during the execution of an LDM, STM, PUSH, POP, VLDM, VSTM, VPUSH, or VPOP instruction, the processor:

- stops the load multiple or store multiple instruction operation temporarily •stores the next register operand in the multiple operation to EPSR bits[15:12].

After servicing the interrupt, the processor:

- returns to the register pointed to by bits[15:12]
- resumes execution of the multiple load or store instruction.

When the EPSR holds ICI execution state, bits[26:25,11:10] are zero.

If-Then block

The If-Then block contains up to four instructions following an IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See [IT](#) for more information.

Thumb state

The Cortex-M4 processor only supports execution of instructions in Thumb state. The following can clear the T bit to 0:

- instructions BLX, BX and POP{PC}
- restoration from the stacked xPSR value on an exception return
- bit[0] of the vector value on an exception entry or reset.

Attempting to execute instructions when the T bit is 0 results in a fault or lockup. See [Lockup](#) for more information.

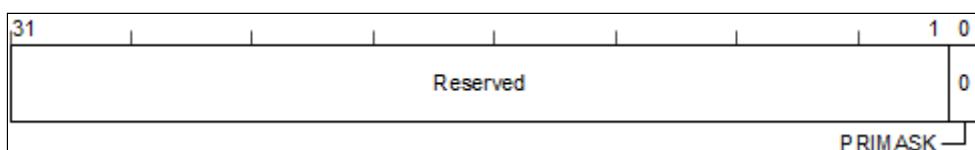
Exception mask registers

The exception mask registers disable the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks.

To access the exception mask registers use the MSR and MRS instructions, or the CPS instruction to change the value of PRIMASK or FAULTMASK. See [MRS](#), [MSR](#), and [CPS](#) for more information.

Priority Mask Register

The PRIMASK register prevents activation of all exceptions with configurable priority. See the register summary in [Table 2-2](#) for its attributes. The bit assignments are:



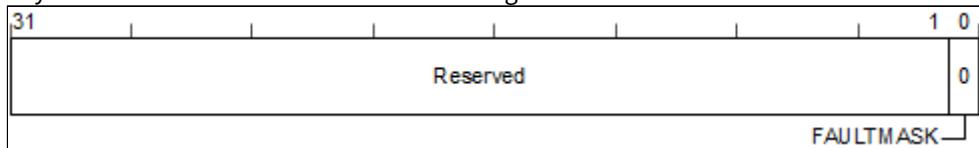
| Bits | Name | Function |
|--------|------|----------|
| [31:1] | - | Reserved |

| Bits | Name | Function |
|------|---------|--|
| [0] | PRIMASK | 1. = no effect 2. = prevents the activation of all exceptions with configurable priority. |

7 . PRIMASK register bit assignments

Fault Mask Register

The FAULTMASK register prevents activation of all exceptions except for *Non-Maskable Interrupt* (NMI). See the register summary in [Table 2-2](#) for its attributes. The bit assignments are:



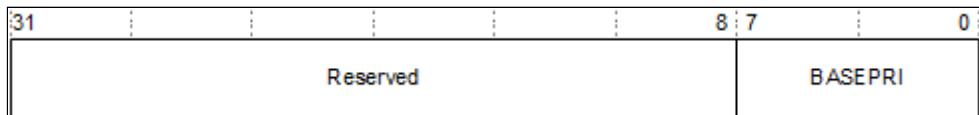
| Bits | Name | Function |
|--------|-----------|--|
| [31:1] | - | Reserved |
| [0] | FAULTMASK | 1. = no effect 2. = prevents the activation of all exceptions except for NMI. |

8 . FAULTMASK register bit assignments

The processor clears the FAULTMASK bit to 0 on exit from any exception handler except the NMI handler.

Base Priority Mask Register

The BASEPRI register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the BASEPRI value. See the register summary in [Table 2-2](#) for its attributes. The bit assignments are:



| Bits | Name | Function |
|--------|------|----------|
| [31:8] | - | Reserved |

| Bits | Name | Function |
|-------|----------------------|---|
| [7:0] | BASEPRI ^a | <p>Priority mask bits: 0x00 = no effect Nonzero = defines the base priority for exception processing.</p> <p>The processor does not process any exception with a priority value greater than or equal to BASEPRI.</p> |

a. This field is similar to the priority fields in the interrupt priority registers. The device implements only bits[7:2] of this field,
bits[1:0] read as zero and ignore writes. See [Interrupt Priority Registers](#) for more information. Remember that higher priority field values correspond to lower exception priorities.

9 . BASEPRI register bit assignments

CONTROL register

The CONTROL register controls the stack used and the privilege level for software execution when the processor is in Thread mode and indicates whether the FPU state is active. See the register summary in [Table 2-2](#) for its attributes. The bit assignments are:

| Bits | Name | Function |
|--------|-------|--|
| [31:3] | - | Reserved |
| [2] | FPCA | Indicates whether floating-point context currently active: 0 = No floating-point context active 1 = Floating-point context active. The Cortex-M4 uses this bit to determine whether to preserve floating-point state when processing an exception. |
| [1] | SPSEL | Defines the currently active stack pointer: In Handler mode this bit reads as zero and ignores writes. The Cortex-M4 updates this bit automatically on exception return. 0 = MSP is the current stack pointer 1 = PSP is the current stack pointer. |
| [0] | nPRIV | Defines the Thread mode privilege level: 1. = Privileged 2. = Unprivileged. |

10 . CONTROL register bit assignments

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms automatically update the CONTROL register based on the [EXC_RETURN](#) value, see [Table 2-17](#).

In an OS environment, ARM recommends that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack.

By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, either:

- use the MSR instruction to set the Active stack pointer bit to 1, see [MSR](#).
- perform an exception return to Thread mode with the appropriate EXC_RETURN value, see [Table 2-17](#).

Note

When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction. This ensures that instructions after the ISB instruction execute using the new stack pointer. See [ISB](#)

Exceptions and interrupts

The Cortex-M4 processor supports interrupts and system exceptions. The processor and the NVIC prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See [Exception entry](#) and [Exception return](#) for more information.

The NVIC registers control interrupt handling. See [Nested Vectored Interrupt Controller](#) for more information.

Data types

The processor:

- supports the following data types:
 - 32-bit words
 - 16-bit halfwords
 - 8-bit bytes
- manages all data memory accesses as little-endian or big-endian. Instruction memory and *Private Peripheral Bus* (PPB) accesses are always performed as little-endian. See [Memory regions](#), [types and attributes](#) for more information.

The Cortex Microcontroller Software Interface Standard

For a Cortex-M4 microcontroller system, the *Cortex Microcontroller Software Interface Standard* (CMSIS) defines:

- a common way to:
 - access peripheral registers
 - define exception vectors
- the names of:
 - the registers of the core peripherals
 - the core exception vectors
- a device-independent interface for RTOS kernels, including a debug channel.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M4 processor. CMSIS simplifies software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Note

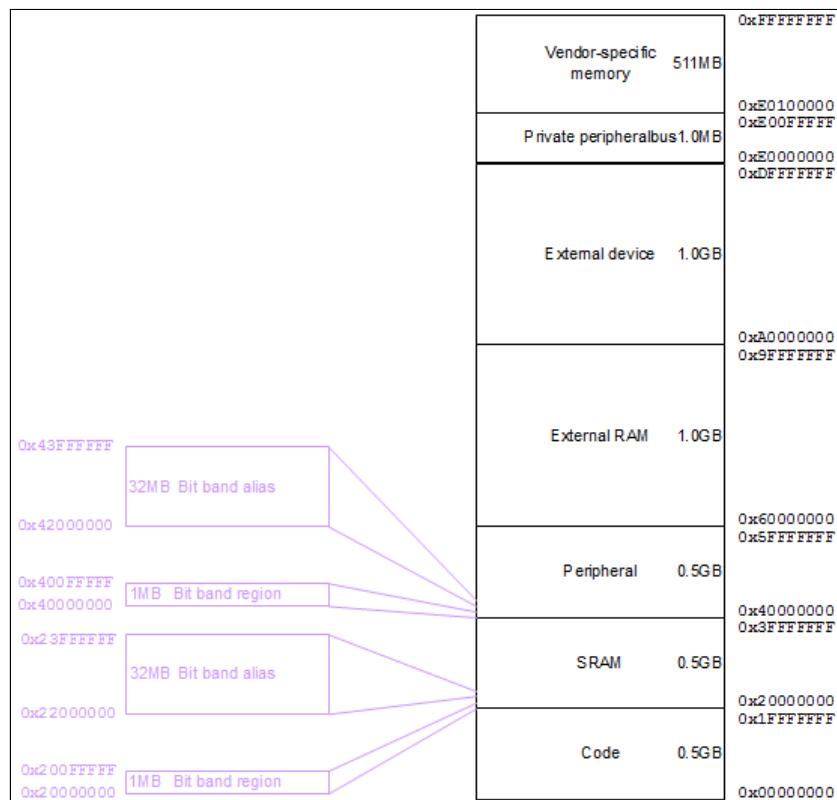
This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- Power management programming hints
- CMSIS functions
- Accessing the Cortex-M4 NVIC registers using CMSIS
- NVIC programming hints

1.2.2 Memory model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed default memory map that provides up to 4GB of addressable memory. The memory map is:



The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data, see [Bit-banding](#).

The processor reserves regions of the *Private peripheral bus (PPB)* address range for core peripheral registers.

Memory regions, types and attributes

The memory map and the programming of the MPU splits the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

Normal The processor can re-order transactions for efficiency, or perform speculative reads.

Device The processor preserves transaction order relative to other transactions to Device or Strongly-ordered memory.

Strongly-ordered The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly-ordered memory mean that the memory system can buffer a write to Device memory, but must not buffer a write to Strongly-ordered memory.

The additional memory attributes include:

Shareable: For a shareable memory region, the memory system provides data synchronization between bus masters in a system with multiple bus masters, for example, a processor with a DMA controller.

Strongly-ordered memory is always shareable.

If multiple bus masters can access a non-shareable memory region, software must ensure data coherency between the bus masters.

Execute Never (XN): Means the processor prevents instruction accesses. A fault exception is generated only on execution of an instruction executed from an XN region.

Memory system ordering of memory accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing this does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions, see [Software ordering of memory accesses](#).

However, the memory system does guarantee some ordering of accesses to Device and Strongly-ordered memory. For two memory access instructions A1 and A2, if A1 occurs before A2 in program order, the ordering of the memory accesses caused by two instructions is:

| A1 | A2 | Normal access | Device access | | Strongly-ordered access |
|------------------------------|----|---------------|---------------|-----------|-------------------------|
| | | | Non-shareable | Shareable | |
| Normal access | - | - | - | - | - |
| Device access, non-shareable | - | < | - | - | < |
| Device access, shareable | - | - | < | < | < |
| Strongly-ordered access | - | < | < | < | < |

Where:

- Means that the memory system does not guarantee the ordering of the accesses.

< Means that accesses are observed in program order, that is, A1 is always observed before A2.

Behavior of memory accesses

The behavior of accesses to each region in the memory map is:

| Address range | Memory region | Memory type ^a | X N ^a | Description |
|-----------------------|---------------|--------------------------|------------------|--|
| 0x00000000-0x1FFFFFFF | Code | Normal | - | Executable region for program code. You can also put data here. |
| 0x20000000-0x3FFFFFFF | SRAM | Normal | - | Executable region for data. You can also put code here. This region includes bit band and bit band alias areas, see Table 2-13 . |
| 0x40000000-0x5FFFFFFF | Peripheral | Device | XN | This region includes bit band and bit band alias areas, see Table 2-14 . |

| Address range | Memory region | Memory type ^a | X N ^a | Description |
|-----------------------|------------------------|--------------------------|------------------|--|
| 0x60000000-0x9FFFFFFF | External RAM | Normal | - | Executable region for data. |
| 0xA0000000-0xDFFFFFFF | External device | Device | XN | External Device memory. |
| 0xE0000000-0xE00FFFFF | Private Peripheral Bus | Strongly-ordered | XN | This region includes the NVIC, System timer, and system control block. |
| 0xE0100000-0xFFFFFFFF | Vendor-specific device | Device | XN | Accesses to this region are to vendor-specific peripherals. |

11 . Memory access behavior

a.See [Memory regions, types and attributes](#) for more information.

The Code, SRAM, and external RAM regions can hold programs. However, ARM recommends that programs always use the Code region. This is because the processor has separate buses that enable instruction fetches and data accesses to occur simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see [Memory protection unit](#).

Additional memory access constraints for caches and shared memory

When a system includes caches or shared memory, some memory regions have additional access constraints, and some regions are subdivided, as Table 2-12 shows:

| Address range | Memory region | Memory type | Shareability | Cache policy |
|------------------------|------------------------|--------------------------------|----------------------------|-------------------|
| 0x00000000- 0x1FFFFFFF | Code | Normal ^a | - | WT ^b |
| 0x20000000- 0x3FFFFFFF | SRAM | Normal ^a | - | WBWA ^b |
| 0x40000000- 0x5FFFFFFF | Peripheral | Device ^a | - | - |
| 0x60000000- 0x7FFFFFFF | External RAM | Normal ^a | - | WBWA ^b |
| 0x80000000- 0x9FFFFFFF | | | | WT ^b |
| 0xA0000000- 0xBFFFFFFF | External device | Device ^a | Shareable ^a | |
| 0xC0000000- 0xDFFFFFFF | | | Non-shareable ^a | |
| 0xE0000000- 0xE00FFFFF | Private Peripheral Bus | Strongly- ordered ^a | Shareable ^a | - |
| 0xE0100000- 0xFFFFFFFF | Vendor-specific device | Device ^a | - | - |

12 . Memory region shareability and cache policies

a - See [Memory regions, types and attributes](#) on for more information.

b - WT = Write through, no write allocate. WBWA = Write back, write allocate. See the [Glossary](#) for more information.

Instruction prefetch and branch prediction

The Cortex-M4 processor:

- prefetches instructions ahead of execution
- speculatively prefetches from branch target addresses.

Software ordering of memory accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:

- the processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- the processor has multiple bus interfaces
- memory or devices in the memory map have different wait states
- some memory accesses are buffered or speculative.

[Memory system ordering of memory accesses](#) describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:

| | |
|------------|---|
| DMB | The <i>Data Memory Barrier</i> (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions. See DMB . |
| DSB | The <i>Data Synchronization Barrier</i> (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute. See DSB . |
| ISB | The <i>Instruction Synchronization Barrier</i> (ISB) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions. See ISB . |

MPU programming

Use a DSB followed by an ISB instruction or exception return to ensure that the new MPU configuration is used by subsequent instructions.

Bit-banding

A bit-band region maps each word in a *bit-band alias* region to a single bit in the *bit-band region*. The bit-band regions occupy the lowest 1MB of the SRAM and peripheral memory regions.

The memory map has two 32MB alias regions that map to two 1MB bit-band regions:

- accesses to the 32MB SRAM alias region map to the 1MB SRAM bit-band region, as shown in [Table 2-13](#)
- accesses to the 32MB peripheral alias region map to the 1MB peripheral bit-band region, as shown in [Table 2-14](#).

| Address range | Memory region | Instruction and data accesses |
|-----------------------|----------------------|--|
| 0x20000000-0x200FFFFF | SRAM bit-band region | Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias. |

| Address range | Memory region | Instruction and data accesses |
|-----------------------|---------------------|---|
| 0x22000000-0x23FFFFFF | SRAM bit-band alias | Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped. |

13 . SRAM memory bit-banding regions

| Address range | Memory region | Instruction and data accesses |
|-----------------------|----------------------------|--|
| 0x40000000-0x400FFFFF | Peripheral bit-band alias | Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias. |
| 0x42000000-0x43FFFFFF | Peripheral bit-band region | Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted. |

14 . Peripheral memory bit-banding regions

Note

- A word access to the SRAM or peripheral bit-band alias regions maps to a single bit in the SRAM or peripheral bit-band region
- Bit band accesses can use byte, halfword, or word transfers. The bit band transfer size matches the transfer size of the instruction making the bit band access.

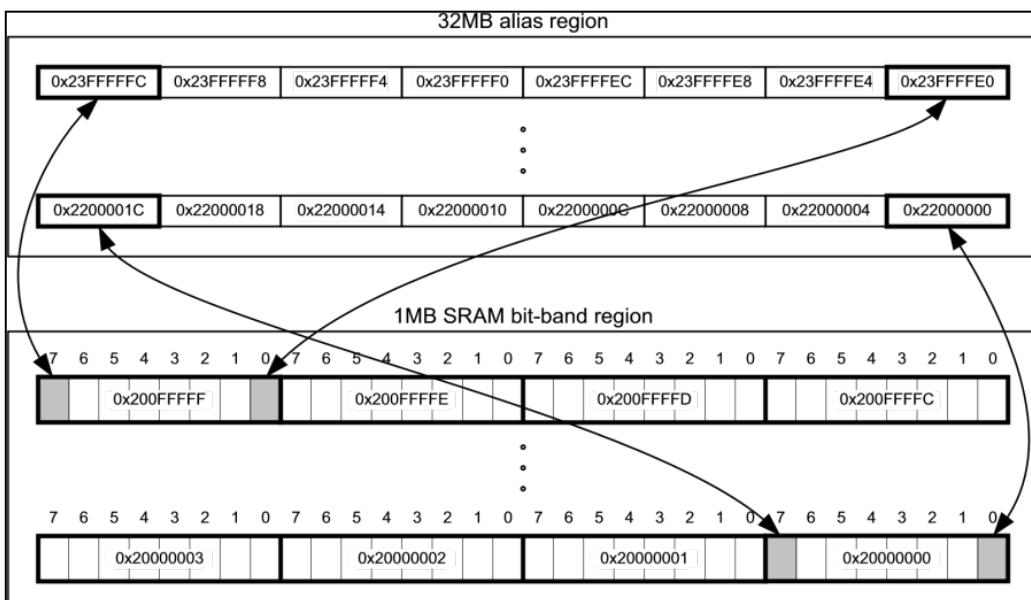
The following formula shows how the alias region maps onto the bit-band region:

$\text{bit_word_offset} = (\text{byte_offset} \times 32) + (\text{bit_number} \times 4)$ $\text{bit_word_addr} = \text{bit_band_base} + \text{bit_word_offset}$ where:

- Bit_word_offset is the position of the target bit in the bit-band memory region.
- Bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- Bit_band_base is the starting address of the alias region.
- Byte_offset is the number of the byte in the bit-band region that contains the targeted bit.
- Bit_number is the bit position, 0-7, of the targeted bit.

Figure 3 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FFFFFFE0 maps to bit[0] of the bit-band byte at 0x200FFFFFF: $0x23FFFFFFE0 = 0x22000000 + (0xFFFF * 32) + (0 * 4)$.
- The alias word at 0x23FFFFFFFC maps to bit[7] of the bit-band byte at 0x200FFFFFF: $0x23FFFFFFFC = 0x22000000 + (0xFFFF * 32) + (7 * 4)$.
- The alias word at 0x2200000000 maps to bit[0] of the bit-band byte at 0x2000000000: $0x2200000000 = 0x22000000 + (0 * 32) + (0 * 4)$.
- The alias word at 0x220000001C maps to bit[7] of the bit-band byte at 0x2000000000: $0x220000001C = 0x22000000 + (0 * 32) + (7 * 4)$.



2 . Bit-band mapping

Directly accessing an alias region

Writing to a word in the alias region updates a single bit in the bit-band region. Bit[0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit[0] set to 1 writes a 1 to the bit-band bit, and writing a value with bit[0] set to 0 writes a 0 to the bit-band bit. Bits[31:1] of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E. Reading a word in the alias region:

- 0x00000000 indicates that the targeted bit in the bit-band region is set to zero
- 0x00000001 indicates that the targeted bit in the bit-band region is set to 1

Directly accessing a bit-band region

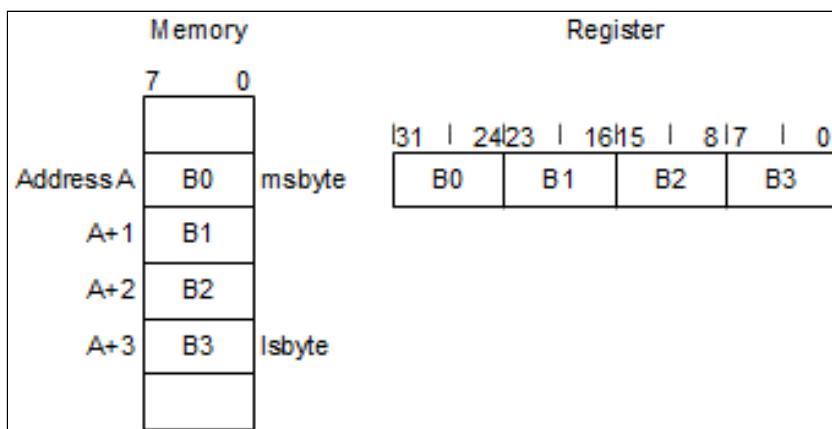
[Behavior of memory accesses](#) describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

Memory endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. *Byte-invariant big-endian format* or *Little-endian format* describes how words of data are stored in memory.

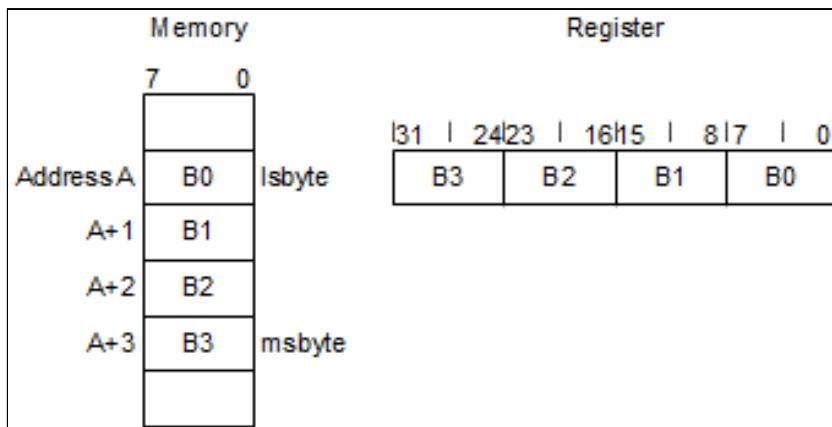
Byte-invariant big-endian format

In byte-invariant big-endian format, the processor stores the most significant byte of a word at the lowest-numbered byte, and the least significant byte at the highest-numbered byte. For example:



Little-endian format

In little-endian format, the processor stores the least significant byte of a word at the lowest-numbered byte, and the most significant byte at the highest-numbered byte. For example:



Synchronization primitives

The Cortex-M4 instruction set includes pairs of *synchronization primitives*. These provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use them to perform a guaranteed read-modify-write memory update sequence, or for a semaphore mechanism. A pair of synchronization primitives comprises:

A Load-Exclusive instruction

Used to read the value of a memory location, requesting exclusive access to that location.

A Store-Exclusive instruction

Used to attempt to write to the same memory location, returning a status bit to a register. If this bit is:

1. It indicates that the thread or process gained exclusive access to the memory, and the write succeeds.
2. It indicates that the thread or process did not gain exclusive access to the memory, and no write was performed.

The pairs of Load-Exclusive and Store-Exclusive instructions are:

- the word instructions LDREX and STREX
- the halfword instructions LDREXH and STREXH

- the byte instructions LDREXB and STREXB.

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction. To perform an exclusive read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Modify the value, as required.
3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location.
4. Test the returned status bit. If this bit is:
 - a. The read-modify-write completed successfully.
 - b. No write was performed. This indicates that the value returned at step 1 might be out of date. The software must retry the entire read-modify-write sequence.

Software can use the synchronization primitives to implement a semaphores as follows:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.
3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.

The Cortex-M4 includes an exclusive access monitor, that tags the fact that the processor has executed a Load-Exclusive instruction.

The processor removes its exclusive access tag if:

- It executes a CLREX instruction.
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs. This means the processor can resolve semaphore conflicts between different threads.

For more information about the synchronization primitive instructions, see [LDREX and STREX](#) and [CLREX](#).

Programming hints for the synchronization primitives

ISO/IEC C cannot directly generate the exclusive access instructions. CMSIS provides intrinsic functions for generation of these instructions:

| Instruction | CMSIS function |
|-------------|--|
| LDREX | uint32_t __LDREXW (uint32_t *addr) |
| LDREXH | uint16_t __LDREXH (uint16_t *addr) |
| LDREXB | uint8_t __LDREXB (uint8_t *addr) |
| STREX | uint32_t __STREXW (uint32_t value, uint32_t *addr) |
| STREXH | uint32_t __STREXH (uint16_t value, uint16_t *addr) |
| STREXB | uint32_t __STREXB (uint8_t value, uint8_t *addr) |
| CLREX | void __CLREX (void) |

15 . CMSIS functions for exclusive access instructions

For example:

```
uint16_t value;  
uint16_t *address = 0x20001002;  
value = __LDREXH (address); // load 16-bit value from memory address 0x20001002
```

1.2.3 Exception model

This section describes the exception model. It describes:

- [Exception states](#)
- [Exception types](#)
- [Exception handlers](#)
- [Vector table](#)
- [Exception priorities](#)
- [Interrupt priority grouping](#)
- [Exception entry and return](#)

Exception states

Each exception is in one of the following states:

| | |
|-----------------|--|
| Inactive | The exception is not active and not pending. |
| Pending | The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending. |
| Active | An exception that is being serviced by the processor but has not completed. Note An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state. |

Active and pending

The exception is being serviced by the processor and there is a pending exception from the same source.

Exception types

The exception types are:

| | |
|------------------|---|
| Reset | Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode. |
| NMI | A <i>NonMaskable Interrupt</i> (NMI) can be signalled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of -2. NMIs cannot be: <ul style="list-style-type: none"> • masked or prevented from activation by any other exception • preempted by any exception other than Reset. |
| HardFault | A HardFault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. HardFaults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority. |

MemManage

A MemManage fault is an exception that occurs because of a memory protection related fault. The MPU or the fixed memory protection constraints determines this fault, for both instruction and data memory transactions. This fault is always used to abort instruction accesses to *Execute Never* (XN) memory regions.

| | |
|------------------------|---|
| BusFault | A BusFault is an exception that occurs because of a memory related fault for an instruction or data memory transaction. This might be from an error detected on a bus in the memory system. |
| UsageFault | <p>A UsageFault is an exception that occurs because of a fault related to instruction execution. This includes:</p> <ul style="list-style-type: none"> • an undefined instruction • an illegal unaligned access • invalid state on instruction execution • an error on exception return. <p>The following can cause a UsageFault when the core is configured to report them:</p> <ul style="list-style-type: none"> • an unaligned address on word and halfword memory access • division by zero. |
| SVCall | A <i>supervisor call</i> (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers. |
| PendSV | PendSV is an interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. |
| SysTick | A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as system tick. |
| Interrupt (IRQ) | A interrupt, or IRQ, is an exception signalled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor. |

17.

| Exception number ^a | IRQ number ^a | Exception type | Priority | Vector address or offset ^b | Activation |
|-------------------------------|-------------------------|----------------|---------------------------|---------------------------------------|---|
| 1 | - | Reset | -3, the highest | 0x00000004 | Asynchronous |
| 2 | -14 | NMI | -2 | 0x00000008 | Asynchronous |
| 3 | -13 | HardFault | -1 | 0x0000000C | - |
| 4 | -12 | MemManage | Configurable ^c | 0x00000010 | Synchronous |
| 5 | -11 | BusFault | Configurable ^c | 0x00000014 | Synchronous when precise, asynchronous when imprecise |
| 6 | -10 | UsageFault | Configurable ^c | 0x00000018 | Synchronous |

| Exception number ^a | IRQ number ^a | Exception type | Priority | Vector address or offset ^b | Activation |
|-------------------------------|-------------------------|-----------------|---------------------------|---------------------------------------|--------------|
| 10:7 | | Reserved | | | |
| 11 | -5 | SVCALL | Configurable ^c | 0x0000002C | Synchronous |
| 12-13 | - | Reserved | - | - | - |
| 14 | -2 | PendSV | Configurable ^c | 0x00000038 | Asynchronous |
| 15 | -1 | SysTick | Configurable ^c | 0x0000003C | Asynchronous |
| 16 and above | 0 and above | Interrupt (IRQ) | Configurable ^d | 0x00000040 and above ^e | Asynchronous |

18 . Properties of the different exception types

a. To simplify the software layer, the CMSIS only uses IRQ numbers and therefore uses negative values for exceptions other than interrupts. The IPSR returns the Exception number, see [Interrupt Program Status Register](#). See *Vector table* for more information.

1. See [System Handler Priority Registers](#)
2. See [Interrupt Priority Registers](#)
3. Increasing in steps of 4.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that [Table 2-16](#) shows as having configurable priority, see:

- [System Handler Control and State Register](#).
- [Interrupt Clear-enable Registers](#).

For more information about HardFaults, MemManage faults, BusFaults, and UsageFaults, see [Fault handling](#).

Exception handlers

The processor handles exceptions using:

Interrupt Service Routines (ISRs)

Interrupts IRQ0 to IRQ98 are the exceptions handled by ISRs.

Fault handlers HardFault, MemManage fault, UsageFault, and BusFault are fault exceptions handled by the fault handlers.

System handlers NMI, PendSV, SVCALL SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

Vector table

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. Figure 4 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code, see [Thumb state](#).

| Exception number | IRQ number | Offset | Vector |
|------------------|------------|--------|-------------------------|
| 255 | 239 | 0x03FC | IRQ 239 |
| . | . | . | . |
| . | . | 0x004C | |
| 18 | 2 | 0x0048 | IRQ2 |
| 17 | 1 | 0x0044 | IRQ1 |
| 16 | 0 | 0x0040 | IRQ0 |
| 15 | -1 | 0x003C | Systick |
| 14 | -2 | 0x0038 | PendSV |
| 13 | | | Reserved |
| 12 | | | Reserved for Debug |
| 11 | -5 | 0x002C | SVCall |
| 10 | | | |
| 9 | | | Reserved |
| 8 | | | |
| 7 | | | |
| 6 | -10 | 0x0018 | Usage fault |
| 5 | -11 | 0x0014 | Bus fault |
| 4 | -12 | 0x0010 | Memory management fault |
| 3 | -13 | 0x000C | Hard fault |
| 2 | -14 | 0x0008 | NMI |
| 1 | | 0x0004 | Reset |
| | | 0x0000 | Initial SP value |

3 . Vector table

On system reset, the vector table is fixed at address 0x00000000. Privileged software can write to the VTOR to relocate the vector table start address to a different memory location, in the range 0x00000080 to 0x3FFFF80, see [Vector Table Offset Register](#).

Exception priorities

As Table 2-16 shows, all exceptions have an associated priority, with:

- a lower priority value indicating a higher priority
- configurable priorities for all exceptions except Reset, HardFault, and NMI.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see

- [System Handler Priority Registers](#).
- [Interrupt Priority Registers](#).

Note

Configurable priority values are in the range 0-63. This means that the Reset, HardFault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

Interrupt priority grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields:

- an upper field that defines the *group priority*
- a lower field that defines a *subpriority* within the group.

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler,

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see [Application Interrupt and Reset Control Register](#).

Exception entry and return

Descriptions of exception handling use the following terms:

| | |
|----------------------|---|
| Preemption | When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See Interrupt priority grouping for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See Exception entry for more information. |
| Return | This occurs when the exception handler is completed, and: <ul style="list-style-type: none">• there is no pending exception with sufficient priority to be serviced• the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See Exception return for more information. |
| Tail-chaining | This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler. |

19.

Late-arriving

This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

Exception entry

Exception entry occurs when there is a pending exception with sufficient priority and either:

- the processor is in Thread mode
- the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

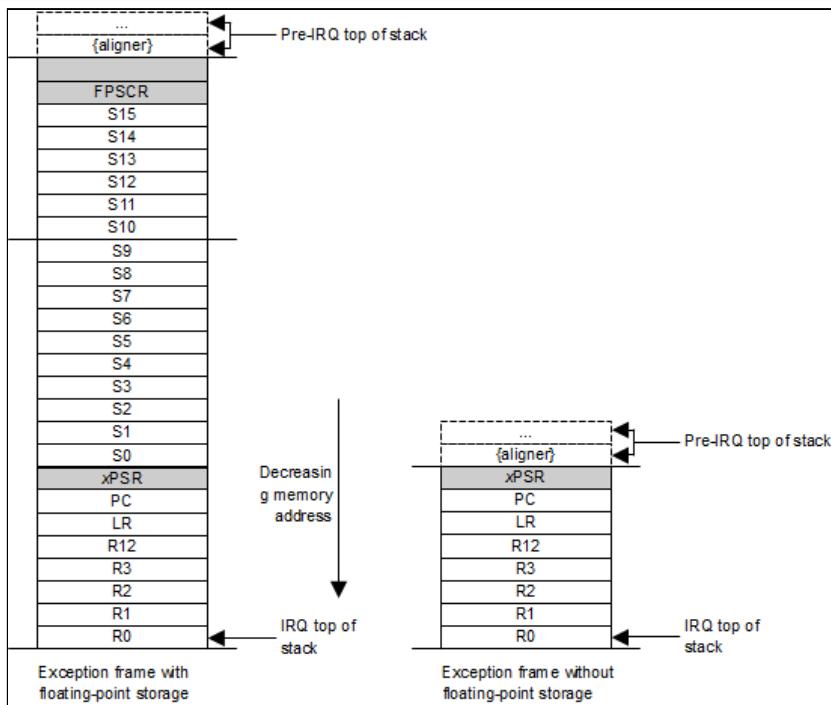
Sufficient priority means the exception has more priority than any limits set by the mask registers, see [Exception mask registers](#). An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as *stacking* and the structure of eight data words is referred as the *stack frame*.

When using floating-point routines, the Cortex-M4 processor automatically stacks the architected floating-point state on exception entry. Figure 5 shows the Cortex-M4 stack frame layout when floating-point state is preserved on the stack as the result of an interrupt or an exception.

Note

Where stack space for floating-point state is not allocated, the stack frame is the same as that of ARMv7-M implementations without an FPU. Figure 5 shows this stack frame also.



4 . Exception stack frame

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. The alignment of the stack frame is controlled via the STKALIGN bit of the *Configuration Control Register* (CCR).

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes. In parallel to the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred. If no higher priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active. If another higher priority exception occurs during exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.

Exception return

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC_RETURN value into the PC:

- an LDM or POP instruction that loads the PC
- an LDR instruction with PC as the destination
- a BX instruction using any register.

EXC_RETURN is the value loaded into the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest five bits of this value provide information on the return stack and processor mode. Table 2-17 shows the EXC_RETURN values with a description of the exception return behavior.

All EXC_RETURN values have bits[31:5] set to one. When this value is loaded into the PC it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

| EXC_RETURN[31 :0] | Description |
|-------------------|--|
| 0xFFFFFFFF1 | Return to Handler mode, exception return uses non-floating-point state from the MSP and execution uses MSP after return. |
| 0xFFFFFFFF9 | Return to Thread mode, exception return uses non-floating-point state from MSP and execution uses MSP after return. |
| 0xFFFFFFFFD | Return to Thread mode, exception return uses non-floating-point state from the PSP and execution uses PSP after return. |
| 0xFFFFFE1 | Return to Handler mode, exception return uses floating-point-state from MSP and execution uses MSP after return. |
| 0xFFFFFE9 | Return to Thread mode, exception return uses floating-point state from MSP and execution uses MSP after return. |
| 0xFFFFFED | Return to Thread mode, exception return uses floating-point state from PSP and execution uses PSP after return. |

20 . Exception return behavior

1.2.4 Fault handling

Faults are a subset of the exceptions, see [Exception model](#). Faults are generated by:

- a bus error on:
—an instruction fetch or vector table load —a data access.
 - an internally-detected error such as an undefined instruction
 - attempting to execute an instruction from a memory region marked as *Non-Executable* (XN).
 - a privilege violation or an attempt to access an unmanaged region causing an MPU fault.

Fault types

Table 2-18 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates that the fault has occurred. See [Configurable Fault Status Register](#) for more information about the fault status registers.

| Fault | Handler | Bit name | Fault status register |
|--|------------|-------------------------|---|
| Bus error on a vector read | HardFault | VECTTBL | <i>HardFault Status Register</i> |
| Fault escalated to a hard fault | | FORCED | |
| MPU or default memory map mismatch: | MemManage | - | - |
| on instruction access | | IACCVIOL ^[a] | <i>MemManage Fault Address Register</i> |
| on data access | | DACCVIOL | |
| during exception stacking | | MSTKERR | |
| during exception unstacking | | MUNSKERR | |
| during lazy floating-point state preservation | | MLSPERR | |
| Bus error: | BusFault | - | - |
| during exception stacking | | STKERR | <i>BusFault Status Register</i> |
| during exception unstacking | | UNSTKERR | |
| during instruction prefetch | | IBUSERR | |
| during lazy floating-point state preservation | | LSPERR | |
| Precise data bus error | | PRECISERR | |
| Imprecise data bus error | | IMPRECISERR | |
| Attempt to access a coprocessor | UsageFault | NOCP | <i>UsageFault Status Register</i> |
| Undefined instruction | | UNDEFINSTR | |
| Attempt to enter an invalid instruction set state ^[b] | | INVSTATE | |
| Invalid EXC_RETURN value | | INVPC | |
| Illegal unaligned load or store | | UNALIGNED | |
| Divide By 0 | | DIVBYZERO | |

21 . Table 2-18 Faults

a. Occurs on an access to an XN region even if the MPU is disabled.

b. Attempting to use an instruction set other than the Thumb instruction set or returns to a non load/store-multiple instruction with ICI continuation.

Fault escalation and hard faults

All faults exceptions except for HardFault have configurable exception priority, see [System Handler Priority Registers](#). Software can disable execution of the handlers for these faults, see [System Handler Control and State Register](#).

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler. as described in [Exception model](#).

In some situations, a fault with configurable priority is treated as a HardFault. This is called *priority escalation*, and the fault is described as *escalated to HardFault*. Escalation to HardFault occurs when:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to HardFault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This is because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.
- A fault occurs and the handler for that fault is not enabled.

If a BusFault occurs during a stack push when entering a BusFault handler, the BusFault does not escalate to a HardFault. This means that if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

Note

Only Reset and NMI can preempt the fixed priority HardFault. A HardFault can preempt any exception other than Reset, NMI, or another HardFault.

Fault status registers and fault address registers

The fault status registers indicate the cause of a fault. For BusFaults and MemManage faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 2-19.

| Handler | Status register name | Address register name | Register description |
|------------|----------------------|-----------------------|---|
| HardFault | HFSR | - | HardFault Status Register |
| MemManage | MMFSR | MMFAR | MemManage Fault Status Register MemManage Fault Address Register |
| BusFault | BFSR | BFAR | BusFault Status Register BusFault Address Register |
| UsageFault | UFSR | - | UsageFault Status Register |

22 . Fault status and fault address registers

Lockup

The processor enters a lockup state if a fault occurs when executing the NMI or HardFault handlers. When the processor is in lockup state it does not execute any instructions. The processor remains in lockup state until either:

- it is reset
- an NMI occurs
- it is halted by a debugger

Note

If lockup state occurs from the NMI handler a subsequent NMI does not cause the processor to leave lockup state.

1.2.5 Power management

The Cortex-M4 processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock.
- Deep sleep mode stops the system clock and switches off the PLL and flash memory.

The SLEEPDEEP bit of the SCR selects which sleep mode is used, see [System Control Register](#). For more information about the behavior of the sleep modes see [Power Architecture](#).

This section describes the mechanisms for entering sleep mode, and the conditions for waking up from sleep mode.

Entering sleep mode

This section describes the mechanisms software can use to put the processor into sleep mode.

The system can generate spurious wakeup events, for example a debug operation wakes up the processor. Therefore software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

Wait for interrupt

The *wait for interrupt* instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true, see [Wakeup from WFI or sleep-on-exit](#). When the processor executes a WFI instruction it stops executing instructions and enters sleep mode. See [WFI](#) for more information.

Wait for event

External events are not supported.

Sleep-on-exit

If the SLEEPONEXIT bit of the SCR is set to 1, when the processor completes the execution of all exception handlers it returns to Thread mode and immediately enters sleep mode. Use this mechanism in applications that only require the processor to run when an exception occurs.

Wakeup from sleep mode

The conditions for the processor to wakeup depend on the mechanism that cause it to enter sleep mode.

Wakeup from WFI or sleep-on-exit

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up, and before it executes an interrupt handler. To achieve this set the PRIMASK bit to 1 and the FAULTMASK bit to 0. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor sets PRIMASK to zero. For more information about PRIMASK and FAULTMASK see [Exception mask registers](#).

Wakeup from WFE

The processor wakes up if:

- it detects an exception with sufficient priority to cause exception entry

In addition, if the SEVONPEND bit in the SCR is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about the SCR see [System Control Register](#).

Power management programming hints

ISO/IEC C cannot directly generate the WFI and WFE instructions. The CMSIS provides the following functions for these instructions:

```
void __WFE(void) // Wait for Event
void __WFI(void) // Wait for Interrupt
```

1.3 Cortex-M4 Instruction Set

Instruction set summary

The processor implements a version of the Thumb instruction set. Table 20 lists the supported instructions.



Note

In Table 19:

- angle brackets, <>, enclose alternative forms of the operand
- braces, {}, enclose optional operands
- the Operands column is not exhaustive
- Op2 is a flexible second operand that can be either a register or a constant
- most instructions can use an optional condition code suffix.

For more information on the instructions and operands, see the instruction descriptions.

| Mnemonic | Operands | Brief description | Flags |
|-----------|----------------------|--------------------------------|---------|
| ADC, ADCS | {Rd,} Rn, Op2 | Add with Carry | N,Z,C,V |
| ADD, ADDS | {Rd,} Rn, Op2 | Add | N,Z,C,V |
| ADD, ADDW | {Rd,} Rn, #imm12 | Add | - |
| ADR | Rd, label | Load PC-relative Address | - |
| AND, ANDS | {Rd,} Rn, Op2 | Logical AND | N,Z,C |
| ASR, ASRS | Rd, Rm, <Rs #n> | Arithmetic Shift Right | N,Z,C |
| B | label | Branch | - |
| BFC | Rd, #lsb, #width | Bit Field Clear | - |
| BFI | Rd, Rn, #lsb, #width | Bit Field Insert | - |
| BIC, BICS | {Rd,} Rn, Op2 | Bit Clear | N,Z,C |
| BKPT | #imm | Breakpoint | - |
| BL | label | Branch with Link | - |
| BLX | Rm | Branch indirect with Link | - |
| BX | Rm | Branch indirect | - |
| CBNZ | Rn, label | Compare and Branch if Non Zero | - |

| Mnemonic | Operands | Brief description | Flags |
|----------|-----------|--|---------|
| CBZ | Rn, label | Compare and Branch if Zero | - |
| CLREX | - | Clear Exclusive | - |
| CLZ | Rd, Rm | Count Leading Zeros | - |
| CMN | Rn, Op2 | Compare Negative | N,Z,C,V |
| CMP | Rn, Op2 | Compare | N,Z,C,V |
| CPSID | i | Change Processor State, Disable Interrupts | - |
| CPSIE | i | Change Processor State, Enable Interrupts | - |

| Mnemonic | Operands | Brief description | Flags |
|---------------|------------------------|---|-------|
| DMB | - | Data Memory Barrier | - |
| DSB | - | Data Synchronization Barrier | - |
| EOR, EORS | {Rd,} Rn, Op2 | Exclusive OR | N,Z,C |
| ISB | - | Instruction Synchronization Barrier | - |
| IT | - | If-Then condition block | - |
| LDM | Rn{!}, reglist | Load Multiple registers, increment after | - |
| LDMDB, LDMEA | Rn{!}, reglist | Load Multiple registers, decrement before | - |
| LDMFD, LDMIA | Rn{!}, reglist | Load Multiple registers, increment after | - |
| LDR | Rt, [Rn, #offset] | Load Register with word | - |
| LDRB, LDRBT | Rt, [Rn, #offset] | Load Register with byte | - |
| LDRD | Rt, Rt2, [Rn, #offset] | Load Register with two bytes | - |
| LDREX | Rt, [Rn, #offset] | Load Register Exclusive | - |
| LDREXB | Rt, [Rn] | Load Register Exclusive with Byte | - |
| LDREXH | Rt, [Rn] | Load Register Exclusive with Halfword | - |
| LDRH, LDRHT | Rt, [Rn, #offset] | Load Register with Halfword | - |
| LDRSB, LDRSBT | Rt, [Rn, #offset] | Load Register with Signed Byte | - |
| LDRSH, LDRSHT | Rt, [Rn, #offset] | Load Register with Signed Halfword | - |
| LDRT | Rt, [Rn, #offset] | Load Register with word | - |
| LSL, LSLS | Rd, Rm, <Rs #n> | Logical Shift Left | N,Z,C |
| LSR, LSRS | Rd, Rm, <Rs #n> | Logical Shift Right | N,Z,C |
| LSL, LSLS | Rd, Rm, <Rs #n> | Logical Shift Left | N,Z,C |
| LSR, LSRS | Rd, Rm, <Rs #n> | Logical Shift Right | N,Z,C |
| MLA | Rd, Rn, Rm, Ra | Multiply with Accumulate, 3bit result | - |

| Mnemonic | Operands | Brief description | Flags |
|--------------|-------------------|--|---------|
| MLS | Rd, Rn, Rm, Ra | Multiply and Subtract, 3bit result | - |
| MOV, MOVS | Rd, Op2 | Move | N,Z,C |
| MOVT | Rd, #imm16 | Move Top | - |
| MOVW, MOV | Rd, #imm16 | Move 16-bit constant | N,Z,C |
| MRS | Rd, spec_reg | Move from Special Register to general register | - |
| MSR | spec_reg, Rm | Move from general register to Special Register | N,Z,C,V |
| MUL, MULS | {Rd,} Rn, Rm | Multiply, 3bit result | N,Z |
| MVN, MVNS | Rd, Op2 | Move NOT | N,Z,C |
| NOP | - | No Operation | - |
| ORN, ORNS | {Rd,} Rn, Op2 | Logical OR NOT | N,Z,C |
| ORR, ORRS | {Rd,} Rn, Op2 | Logical OR | N,Z,C |
| PKHTB, PKHBT | {Rd,} Rn, Rm, Op2 | Pack Halfword | - |

| Mnemonic | Operands | Brief description | Flags |
|-----------|-------------------|--|-------|
| POP | reglist | Pop registers from stack | - |
| PUSH | reglist | Push registers onto stack | - |
| QADD | {Rd,} Rn, Rm | Saturating double and Add | Q |
| QADD16 | {Rd,} Rn, Rm | Saturating Add 16 | - |
| QADD8 | {Rd,} Rn, Rm | Saturating Add 8 | - |
| QASX | {Rd,} Rn, Rm | Saturating Add and Subtract with Exchange | - |
| QDADD | {Rd,} Rn, Rm | Saturating Add | Q |
| QDSUB | {Rd,} Rn, Rm | Saturating double and Subtract | Q |
| QSAX | {Rd,} Rn, Rm | Saturating Subtract and Add with Exchange | - |
| QSUB | {Rd,} Rn, Rm | Saturating Subtract | Q |
| QSUB16 | {Rd,} Rn, Rm | Saturating Subtract 16 | - |
| QSUB8 | {Rd,} Rn, Rm | Saturating Subtract 8 | - |
| RBIT | Rd, Rn | Reverse Bits | - |
| REV | Rd, Rn | Reverse byte order in a word | - |
| REV16 | Rd, Rn | Reverse byte order in each halfword | - |
| REVSH | Rd, Rn | Reverse byte order in bottom halfword and sign extend- | |
| ROR, RORS | Rd, Rm, <Rs #n> | Rotate Right | N,Z,C |
| RRX, RRXS | Rd, Rm | Rotate Right with Extend | N,Z,C |

| Mnemonic | Operands | Brief description | Flags |
|-----------|----------------------|---|---------|
| RSB, RSBS | {Rd,} Rn, Op2 | Reverse Subtract | N,Z,C,V |
| SADD16 | {Rd,} Rn, Rm | Signed Add 16 | GE |
| SADD8 | {Rd,} Rn, Rm | Signed Add 8 | GE |
| SASX | {Rd,} Rn, Rm | Signed Add and Subtract with Exchange | GE |
| SBC, SBCS | {Rd,} Rn, Op2 | Subtract with Carry | N,Z,C,V |
| SBFX | Rd, Rn, #lsb, #width | Signed Bit Field Extract | - |
| SDIV | {Rd,} Rn, Rm | Signed Divide | - |
| SEL | {Rd,} Rn, Rm | Select bytes | - |
| SEV | - | Send Event | - |
| SHADD16 | {Rd,} Rn, Rm | Signed Halving Add 16 | - |
| SHADD8 | {Rd,} Rn, Rm | Signed Halving Add 8 | - |
| SHASX | {Rd,} Rn, Rm | Signed Halving Add and Subtract with Exchange | - |
| SHSAX | {Rd,} Rn, Rm | Signed Halving Subtract and Add with Exchange | - |
| SHSUB16 | {Rd,} Rn, Rm | Signed Halving Subtract 16 | - |
| SHSUB8 | {Rd,} Rn, Rm | Signed Halving Subtract | - |

| Mnemonic | Operands | Brief description | Flags |
|---|--------------------|---|-------|
| SMLABB, SMLABT, SMLATB, SMLATT | Rd, Rn, Rm, Ra | Signed Multiply Accumulate Long (halfwords) | Q |
| SMLAD, SMLADX | Rd, Rn, Rm, Ra | Signed Multiply Accumulate Dual | Q |
| SMLAL | RdLo, RdHi, Rn, Rm | Signed Multiply with Accumulate (32 x 32 + 64), 64-bit result | - |
| SMLALBB, SMLALBT, SMLALTB, SMLALTT | RdLo, RdHi, Rn, Rm | Signed Multiply Accumulate Long, halfwords | - |
| SMLALD, SMLALDX | RdLo, RdHi, Rn, Rm | Signed Multiply Accumulate Long Dual | - |
| SMLAWB, SMLAWT | Rd, Rn, Rm, Ra | Signed Multiply Accumulate, word by halfword | Q |
| SMLSD | Rd, Rn, Rm, Ra | Signed Multiply Subtract Dual | Q |
| SMLS LD | RdLo, RdHi, Rn, Rm | Signed Multiply Subtract Long Dual | - |
| SMMLA | Rd, Rn, Rm, Ra | Signed Most significant word Multiply Accumulate | - |
| SMMLS, SMMLR | Rd, Rn, Rm, Ra | Signed Most significant word Multiply Subtract | - |
| SMMUL, SMMULR | {Rd,} Rn, Rm | Signed Most significant word Multiply | - |

| Mnemonic | Operands | Brief description | Flags |
|-------------------------------------|------------------------|--|-------|
| SMUAD | {Rd,} Rn, Rm | Signed dual Multiply Add | Q |
| SMULBB, SMULBT SMULTB, SMULLT | {Rd,} Rn, Rm | Signed Multiply (halfwords) | - |
| SMULL | RdLo, RdHi, Rn, Rm | Signed Multiply (32 x 32), 64-bit result | - |
| SMULWB, SMULWT | {Rd,} Rn, Rm | Signed Multiply word by halfword | - |
| SMUSD, SMUSDX | {Rd,} Rn, Rm | Signed dual Multiply Subtract | - |
| SSAT | Rd, #n, Rm {shift #s} | Signed Saturate | Q |
| SSAT16 | Rd, #n, Rm | Signed Saturate 16 | Q |
| SSAX | {Rd,} Rn, Rm | Signed Subtract and Add with Exchange | GE |
| SSUB16 | {Rd,} Rn, Rm | Signed Subtract 16 | - |
| SSUB8 | {Rd,} Rn, Rm | Signed Subtract 8 | - |
| STM | Rn{!}, reglist | Store Multiple registers, increment after | - |
| STMDB, STMEA | Rn{!}, reglist | Store Multiple registers, decrement before | - |
| STMFD, STMIA | Rn{!}, reglist | Store Multiple registers, increment after | - |
| STR | Rt, [Rn, #offset] | Store Register word | - |
| STRB, STRBT | Rt, [Rn, #offset] | Store Register byte | - |
| STRD | Rt, Rt2, [Rn, #offset] | Store Register two words | - |
| STREX | Rd, Rt, [Rn, #offset] | Store Register Exclusive | - |

| Mnemonic | Operands | Brief description | Flags |
|-------------|-----------------------|-----------------------------------|---------|
| STREXB | Rd, Rt, [Rn] | Store Register Exclusive Byte | - |
| STREXH | Rd, Rt, [Rn] | Store Register Exclusive Halfword | - |
| STRH, STRHT | Rt, [Rn, #offset] | Store Register Halfword | - |
| STRT | Rt, [Rn, #offset] | Store Register word | - |
| SUB, SUBS | {Rd,} Rn, Op2 | Subtract | N,Z,C,V |
| SUB, SUBW | {Rd,} Rn, #imm12 | Subtract | - |
| SVC | #imm | Supervisor Call | - |
| SXTAB | {Rd,} Rn, Rm,{,ROR #} | Extend 8 bits to 32 and add | - |
| SXTAB16 | {Rd,} Rn, Rm,{,ROR #} | Dual extend 8 bits to 16 and add | - |
| SXTAH | {Rd,} Rn, Rm,{,ROR #} | Extend 16 bits to 32 and add | - |
| SXTB16 | {Rd,} Rm {,ROR #n} | Signed Extend Byte 16 | - |
| SXTB | {Rd,} Rm {,ROR #n} | Sign extend a byte | - |

| Mnemonic | Operands | Brief description | Flags |
|----------|----------------------|---|-------|
| SXTH | {Rd,} Rm {,ROR #n} | Sign extend a halfword | - |
| TBB | [Rn, Rm] | Table Branch Byte | - |
| TBH | [Rn, Rm, LSL #1] | Table Branch Halfword | - |
| TEQ | Rn, Op2 | Test Equivalence | N,Z,C |
| TST | Rn, Op2 | Test | N,Z,C |
| UADD16 | {Rd,} Rn, Rm | Unsigned Add 16 | GE |
| UADD8 | {Rd,} Rn, Rm | Unsigned Add 8 | GE |
| USAX | {Rd,} Rn, Rm | Unsigned Subtract and Add with Exchange | GE |
| UHADD16 | {Rd,} Rn, Rm | Unsigned Halving Add 16 | - |
| UHADD8 | {Rd,} Rn, Rm | Unsigned Halving Add 8 | - |
| UHASX | {Rd,} Rn, Rm | Unsigned Halving Add and Subtract with Exchange | - |
| UHSAX | {Rd,} Rn, Rm | Unsigned Halving Subtract and Add with Exchange | - |
| UHSUB16 | {Rd,} Rn, Rm | Unsigned Halving Subtract 16 | - |
| UHSUB8 | {Rd,} Rn, Rm | Unsigned Halving Subtract 8 | - |
| UBFX | Rd, Rn, #lsb, #width | Unsigned Bit Field Extract | - |
| UDIV | {Rd,} Rn, Rm | Unsigned Divide | - |
| UMAAL | RdLo, RdHi, Rn, Rm | Unsigned Multiply Accumulate Accumulate Long (32 x 32 + 32 + 32), 64-bit result | - |
| UMLAL | RdLo, RdHi, Rn, Rm | Unsigned Multiply with Accumulate (32 x 32 + 64), 64-bit result | - |
| UMULL | RdLo, RdHi, Rn, Rm | Unsigned Multiply (32 x 32), 64-bit result | - |

| Mnemonic | Operands | Brief description | Flags |
|----------|------------------------|---|-------|
| UQADD16 | {Rd,} Rn, Rm | Unsigned Saturating Add 16 | - |
| UQADD8 | {Rd,} Rn, Rm | Unsigned Saturating Add 8 | - |
| UQASX | {Rd,} Rn, Rm | Unsigned Saturating Add and Subtract with Exchange | - |
| UQSAX | {Rd,} Rn, Rm | Unsigned Saturating Subtract and Add with Exchange | - |
| UQSUB16 | {Rd,} Rn, Rm | Unsigned Saturating Subtract 16 | - |
| UQSUB8 | {Rd,} Rn, Rm | Unsigned Saturating Subtract 8 | - |
| USAD8 | {Rd,} Rn, Rm | Unsigned Sum of Absolute Differences | - |
| USADA8 | {Rd,} Rn, Rm, Ra | Unsigned Sum of Absolute Differences and Accumulate | - |
| USAT | Rd, #n, Rm {,shift #s} | Unsigned Saturate | Q |

| Mnemonic | Operands | Brief description | Flags |
|-------------------------|---------------------------|--|-------|
| USAT16 | Rd, #n, Rm | Unsigned Saturate 16 | Q |
| UASX | {Rd,} Rn, Rm | Unsigned Add and Subtract with Exchange | GE |
| USUB16 | {Rd,} Rn, Rm | Unsigned Subtract 16 | GE |
| USUB8 | {Rd,} Rn, Rm | Unsigned Subtract 8 | GE |
| UXTAB | {Rd,} Rn, Rm, ,{ROR #} | Rotate, extend 8 bits to 32 and Add | - |
| UXTAB16 | {Rd,} Rn, Rm, ,{ROR #} | Rotate, dual extend 8 bits to 16 and Add | - |
| UXTAH | {Rd,} Rn, Rm, ,{ROR #} | Rotate, unsigned extend and Add Halfword | - |
| UXTB | {Rd,} Rm ,{ROR #n} | Zero extend a Byte | - |
| UXTB16 | {Rd,} Rm ,{ROR #n} | Unsigned Extend Byte 16 | - |
| UXTH | {Rd,} Rm ,{ROR #n} | Zero extend a Halfword | - |
| VABS.F32 | Sd, Sm | Floating-point Absolute | - |
| VADD.F32 | {Sd,} Sn, Sm | Floating-point Add | - |
| VCMP.F32 | Sd, <Sm #0.0> | | |
| VCMP.E.F32 | | | |
| VCMP.F32 | Sd, <Sm #0.0> | Compare two floating-point registers, or one floating-point register and zero | FPSCR |
| VCMPE.F32 | Sd, <Sm #0.0> | Compare two floating-point registers, or one floating-point register and zero with Invalid Operation check | FPSCR |
| VCVT.S32.F32 | Sd, Sm | Convert between floating-point and integer | - |
| VCVT.S16.F32 | Sd, Sd, #fbits | Convert between floating-point and fixed point | - |
| VCVTR.S32.F32 | Sd, Sm | Convert between floating-point and integer with rounding | - |
| VCVT<B H>.F32.F16 | Sd, Sm | Converts half-precision value to single-precision | - |
| VCVTT<B T>.F32.F16 | Sd, Sm | Converts single-precision register to half-precision | - |
| VDIV.F32 | {Sd,} Sn, Sm | Floating-point Divide | - |
| VFMA.F32 | {Sd,} Sn, Sm | Floating-point Fused Multiply Accumulate | - |

| Mnemonic | Operands | Brief description | Flags |
|-----------------|------------------|---|---------|
| VFNMA.F32 | {Sd,} Sn, Sm | Floating-point Fused Negate Multiply Accumulate | - |
| VFMS.F32 | {Sd,} Sn, Sm | Floating-point Fused Multiply Subtract | - |
| VFNMS.F32 | {Sd,} Sn, Sm | Floating-point Fused Negate Multiply Subtract | - |
| VLDM.F<32 64> | Rn{!}, list | Load Multiple extension registers | - |
| VLDR.F<32 64> | <Dd Sd>, [Rn] | Load an extension register from memory | - |
| VLMA.F32 | {Sd,} Sn, Sm | Floating-point Multiply Accumulate | - |
| VLMS.F32 | {Sd,} Sn, Sm | Floating-point Multiply Subtract | - |
| VMOV.F32 | Sd, #imm | Floating-point Move immediate | - |
| VMOV | Sd, Sm | Floating-point Move register | - |
| VMOV | Sn, Rt | Copy ARM core register to single precision | - |
| VMOV | Sm, Sm1, Rt, Rt2 | Copy 2 ARM core registers to 2 single precision | - |
| VMOV | Dd[x], Rt | Copy ARM core register to scalar | - |
| VMOV | Rt, Dn[x] | Copy scalar to ARM core register | - |
| VMRS | Rt, FPSCR | Move FPSCR to ARM core register or APSR | N,Z,C,V |
| VMSR | FPSCR, Rt | Move to FPSCR from ARM Core register | FPSCR |
| VMUL.F32 | {Sd,} Sn, Sm | Floating-point Multiply | - |
| VNEG.F32 | Sd, Sm | Floating-point Negate | - |
| VNMLA.F32 | Sd, Sn, Sm | Floating-point Multiply and Add | - |
| VNMLS.F32 | Sd, Sn, Sm | Floating-point Multiply and Subtract | - |
| VNMUL | {Sd,} Sn, Sm | Floating-point Multiply | - |
| VPOP | list | Pop extension registers | - |
| VPUSH | list | Push extension registers | - |
| VSQRT.F32 | Sd, Sm | Calculates floating-point Square Root | - |
| VSTM | Rn{!}, list | Floating-point register Store Multiple | - |
| VSTR.F<32 64> | Sd, [Rn] | Stores an extension register to memory | - |
| VSUB.F<32 64> | {Sd,} Sn, Sm | Floating-point Subtract | - |
| WFE | - | Wait For Event | - |
| WFI | - | Wait For Interrupt | - |

23 . Cortex-M4 instructions

1.3.1 CMSIS functions

ISO/IEC C code cannot directly access some Cortex-M4 instructions. This section describes intrinsic functions that can generate these instructions, provided by the CMSIS and that might be provided by a C compiler. If a C compiler does not support an appropriate intrinsic function, you might have to use inline assembler to access some

instructions. The CMSIS provides the following intrinsic functions to generate instructions that ISO/IEC C code cannot directly access:

| Instruction | CMSIS function |
|-------------|--------------------------------------|
| CPSIE I | void __enable_irq(void) |
| CPSID I | void __disable_irq(void) |
| CPSIE F | void __enable_fault_irq(void) |
| CPSID F | void __disable_fault_irq(void) |
| ISB | void __ISB(void) |
| DSB | void __DSB(void) |
| DMB | void __DMB(void) |
| REV | uint32_t __REV(uint32_t int value) |
| REV16 | uint32_t __REV16(uint32_t int value) |
| REVSH | uint32_t __REVSH(uint32_t int value) |
| RBIT | uint32_t __RBIT(uint32_t int value) |
| SEV | void __SEV(void) |
| WFE | void __WFE(void) |
| WFI | void __WFI(void) |

The CMSIS also provides a number of functions for accessing the special registers using MRS and MSR instructions:

| Special register | Access | CMSIS function |
|------------------|--------|---------------------------------------|
| PRIMASK | Read | uint32_t __get_PRIMASK (void) |
| | Write | void __set_PRIMASK (uint32_t value) |
| FAULTMASK | Read | uint32_t __get_FAULTMASK (void) |
| | Write | void __set_FAULTMASK (uint32_t value) |
| BASEPRI | Read | uint32_t __get_BASEPRI (void) |
| | Write | void __set_BASEPRI (uint32_t value) |
| CONTROL | Read | uint32_t __get_CONTROL (void) |

Write void __set_CONTROL (uint32_t value)

| Special register | Access | CMSIS function |
|------------------|--------|---------------------------|
| MSP | Read | uint32_t __get_MSP (void) |

| Special register | Access | CMSIS function |
|------------------|--------|--|
| | Write | void __set_MSP (uint32_t TopOfMainStack) |
| PSP | Read | uint32_t __get_PSP (void) |
| | Write | void __set_PSP (uint32_t TopOfProcStack) |

1.3.2 About the instruction descriptions

The following sections give more information about using the instructions:

- [Operands](#)
- [Restrictions when using PC or SP](#)
- [Flexible second operand](#)
- [Shift Operations](#)
- [Address alignment](#)
- [PC-relative expressions](#)
- [Conditional execution](#)
- [Instruction width selection](#)

Operands

An instruction operand can be an ARM register, a constant, or another instruction-specific parameter. Instructions act on the operands and often store the result in a destination register. When there is a destination register in the instruction, it is usually specified before the operands.

Operands in some instructions are flexible in that they can either be a register or a constant. See [Flexible second operand](#).

Restrictions when using PC or SP

Many instructions have restrictions on whether you can use the *Program Counter* (PC) or *Stack Pointer* (SP) for the operands or destination register. See instruction descriptions for more information.

Note: Bit[0] of any address you write to the PC with a BX, BLX, LDM, LDR, or POP instruction must be 1 for correct execution, because this bit indicates the required instruction set, and the Cortex-M4 processor only supports Thumb instructions.

Flexible second operand

Many general data processing instructions have a flexible second operand. This is shown as *Operand2* in the descriptions of the syntax of each instruction.

Operand2 can be a:

- [Constant](#)
- [Register with optional shift](#)

Constant

You specify an *Operand2* constant in the form:

#*constant* where *constant* can be:

- any constant that can be produced by shifting an 8-bit value left by any number of bits within a 3bit word
- any constant of the form 0x00XY00XY
- any constant of the form 0xXY00XY00
- any constant of the form 0xXYXYXYXY.

Note: In the constants shown above, X and Y are hexadecimal digits.

In addition, in a small number of instructions, *constant* can take a wider range of values. These are described in the individual instruction descriptions.

When an Operand2 constant is used with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to bit[31] of the constant, if the constant is greater than 255 and can be produced by shifting an 8-bit value. These instructions do not affect the carry flag if Operand2 is any other constant.

Instruction substitution

Your assembler might be able to produce an equivalent instruction in cases where you specify a constant that is not permitted. For example, an assembler might assemble the instruction CMP Rd, #0xFFFFFFF as the equivalent instruction CMN Rd, #0x2.

Register with optional shift

You specify an Operand2 register in the form: *Rm*{, *shift*} where:

Rm is the register holding the data for the second operand. *shift* is an optional shift to be applied to *Rm*. It can be one of:

| | |
|--------|--|
| ASR #n | arithmetic shift right <i>n</i> bits, $1 \leq n \leq 32$. |
| LSL #n | logical shift left <i>n</i> bits, $1 \leq n \leq 31$. |
| LSR #n | logical shift right <i>n</i> bits, $1 \leq n \leq 32$. |
| ROR #n | rotate right <i>n</i> bits, $1 \leq n \leq 31$. |
| RRX | rotate right one bit, with extend. |
| - | if omitted, no shift occurs, equivalent to LSL #0. |

If you omit the shift, or specify LSL #0, the instruction uses the value in *Rm*.

If you specify a shift, the shift is applied to the value in *Rm*, and the resulting 3bit value is used by the instruction. However, the contents in the register *Rm* remains unchanged. Specifying a register with shift also updates the carry flag when used with certain instructions. For information on the shift operations and how they affect the carry flag, see [Shift Operations](#).

Shift Operations

Register shift operations move the bits in a register left or right by a specified number of bits, the *shift length*. Register shift can be performed:

- directly by the instructions ASR, LSR, LSL, ROR, and RRX, and the result is written to a destination register
- during the calculation of *Operand2* by the instructions that specify the second operand as a register with shift, see [Flexible second operand](#). The result is used by the instruction.

The permitted shift lengths depend on the shift type and the instruction, see the individual instruction description or [Flexible second operand](#). If the shift length is 0, no shift occurs. Register shift operations update the carry flag except when the specified shift length is 0. The following sub-sections describe the various shift operations and how they affect the carry flag. In these descriptions, *Rm* is the register containing the value to be shifted, and *n* is the shift length.

ASR

Arithmetic shift right by *n* bits moves the left-hand $3n$ bits of the register *Rm*, to the right by *n* places, into the right-hand $3n$ bits of the result. And it copies the original bit[31] of the register into the left-hand *n* bits of the result. See Figure 6.

You can use the ASR #*n* operation to divide the value in the register *Rm* by 2^n , with the result being rounded towards negative-infinity.

When the instruction is ASRS or when ASR #*n* is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit shifted out, bit[*n*-1], of the register *Rm*.

Note:

- If n is 32 or more, then all the bits in the result are set to the value of bit[31] of Rm .
- If n is 32 or more and the carry flag is updated, it is updated to the value of bit[31] of Rm .

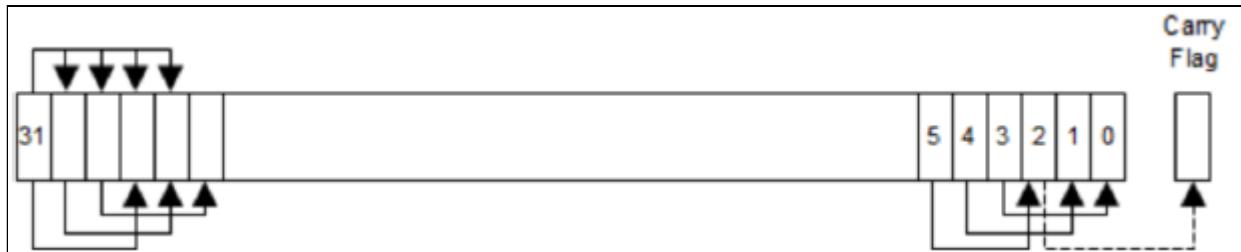


Figure 6: ASR #3

LSR

Logical shift right by n bits moves the left-hand $3n$ bits of the register Rm , to the right by n places, into the right-hand $3n$ bits of the result. And it sets the left-hand n bits of the result to 0. See Figure 7.

You can use the LSR # n operation to divide the value in the register Rm by 2^n , if the value is regarded as an unsigned integer.

When the instruction is LSRS or when LSR # n is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit shifted out, bit[$n-1$], of the register Rm .

Note

- If n is 32 or more, then all the bits in the result are cleared to 0.
- If n is 33 or more and the carry flag is updated, it is updated to 0.

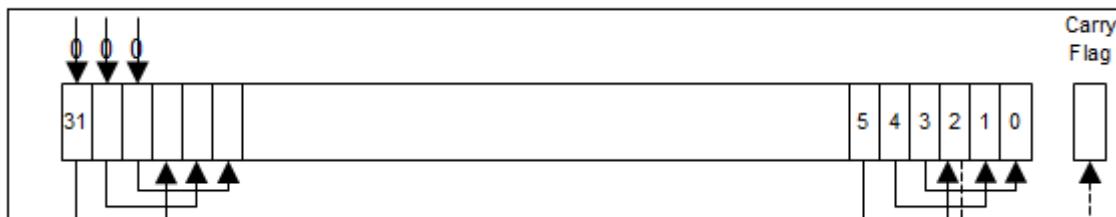


Figure 7: LSR #3

LSL

Logical shift left by n bits moves the right-hand $3n$ bits of the register Rm , to the left by n places, into the left-hand $3n$ bits of the result. And it sets the right-hand n bits of the result to 0. See Figure 8.

You can use the LSL # n operation to multiply the value in the register Rm by 2^n , if the value is regarded as an unsigned integer or a two's complement signed integer. Overflow can occur without warning.

When the instruction is LSLS or when LSL # n , with non-zero n , is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit shifted out, bit[$3n$], of the register Rm . These instructions do not affect the carry flag when used with LSL #0.

Note

- If n is 32 or more, then all the bits in the result are cleared to 0.

- If n is 33 or more and the carry flag is updated, it is updated to 0.

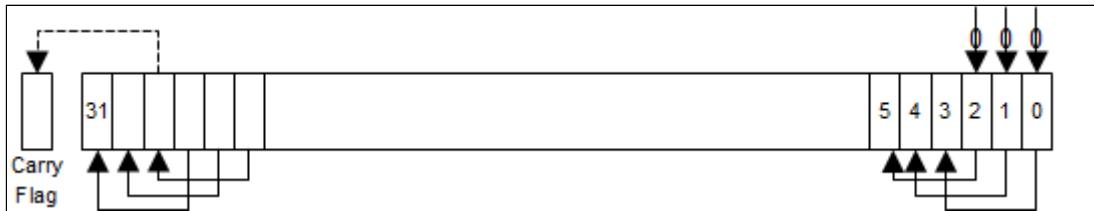


Figure 8: LSL #3

ROR

Rotate right by n bits moves the left-hand $3n$ bits of the register Rm , to the right by n places, into the right-hand $3n$ bits of the result. And it moves the right-hand n bits of the register into the left-hand n bits of the result. See Figure 9.

When the instruction is RORS or when ROR # n is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to the last bit rotation, bit[$n-1$], of the register Rm .

Note

- If n is 32, then the value of the result is same as the value in Rm , and if the carry flag is updated, it is updated to bit[31] of Rm .
- ROR with shift length, n , more than 32 is the same as ROR with shift length $n-32$.

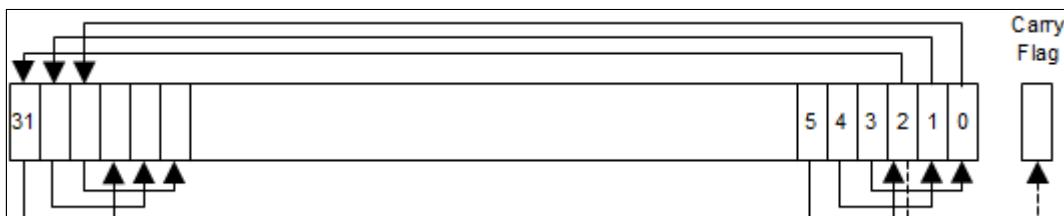


Figure 9: ROR #3

RRX

Rotate right with extend moves the bits of the register Rm to the right by one bit. And it copies the carry flag into bit[31] of the result. See Figure 10.

When the instruction is RRXS or when RRX is used in *Operand2* with the instructions MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ or TST, the carry flag is updated to bit[0] of the register Rm .

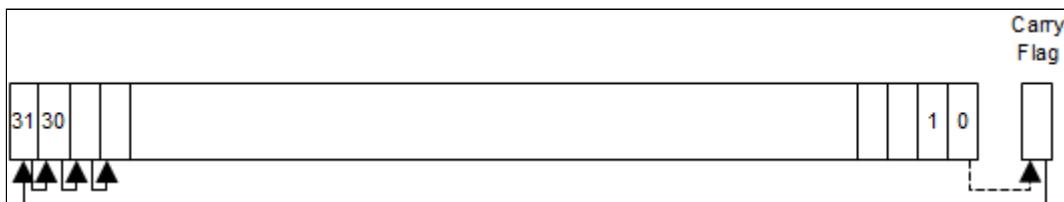


Figure 10: RRX

Address alignment

An aligned access is an operation where a word-aligned address is used for a word, dual word, or multiple word access, or where a halfword-aligned address is used for a halfword access. Byte accesses are always aligned. The Cortex-M4 processor supports unaligned access only for the following instructions:

- LDR, LDRT
- LDRH, LDRHT
- LDRSH, LDRSHT
- STR, STRT
- STRH, STRHT

All other load and store instructions generate a UsageFault exception if they perform an unaligned access, and therefore their accesses must be address aligned. For more information about UsageFaults see, *Fault handling*.

Unaligned accesses are usually slower than aligned accesses. In addition, some memory regions might not support unaligned accesses. Therefore, ARM recommends that programmers ensure that accesses are aligned. To trap accidental generation of unaligned accesses, use the UNALIGN_TRP bit in the Configuration and Control Register, see [Configuration and Control Register](#).

PC-relative expressions

A PC-relative expression or *label* is a symbol that represents the address of an instruction or literal data. It is represented in the instruction as the PC value plus or minus a numeric offset. The assembler calculates the required offset from the label and the address of the current instruction. If the offset is too big, the assembler produces an error.

Note

- For B, BL, CBNZ, and CBZ instructions, the value of the PC is the address of the current instruction plus 4 bytes.
- For all other instructions that use labels, the value of the PC is the address of the current instruction plus 4 bytes, with bit[1] of the result cleared to 0 to make it word-aligned.
- Your assembler might permit other syntaxes for PC-relative expressions, such as a label plus or minus a number, or an expression of the form [PC, #number].

Conditional execution

Most data processing instructions can optionally update the condition flags in the *Application Program Status Register* (APSR) according to the result of the operation, see [Application Program Status Register](#). Some instructions update all flags, and some only update a subset. If a flag is not updated, the original value is preserved. See the instruction descriptions for the flags they affect.

You can execute an instruction conditionally, based on the condition flags set in another instruction, either:

- immediately after the instruction that updated the flags
- after any number of intervening instructions that have not updated the flags.

Conditional execution is available by using conditional branches or by adding condition code suffixes to instructions. See [Table 24](#) for a list of the suffixes to add to instructions to make them conditional instructions. The condition code suffix enables the processor to test a condition based on the flags. If the condition test of a conditional instruction fails, the instruction:

- does not execute
- does not write any value to its destination register
- does not affect any of the flags
- does not generate any exception.

Conditional instructions, except for conditional branches, must be inside an If-Then instruction block. See [IT](#) for more information and restrictions when using the IT instruction. Depending on the vendor, the assembler might automatically insert an IT instruction if you have conditional instructions outside the IT block. Use the CBZ and CBNZ instructions to compare the value of a register against zero and branch on the result. This section describes:

- [The condition flags](#)
- [Condition code suffixes](#)

The condition flags

The APSR contains the following condition flags:

- N** Set to 1 when the result of the operation was negative, cleared to 0 otherwise.
- Z** Set to 1 when the result of the operation was zero, cleared to 0 otherwise.
- C** Set to 1 when the operation resulted in a carry, cleared to 0 otherwise.
- V** Set to 1 when the operation caused overflow, cleared to 0 otherwise.

For more information about the APSR see [Program Status Register](#).

A carry occurs:

- if the result of an addition is greater than or equal to 2^{32}
- if the result of a subtraction is positive or zero
- as the result of an inline barrel shifter operation in a move or logical instruction.

Overflow occurs when the sign of the result, in bit[31], does not match the sign of the result had the operation been performed at infinite precision, for example:

- if adding two negative values results in a positive value
- if adding two positive values results in a negative value
- if subtracting a positive value from a negative value generates a positive value
- if subtracting a negative value from a positive value generates a negative value.

The Compare operations are identical to subtracting, for CMP, or adding, for CMN, except that the result is discarded. See the instruction descriptions for more information.



Note

Most instructions update the status flags only if the S suffix is specified. See the instruction descriptions for more information.

Condition code suffixes

The instructions that can be conditional have an optional condition code, shown in syntax descriptions as {cond}. Conditional execution requires a preceding IT instruction. An instruction with a condition code is only executed if the condition code flags in the APSR meet the specified condition. Table 24 shows the condition codes to use. You can use conditional execution with the IT instruction to reduce the number of branch instructions in code. Table 24 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

| Suffix | Flags | Meaning |
|----------|-------|--------------------------|
| EQ | Z = 1 | Equal |
| NE | Z = 0 | Not equal |
| CS or HS | C = 1 | Higher or same, unsigned |
| CC or LO | C = 0 | Lower, unsigned |
| MI | N = 1 | Negative |

| Suffix | Flags | Meaning |
|--------|-------|------------------|
| PL | N = 0 | Positive or zero |

| Suffix | Flags | Meaning |
|----------|-------|--------------------------|
| EQ | Z = 1 | Equal |
| NE | Z = 0 | Not equal |
| CS or HS | C = 1 | Higher or same, unsigned |
| CC or LO | C = 0 | Lower, unsigned |
| MI | N = 1 | Negative |
| PL | N = 0 | Positive or zero |

24 . Condition Code Suffixes

| Suffix | Flags | Meaning |
|--------|--------------------|--|
| VS | V = 1 | Overflow |
| VC | V = 0 | No overflow |
| HI | C = 1 and Z = 0 | Higher, unsigned |
| LS | C = 0 or Z = 1 | Lower or same, unsigned |
| GE | N = V | Greater than or equal, signed |
| LT | N != V | Less than, signed |
| GT | Z = 0 and N = V | Greater than, signed |
| LE | Z = 1 and N != V | Less than or equal, signed |
| AL | Can have any value | Always. This is the default when no suffix is specified. |

25 . Condition Code Suffixes (continued)

Table 25 shows the use of a conditional instruction to find the absolute value of a number. R0 = abs(R1).

| | | |
|-------|------------|--|
| MOVS | R0, R1 | ; R0 = R1, setting flags |
| IT | MI | ; skipping next instruction if value 0 or positive |
| RSBMI | R0, R0, #0 | ; If negative, R0 = -R0 |

26 . Absolute Value

Table 26 shows the use of conditional instructions to update the value of R4 if the signed values R0 is greater than R1 and R2 is greater than R3.

| | | |
|-----|--------|--|
| CMP | R0, R1 | ; Compare R0 and R1, setting flags |
| ITT | GT | ; Skip next two instructions unless GT condition holds |

| | | |
|-------|--------|---|
| CMPGT | R2, R3 | ; If 'greater than', compare R2 and R3, setting flags |
| MOVGT | R4, R5 | ; If still 'greater than', do R4 = R5 |

27 . Compare and update values

Instruction width selection

There are many instructions that can generate either a 16-bit encoding or a 3bit encoding depending on the operands and destination register specified. For some of these instructions, you can force a specific instruction size by using an instruction width suffix. The .W suffix forces a 3bit instruction encoding. The .N suffix forces a 16-bit instruction encoding.

If you specify an instruction width suffix and the assembler cannot generate an instruction encoding of the requested width, it generates an error.

Note: In some cases it might be necessary to specify the .W suffix, for example if the operand is the label of an instruction or literal data, as in the case of branch instructions. This is because the assembler might not automatically generate the right size encoding.

To use an instruction width suffix, place it immediately after the instruction mnemonic and condition code, if any. Table 27 shows instructions with the instruction width suffix.

| | | |
|---------|------------|---|
| BCS.W | label | ; creates a 3bit instruction even for a short branch |
| ADD.S.W | R0, R0, R1 | ; creates a 3bit instruction even though the same operation can be done by a 16-bit instruction |

28 . Instruction Width Selection

1.3.3 Memory access instructions

Table 28 shows the memory access instructions:

| Mnemonic | Brief description | See |
|-------------|---|---|
| ADR | Generate PC-relative address | ADR |
| CLREX | Clear Exclusive | CLREX |
| LDM{mode} | Load Multiple registers | LDM and STM |
| LDR{type} | Load Register using immediate offset | LDR and STR, immediate offset |
| LDR{type} | Load Register using register offset | LDR and STR, register offset |
| LDR{type}T | Load Register with unprivileged access | LDR and STR, unprivileged |
| LDR | Load Register using PC-relative address | LDR, PC-relative |
| LDRD | Load Register Dual | LDR and STR, immediate offset |
| LDREX{type} | Load Register Exclusive | LDREX and STREX |
| POP | Pop registers from stack | PUSH and POP |
| PUSH | Push registers onto stack | PUSH and POP |
| STM{mode} | Store Multiple registers | LDM and STM |
| STR{type} | Store Register using immediate offset | LDR and STR, immediate offset |
| STR{type} | Store Register using register offset | LDR and STR, register offset |

| Mnemonic | Brief description | See |
|-------------|---|---|
| STR{type}T | Store Register with unprivileged access | LDR and STR, unprivileged |
| STREX{type} | Store Register Exclusive | LDREX and STREX |

29 . Memory access instructions

ADR

Generate PC-relative address.

Syntax

`ADR{cond} Rd, /label` where: *cond* is an optional condition code, see [Conditional execution](#).
Rd is the destination register.
label is a PC-relative expression. See [PC-relative expressions](#).

Operation

ADR generates an address by adding an immediate value to the PC, and writes the result to the destination register. ADR provides the means by which position-independent code can be generated, because the address is PC-relative. If you use ADR to generate a target address for a BX or BLX instruction, you must ensure that bit[0] of the address you generate is set to 1 for correct execution.
Values of *label* must be within the range of -4095 to +4095 from the address in the PC.

Note: You might have to use the .W suffix to get the maximum offset range or to generate addresses that are not word-aligned. See [Instruction width selection](#).

Restrictions

Rd must not be SP and must not be PC.

Condition flags

This instruction does not change the flags.

Examples

| |
|---|
| <code>ADR R1, TextMessage ; Write address value of a location labelled as ; TextMessage to R1.</code> |
|---|

LDR and STR, immediate offset

Load and Store with immediate offset, pre-indexed immediate offset, or post-indexed immediate offset.

Syntax

| |
|--|
| <code>op{type}{cond} Rt, [Rn {, #offset}]</code> ; immediate offset |
| <code>op{type}{cond} Rt, [Rn, #offset]!</code> ; pre-indexed |
| <code>op{type}{cond} Rt, [Rn], #offset</code> ; post-indexed |
| <code>opD{cond} Rt, Rt2, [Rn {, #offset}]</code> ; immediate offset, two words |
| <code>opD{cond} Rt, Rt2, [Rn, #offset]!</code> ; pre-indexed, two words |
| <code>opD{cond} Rt, Rt2, [Rn], #offset</code> ; post-indexed, two words |

where:

op is one of:

LDR Load Register.

STR Store Register.

type is one of:

B Unsigned byte, zero extend to 32 bits on loads.

SB Signed byte, sign extend to 32 bits (LDR only).

H unsigned halfword, zero extend to 32 bits on loads.
SH signed halfword, sign extend to 32 bits (LDR only).

- omit, for word.

cond is an optional condition code, see [Conditional execution](#).

Rt is the register to load or store.

Rn is the register on which the memory address is based.

offset is an offset from *Rn*. If *offset* is omitted, the address is the contents of *Rn*.

Rt2 is the additional register to load or store for two-word operations.

Operation

LDR instructions load one or two registers with a value from memory.

STR instructions store one or two register values to memory.

Load and store instructions with immediate offset can use the following addressing modes:

Offset addressing

The offset value is added to or subtracted from the address obtained from the register *Rn*. The result is used as the address for the memory access. The register *Rn* is unaltered. The assembly language syntax for this mode is:

[*Rn*, #*offset*]

Pre-indexed addressing

The offset value is added to or subtracted from the address obtained from the register *Rn*. The result is used as the address for the memory access and written back into the register *Rn*. The assembly language syntax for this mode is:

[*Rn*, #*offset*]!

Post-indexed addressing

The address obtained from the register *Rn* is used as the address for the memory access. The offset value is added to or subtracted from the address, and written back into the register *Rn*. The assembly language syntax for this mode is:

[*Rn*], #*offset*

The value to load or store can be a byte, halfword, word, or two words. Bytes and halfwords can either be signed or unsigned. See [Address alignment](#).

Table 29 shows the ranges of offset for immediate, pre-indexed and post-indexed forms.

| Instruction type | Immediate offset | Pre-indexed | Post-indexed |
|---|--|--|--|
| Word, halfword, signed halfword, byte, or signed byte | -255 to 4095 | -255 to 255 | -255 to 255 |
| Two words | multiple of 4 in the range -1020 to 1020 | multiple of 4 in the range -1020 to 1020 | multiple of 4 in the range -1020 to 1020 |

30 . Offset ranges

Restrictions

For load instructions:

- *Rt* can be SP or PC for word loads only
- *Rt* must be different from *Rt2* for two-word loads
- *Rn* must be different from *Rt* and *Rt2* in the pre-indexed or post-indexed forms.

When *Rt* is PC in a word load instruction:

- bit[0] of the loaded value must be 1 for correct execution
- a branch occurs to the address created by changing bit[0] of the loaded value to 0
- if the instruction is conditional, it must be the last instruction in the IT block.

For store instructions:

- Rt can be SP for word stores only
- Rt must not be PC
- Rn must not be PC
- Rn must be different from Rt and $Rt2$ in the pre-indexed or post-indexed forms.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------------|-----------------------|--|
| LDR | R8, [R10] | ; Loads R8 from the address in R10. |
| LDRNE | R2, [R5, #960]! | ; Loads (conditionally) R2 from a word ; 960 bytes above the address in R5, and ; increments R5 by 960 |
| STR | R2, [R9,#const-struc] | ; const-struc is an expression evaluating ; to a constant in the range 0-4095. |
| STRH | R3, [R4], #4 | ; Store R3 as halfword data into address in ; R4, then increment R4 by 4 |
| LDRD | R8, R9, [R3, #0x20] | ; Load R8 from a word 32 bytes above the ; address in R3, and load R9 from a word 36 ; bytes above the address in R3 |
| STRD | R0, R1, [R8], #-16 | ; Store R0 to address in R8, and store R1 to ; a word 4 bytes above the address in R8, ; and then decrement R8 by 16. |

LDR and STR, register offset

Load and Store with register offset.

Syntax

$op\{type\}\{cond\} Rt, [Rn, Rm\{, LSL \#n\}]$ where:

op is one of:

LDR Load Register.

STR Store Register.

$type$ is one of:

B unsigned byte, zero extend to 32 bits on loads.

SB signed byte, sign extend to 32 bits (LDR only).

H unsigned halfword, zero extend to 32 bits on loads.

SH signed halfword, sign extend to 32 bits (LDR only).

- omit, for word.

$cond$ is an optional condition code, see [Conditional execution](#).

Rt is the register to load or store.

Rn is the register on which the memory address is based.

Rm is a register containing a value to be used as the offset.

LSL $\#n$ is an optional shift, with n in the range 0 to 3.

Operation

LDR instructions load a register with a value from memory.

STR instructions store a register value into memory.

The memory address to load from or store to is at an offset from the register Rn . The offset is specified by the register Rm and can be shifted left by up to 3 bits using LSL.

The value to load or store can be a byte, halfword, or word. For load instructions, bytes and halfwords can either be signed or unsigned. See [Address alignment](#).

Restrictions

In these instructions:

- Rn must not be PC
- Rm must not be SP and must not be PC
- Rt can be SP only for word loads and word stores
- Rt can be PC only for word loads.

When Rt is PC in a word load instruction:

- bit[0] of the loaded value must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------------|----------------------|--|
| STR | R0, [R5, R1] | ; Store value of R0 into an address equal to ; sum of R5 and R1 |
| LDRSB | R0, [R5, R1, LSL #1] | ; Read byte value from an address equal to ; sum of R5 and two times R1, sign extended it ; to a word value and put it in R0 |
| STR | R0, [R1, R2, LSL #2] | ; Stores R0 to an address equal to sum of R1 ; and four times R2. |

LDR and STR, unprivileged

Load and Store with unprivileged access.

Syntax $op\{type\}T\{cond\} Rt, [Rn\{, \#offset\}]; immediate offset where:$
 op is one of:

LDR Load Register.

STR Store Register.

$type$ is one of:

B unsigned byte, zero extend to 32 bits on loads.

SB signed byte, sign extend to 32 bits (LDR only).

H unsigned halfword, zero extend to 32 bits on loads.

SH signed halfword, sign extend to 32 bits (LDR only).

- omit, for word.

$cond$ is an optional condition code, see [Conditional execution](#).

Rt is the register to load or store.

Rn is the register on which the memory address is based.

$offset$ is an offset from Rn and can be 0 to 255. If $offset$ is omitted, the address is the value in Rn .

Operation

These load and store instructions perform the same function as the memory access instructions with immediate offset, see [LDR and STR, immediate offset](#). The difference is that these instructions have only unprivileged access even when used in privileged software.

When used in unprivileged software, these instructions behave in exactly the same way as normal memory access instructions with immediate offset.

Restrictions

In these instructions:

- Rn must not be PC

- Rt must not be SP and must not be PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|----------------|--------------|---|
| STRBTEQ | R4, [R7] | ; Conditionally store least significant byte in R4 to an address in R7, with unprivileged access |
| LDRHT | R2, [R2, #8] | ; Load halfword value from an address equal to sum of R2 and 8 into R2, with unprivileged access. |

LDR, PC-relative

Load register from memory.

Syntax

$LDR\{type\}\{cond\} Rt, label$
 $LDRD\{cond\} Rt, Rt2, label$; Load two words

where:

| type | is one of: |
|--------------|--|
| B | unsigned byte, zero extend to 32 bits. |
| SB | signed byte, sign extend to 32 bits. |
| H | unsigned halfword, zero extend to 32 bits. |
| SH | signed halfword, sign extend to 32 bits. |
| - | omit, for word. |
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>Rt</i> | is the register to load or store. |
| <i>Rt2</i> | is the second register to load or store. |
| <i>label</i> | is a PC-relative expression. See PC-relative expressions . |

Operation

LDR loads a register with a value from a PC-relative memory address. The memory address is specified by a label or by an offset from the PC.

The value to load or store can be a byte, halfword, or word. For load instructions, bytes and halfwords can either be signed or unsigned. See [Address alignment](#).

label must be within a limited range of the current instruction. Table 30 shows the possible offsets between *label* and the PC.

| Instruction type | Offset range |
|--|---------------------|
| Word, halfword, signed halfword, byte, signed byte | -4095 to 4095 |
| Two words | -1020 to 1020 |

[31 . Offset Ranges](#)

Note:

You might have to use the .W suffix to get the maximum offset range. See [Instruction width selection](#).

Restrictions

In these instructions:

- Rt can be SP or PC only for word loads
- $Rt2$ must not be SP and must not be PC
- Rt must be different from $Rt2$.

When Rt is PC in a word load instruction:

- bit[0] of the loaded value must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------------|-----------------|--|
| LDR | R0, LookUpTable | ; Load R0 with a word of data from an address labelled as LookUpTable |
| LDRSB | R7, localdata | ; Load a byte value from an address labelled as localdata, sign extend it to a word ; value, and put it in R7. |

LDM and STM

Load and Store Multiple registers.

Syntax

$op\{addr_mode\}\{cond\} Rn\{!\}, reglist$ where:

| | |
|------------------|---|
| op | is one of: LDM Load Multiple registers. STM Store Multiple registers. |
| addr-mode | is any one of the following: IA Increment address After each access. This is the default. DB Decrement address Before each access. |
| cond | is an optional condition code, see Conditional execution . |
| Rn | is the register on which the memory addresses are based. |
| ! | is an optional writeback suffix. If ! is present the final address, that is loaded from or stored to, is written back into Rn . |
| reglist | is a list of one or more registers to be loaded or stored, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range, see Examples . |

LDM and LDMFD are synonyms for LDMIA. LDMFD refers to its use for popping data from Full Descending stacks. LDMEA is a synonym for LDMDB, and refers to its use for popping data from Empty Ascending stacks.

STM and STMEA are synonyms for STMIA. STMEA refers to its use for pushing data onto Empty Ascending stacks. STMFD is a synonym for STMDB, and refers to its use for pushing data onto Full Descending stacks.

Operation

LDM instructions load the registers in *reglist* with word values from memory addresses based on *Rn*.

STM instructions store the word values in the registers in *reglist* to memory addresses based on *Rn*.

For LDM, LDMIA, LDMFD, STM, STMIA, and STMEA the memory addresses used for the accesses are at 4-byte intervals ranging from *Rn* to *Rn + 4 * (n-1)*, where *n* is the number of registers in *reglist*. The accesses happen in order of increasing register numbers, with the lowest numbered register using the lowest memory address and the highest number register using the highest memory address. If the writeback suffix is specified, the value of *Rn + 4 * (n-1)* is written back to *Rn*.

For LDMDB, LDMEA, STMDB, and STMFD the memory addresses used for the accesses are at 4-byte intervals ranging from *Rn* to *Rn - 4 * (n-1)*, where *n* is the number of registers in *reglist*. The accesses happen in order of decreasing register numbers, with the highest numbered register using the highest memory address and the lowest number register using the lowest memory address. If the writeback suffix is specified, the value of *Rn - 4 * (n-1)* is written back to *Rn*.

The PUSH and POP instructions can be expressed in this form. See *PUSH* and *POP* on page 3-33 for details.

Restrictions

In these instructions:

- *Rn* must not be PC
- *reglist* must not contain SP
- in any STM instruction, *reglist* must not contain PC
- in any LDM instruction, *reglist* must not contain PC if it contains LR
- *reglist* must not contain *Rn* if you specify the writeback suffix.

When PC is in *reglist* in an LDM instruction:

- bit[0] of the value loaded to the PC must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------------|---------------------|----------------------------|
| LDM | R8,{R0,R2,R9}; | LDMIA is a synonym for LDM |
| STMDB | R1!,{R3-R6,R11,R12} | |

Incorrect examples

| | |
|------------|---|
| STM | R5!,{R5,R4,R9} ; Value stored for R5 is unpredictable |
| LDM | R2, {} ; There must be at least one register in the list. |

PUSH and POP

Push registers onto, and pop registers off a full-descending stack.

Syntax

PUSH{cond} *reglist* POP{cond} *reglist* where: *cond* is an optional condition code, see [Conditional execution](#).

reglist is a non-empty list of registers, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range.

PUSH and POP are synonyms for STMDB and LDM (or LDMIA) with the memory addresses for the access based on

SP, and with the final address for the access written back to the SP. PUSH and POP are the preferred mnemonics in these cases.

Operation

PUSH stores registers on the stack, with the lowest numbered register using the lowest memory address and the highest numbered register using the highest memory address.

POP loads registers from the stack, with the lowest numbered register using the lowest memory address and the highest numbered register using the highest memory address.

PUSH uses the value in the SP register minus four as the highest memory address, POP uses the value in the SP register as the lowest memory address, implementing a full-descending stack. On completion, PUSH updates the SP register to point to the location of the lowest store value, POP updates the SP register to point to the location above the highest location loaded.

If a POP instruction includes PC in its reglist, a branch to this location is performed when the POP instruction has completed. Bit[0] of the value read for the PC is used to update the APSR T-bit. This bit must be 1 to ensure correct operation.

See *LDM* and *STM* for more information.

Restrictions

In these instructions:

- *reglist* must not contain SP
- for the PUSH instruction, *reglist* must not contain PC
- for the POP instruction, *reglist* must not contain PC if it contains LR.

When PC is in *reglist* in a POP instruction:

- bit[0] of the value loaded to the PC must be 1 for correct execution, and a branch occurs to this halfword-aligned address
- if the instruction is conditional, it must be the last instruction in the IT block.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|-------------|------------|---|
| PUSH | {R0,R4-R7} | ; Push R0,R4,R5,R6,R7 onto the stack |
| PUSH | {R2,LR} | ;Push R2 and the link-register onto the stack |
| POP | {R0,R6,PC} | ; Pop r0,r6 and PC from the stack, then branch to the new PC. |

LDREX and STREX

Load and Store Register Exclusive.

Syntax

LDREX{cond} Rt,[Rn{,#offset}]
STREX{cond} Rd,Rt,[Rn{,#offset}]

LDREXB{cond} Rt,[Rn]

STREXB{cond} Rd,Rt,[Rn] LDREXH{cond} Rt,[Rn] STREXH{cond} Rd,Rt,[Rn] where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register for the returned status.

Rt is the register to load or store.

Rn is the register on which the memory address is based.

offset is an optional offset applied to the value in *Rn*. If *offset* is omitted, the address is the value in *Rn*.

Operation

LDREX, LDREXB, and LDREXH load a word, byte, and halfword respectively from a memory address.

STREX, STREXB, and STREXH attempt to store a word, byte, and halfword respectively to a memory address. The address used in any Store-Exclusive instruction must be the same as the address in the most recently executed Load-exclusive instruction. The value stored by the Store-Exclusive instruction must also have the same data size as the value loaded by the preceding

Load-exclusive instruction. This means software must always use a Load-exclusive instruction and a matching Store-Exclusive instruction to perform a synchronization operation, see [Synchronization primitives](#).

If an Store-Exclusive instruction performs the store, it writes 0 to its destination register. If it does not perform the store, it writes 1 to its destination register. If the Store-Exclusive instruction writes 0 to the destination register, it is guaranteed that no other process in the system has accessed the memory location between the Load-exclusive and Store-Exclusive instructions.

For reasons of performance, keep the number of instructions between corresponding Load-Exclusive and Store-Exclusive instruction to a minimum.



Note

The result of executing a Store-Exclusive instruction to an address that is different from that used in the preceding Load-Exclusive instruction is unpredictable.

Restrictions

In these instructions:

- do not use PC
- do not use SP for *Rd* and *Rt*
- for STREX, *Rd* must be different from both *Rt* and *Rn*
- the value of *offset* must be a multiple of four in the range 0-1020.

Condition flags

These instructions do not change the flags.

Examples

| | | | |
|-----|---------|--------------------|--|
| | MOV | R1, #0x1 | ; Initialize the 'lock taken' value |
| try | LDREX | R0, [LockAddr] | ; Load the lock value |
| | CMP | R0, #0 | ; Is the lock free? |
| | ITT | EQ | ; IT instruction for STREXEQ and CMPEQ |
| | STREXEQ | R0, R1, [LockAddr] | ; Try and claim the lock |
| | CMPEQ | R0, #0 | ; Did this succeed? |
| | BNE | try | ; No – try again |
| | | | ; Yes – we have the lock. |

CLREX

Clear Exclusive.

Syntax

CLREX{*cond*} where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

Use CLREX to make the next STREX, STREXB, or STREXH instruction write 1 to its destination register and fail to perform the store. It is useful in exception handler code to force the failure of the store exclusive if the exception occurs between a load exclusive instruction and the matching store exclusive instruction in a synchronization operation.

See [Synchronization primitives](#) for more information.

Condition flags

This instruction does not change the flags.

Examples

CLREX

1.3.4 General data processing instructions

Table 31 shows the data processing instructions:

| Mnemonic | Brief description | See |
|----------|---|-----------------------------|
| ADC | Add with Carry | ADD, ADC, SUB, SBC, and RSB |
| ADD | Add | ADD, ADC, SUB, SBC, and RSB |
| ADDW | Add | ADD, ADC, SUB, SBC, and RSB |
| AND | Logical AND | AND, ORR, EOR, BIC, and ORN |
| ASR | Arithmetic Shift Right | ASR, LSL, LSR, ROR, and RRX |
| BIC | Bit Clear | AND, ORR, EOR, BIC, and ORN |
| CLZ | Count leading zeros | CLZ |
| CMN | Compare Negative | CMP and CMN |
| CMP | Compare | CMP and CMN |
| EOR | Exclusive OR | AND, ORR, EOR, BIC, and ORN |
| LSL | Logical Shift Left | ASR, LSL, LSR, ROR, and RRX |
| LSR | Logical Shift Right | ASR, LSL, LSR, ROR, and RRX |
| MOV | Move | MOV and MVN |
| MOVT | Move Top | MOVT |
| MOVW | Move 16-bit constant | MOV and MVN |
| MVN | Move NOT | MOV and MVN |
| ORN | Logical OR NOT | AND, ORR, EOR, BIC, and ORN |
| ORR | Logical OR | AND, ORR, EOR, BIC, and ORN |
| RBIT | Reverse Bit | REV, REV16, REVSH, and RBIT |
| REV | Reverse byte order in a word | REV, REV16, REVSH, and RBIT |
| REV16 | Reverse byte order in each halfword | REV, REV16, REVSH, and RBIT |
| REVSH | Reverse byte order in bottom halfword and sign extend | REV, REV16, REVSH, and RBIT |
| ROR | Rotate Right | ASR, LSL, LSR, ROR, and RRX |
| RRX | Rotate Right with Extend | ASR, LSL, LSR, ROR, and RRX |
| RSB | Reverse Subtract | ADD, ADC, SUB, SBC, and RSB |
| SADD16 | Signed Add 16 | SADD16 and SADD8 |
| SADD8 | Signed Add 8 | SADD16 and SADD8 |
| SASX | Signed Add and Subtract with Exchange | SASX and SSAX |
| SSAX | Signed Subtract and Add with Exchange | SASX and SSAX |

| Mnemonic | Brief description | See |
|----------|---------------------|---|
| SBC | Subtract with Carry | ADD, ADC, SUB, SBC, and RSB |

32 . Data processing instructions

| Mnemonic | Brief description | See |
|----------|---|---|
| SHADD16 | Signed Halving Add 16 | SHADD16 and SHADD8 |
| SHADD8 | Signed Halving Add 8 | SHADD16 and SHADD8 |
| SHASX | Signed Halving Add and Subtract with Exchange | SHASX and SHSAX |
| SHSAX | Signed Halving Subtract and Add with Exchange | SHASX and SHSAX |
| SHSUB16 | Signed Halving Subtract 16 | SHSUB16 and SHSUB8 |
| SHSUB8 | Signed Halving Subtract 8 | SHSUB16 and SHSUB8 |
| SSUB16 | Signed Subtract 16 | SSUB16 and SSUB8 |
| SSUB8 | Signed Subtract 8 | SSUB16 and SSUB8 |
| SUB | Subtract | ADD, ADC, SUB, SBC, and RSB |
| SUBW | Subtract | ADD, ADC, SUB, SBC, and RSB |
| TEQ | Test Equivalence | TST and TEQ |
| TST | Test | TST and TEQ |
| UADD16 | Unsigned Add 16 | UADD16 and UADD8 |
| UADD8 | Unsigned Add 8 | UADD16 and UADD8 |
| UASX | Unsigned Add and Subtract with Exchange | UASX and USAX |
| USAX | Unsigned Subtract and Add with Exchange | UASX and USAX |
| UHADD16 | Unsigned Halving Add 16 | UHADD16 and UHADD8 |
| UHADD8 | Unsigned Halving Add 8 | UHADD16 and UHADD8 |
| UHASX | Unsigned Halving Add and Subtract with Exchange | UHASX and UHSAX |
| UHSAX | Unsigned Halving Subtract and Add with Exchange | UHASX and UHSAX |
| UHSUB16 | Unsigned Halving Subtract 16 | UHSUB16 and UHSUB8 |
| UHSUB8 | Unsigned Halving Subtract 8 | UHSUB16 and UHSUB8 |
| USAD8 | Unsigned Sum of Absolute Differences | USAD8 |
| USADA8 | Unsigned Sum of Absolute Differences and Accumulate | USADA8 |
| USUB16 | Unsigned Subtract 16 | USUB16 and USUB8 |
| USUB8 | Unsigned Subtract 8 | USUB16 and USUB8 |

33 . Data processing instructions (continued)

ADD, ADC, SUB, SBC, and RSB

Add, Add with carry, Subtract, Subtract with carry, and Reverse Subtract.

Syntax

op{S}{cond} {Rd} Rn, Operand2

| | |
|---|---|
| <i>op{cond} {Rd} Rn, #imm12; ADD and SUB only</i> | |
| where: | |
| <i>op</i> | is one of: ADD Add. ADC Add with Carry. SUB Subtract. SBC Subtract with Carry. RSB Reverse Subtract. |

S is an optional suffix. If *S* is specified, the condition code flags are updated on the result of the operation, see [Conditional execution](#).

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See [Flexible second operand](#) for details of the options.

imm12 is any value in the range 0-4095.

Operation

The ADD instruction adds the value of *Operand2* or *imm12* to the value in *Rn*.

The ADC instruction adds the values in *Rn* and *Operand2*, together with the carry flag.

The SUB instruction subtracts the value of *Operand2* or *imm12* from the value in *Rn*.

The SBC instruction subtracts the value of *Operand2* from the value in *Rn*. If the carry flag is clear, the result is reduced by one.

The RSB instruction subtracts the value in *Rn* from the value of *Operand2*. This is useful because of the wide range of options for *Operand2*.

Use ADC and SBC to synthesize multiword arithmetic, see [Multiword arithmetic examples](#).

See also [ADR](#).



Note

ADDW is equivalent to the ADD syntax that uses the *imm12* operand. SUBW is equivalent to the SUB syntax that uses the *imm12* operand.

Restrictions

In these instructions:

- *Operand2* must not be SP and must not be PC
- *Rd* can be SP only in ADD and SUB, and only with the additional restrictions:

— *Rn* must also be SP
— any shift in *Operand2* must be limited to a maximum of 3 bits using LSL

- *Rn* can be SP only in ADD and SUB
- *Rd* can be PC only in the ADD{*cond*} PC, PC, Rm instruction where:

— you must not specify the *S* suffix
— *Rm* must not be PC and must not be SP
— if the instruction is conditional, it must be the last instruction in the IT block

- with the exception of the ADD{*cond*} PC, PC, Rm instruction, *Rn* can be PC only in ADD and SUB, and only with the additional restrictions:

- you must not specify the S suffix
- the second operand must be a constant in the range 0 to 4095.

Note:

- When using the PC for an addition or a subtraction, bits[1:0] of the PC are rounded to 0b00 before performing the calculation, making the base address for the calculation word-aligned.
- If you want to generate the address of an instruction, you have to adjust the constant based on the value of the PC. ARM recommends that you use the ADR instruction instead of ADD or SUB with Rn equal to the PC, because your assembler automatically calculates the correct constant for the ADR instruction.

When Rd is PC in the ADD{cond} PC, PC, Rm instruction:

- bit[0] of the value written to the PC is ignored
- a branch occurs to the address created by forcing bit[0] of that value to 0.

Condition flags

If S is specified, these instructions update the N, Z, C and V flags according to the result.

Examples

| | | |
|--------------|---|--|
| ADD | R2, R1, R3 | |
| SUBS | R8, R6, #240 ; Sets the flags on the result | |
| RSB | R4, R4, #1280; Subtracts contents of R4 from 1280 | |
| ADCHI | R11, R0, R3 ; Only executed if C flag set and Z flag clear. | |

Multiword arithmetic examples

Example shows two instructions that add a 64-bit integer contained in R2 and R3 to another 64-bit integer contained in R0 and R1, and place the result in R4 and R5.

Examples

| | | |
|-------------|------------|---|
| ADDS | R4, R0, R2 | add the least significant words |
| ADC | R5, R1, R3 | add the most significant words with carry |

Multiword values do not have to use consecutive registers. Example 3-5 shows instructions that subtract a 96-bit integer contained in R9, R1, and R11 from another contained in R6, R2, and R8. The example stores the result in R6, R9, and R2.

Examples 96-bit subtraction

| | | |
|-------------|---------|--|
| SUBS | R6, R9 | subtract the least significant words |
| SBCS | R2, R1 | subtract the middle words with carry |
| SBC | R8, R11 | subtract the most significant words with carry |

AND, ORR, EOR, BIC, and ORN

Logical AND, OR, Exclusive OR, Bit Clear, and OR NOT.

Syntax

op{S}{cond} {Rd,} Rn, Operand2 where:
op is one of:

| | |
|------------|--------------------------------|
| AND | logical AND. |
| ORR | logical OR, or bit set. |
| EOR | logical Exclusive OR. |
| BIC | logical AND NOT, or bit clear. |
| ORN | logical OR NOT. |

S is an optional suffix. If *S* is specified, the condition code flags are updated on the result of the operation, see [Conditional execution](#).

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See [Flexible second operand](#) for details of the options.

Operation

The AND, EOR, and ORR instructions perform bitwise AND, Exclusive OR, and OR operations on the values in *Rn* and *Operand2*.

The BIC instruction performs an AND operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

The ORN instruction performs an OR operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

Restrictions

Do not use SP and do not use PC.

Condition flags

If *S* is specified, these instructions:

- update the N and Z flags according to the result
- can update the C flag during the calculation of *Operand2*, see [Flexible second operand](#)
- do not affect the V flag.

Examples

| | |
|--------------|----------------------|
| AND | R9, R2, #0xFF00 |
| ORREQ | R2, R0, R5 |
| ANDS | R9, R8, #0x19 |
| EORS | R7, R11, #0x18181818 |
| BIC | R0, R1 #0xab |
| ORN | R7, R11, R14, ROR #4 |

ORNS

R7 R11, R14, ASR #32

ASR, LSL, LSR, ROR, and RRX

Arithmetic Shift Right, Logical Shift Left, Logical Shift Right, Rotate Right, and Rotate Right with Extend.

Syntax

$op[S]\{cond\} Rd, Rm, Rs op[S]\{cond\} Rd, Rm, \#n RRX[S]\{cond\} Rd, Rm$ where:
 op is one of:

| | |
|------------|-------------------------|
| ASR | Arithmetic Shift Right. |
| LSL | Logical Shift Left. |
| LSR | Logical Shift Right. |
| ROR | Rotate Right. |

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see [Conditional execution](#).

Rd is the destination register.

Rm is the register holding the value to be shifted.

Rs is the register holding the shift length to apply to the value in Rm . Only the least significant byte is used and can be in the range 0 to 255.

n is the shift length. The range of shift length depends on the instruction:

| | |
|------------|----------------------------|
| ASR | shift length from 1 to 32 |
| LSL | shift length from 0 to 31 |
| LSR | shift length from 1 to 32 |
| ROR | shift length from 1 to 31. |



Note
MOVS Rd, Rm is the preferred syntax for LSLS $Rd, Rm, \#0$

Operation

ASR, LSL, LSR, and ROR move the bits in the register Rm to the left or right by the number of places specified by constant n or register Rs .

RRX moves the bits in register Rm to the right by 1.

In all these instructions, the result is written to Rd , but the value in register Rm remains unchanged. For details on what result is generated by the different instructions, see [Shift Operations](#).

Restrictions

Do not use SP and do not use PC.

Condition flags

If S is specified:

- these instructions update the N and Z flags according to the result
- the C flag is updated to the last bit shifted out, except when the shift length is 0, see [Shift Operations](#).

Examples

| | | |
|-------------|------------|--|
| ASR | R7, R8, #9 | ; Arithmetic shift right by 9 bits |
| LSLS | R1, R2, #3 | ; Logical shift left by 3 bits with flag update |
| LSR | R4, R5, #6 | ; Logical shift right by 6 bits |
| ROR | R4, R5, R6 | ; Rotate right by the value in the bottom byte of R6 |
| RRX | R4, R5 | ; Rotate right with extend. |

CLZ

Count Leading Zeros.

Syntax

CLZ{cond} Rd, Rm where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rm is the operand register.

Operation

The CLZ instruction counts the number of leading zeros in the value in *Rm* and returns the result in *Rd*. The result value is 32 if no bits are set and zero if bit[31] is set.

Restrictions

Do not use SP and do not use PC.

Condition flags

This instruction does not change the flags.

Examples

| | |
|-----|-------------|
| CLZ | R4, R9 |
| | CLZNER2, R3 |

CMP and CMN

Compare and Compare Negative.

Syntax

CMP{cond} Rn, Operand2 CMN{cond} Rn, Operand2 where: *cond* is an optional condition code, see [Conditional execution](#).

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See [Flexible second operand](#) for details of the options.

Operation

These instructions compare the value in a register with *Operand2*. They update the condition flags on the result, but do not write the result to a register.

The CMP instruction subtracts the value of *Operand2* from the value in *Rn*. This is the same as a SUBS instruction, except that the result is discarded.

The CMN instruction adds the value of *Operand2* to the value in *Rn*. This is the same as an ADDS instruction, except that the result is discarded.

Restrictions

In these instructions:

- do not use PC
- *Operand2* must not be SP.

Condition flags

These instructions update the N, Z, C and V flags according to the result.

Examples

| | |
|-------|----------------|
| CMP | R2, R9 |
| CMN | R0, #6400 |
| CMPGT | SP, R7, LSL #2 |

MOV and MVN

Move and Move NOT.

Syntax

$\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Operand2}$ $\text{MOV}\{\text{cond}\} \text{Rd}, \#imm16$ $\text{MVN}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Operand2}$ where:
S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see [Conditional execution](#).

cond is an optional condition code, see [Conditional execution](#) on page 3-17.

Rd is the destination register.

Operand2 is a flexible second operand. See [Flexible second operand](#) for details of the options.

imm16 is any value in the range 0-65535.

Operation

The MOV instruction copies the value of *Operand2* into *Rd*.

When *Operand2* in a MOV instruction is a register with a shift other than LSL #0, the preferred syntax is the corresponding shift instruction:

- ASR{S}{cond} Rd, Rm, #n is the preferred syntax for $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{ASR } \#n$
- LSL{S}{cond} Rd, Rm, #n is the preferred syntax for $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{LSL } \#n$ if $n \neq 0$
- LSR{S}{cond} Rd, Rm, #n is the preferred syntax for $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{LSR } \#n$
- ROR{S}{cond} Rd, Rm, #n is the preferred syntax for $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{ROR } \#n$
- RRX{S}{cond} Rd, Rm is the preferred syntax for $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{RRX}$.

Also, the MOV instruction permits additional forms of *Operand2* as synonyms for shift instructions:

- $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{ASR } \text{Rs}$ is a synonym for $\text{ASR}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{Rs}$
- $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{LSL } \text{Rs}$ is a synonym for $\text{LSL}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{Rs}$
- $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{LSR } \text{Rs}$ is a synonym for $\text{LSR}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{Rs}$
- $\text{MOV}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{ROR } \text{Rs}$ is a synonym for $\text{ROR}\{\text{S}\}\{\text{cond}\} \text{Rd}, \text{Rm}, \text{Rs}$

See [ASR, LSL, LSR, ROR, and RRX](#) on page 3-45.

The MVN instruction takes the value of *Operand2*, performs a bitwise logical NOT operation on the value, and places the result into *Rd*.



Note

The MOVW instruction provides the same function as MOV, but is restricted to using the *imm16* operand.

Restrictions

You can use SP and PC only in the MOV instruction, with the following restrictions:

- the second operand must be a register without shift
- you must not specify the S suffix.

When *Rd* is PC in a MOV instruction:

- bit[0] of the value written to the PC is ignored
- a branch occurs to the address created by forcing bit[0] of that value to 0.

⚠ Note

Though it is possible to use MOV as a branch instruction, ARM strongly recommends the use of a BX or BLX instruction to branch for software portability to the ARM instruction set.

Condition flags

If S is specified, these instructions:

- update the N and Z flags according to the result
- can update the C flag during the calculation of *Operand2*, see [Flexible second operand](#)
- do not affect the V flag.

Example

| | | |
|------|--------------|--|
| MOVS | R11, #0x000B | ; Write value of 0x000B to R11, flags get updated |
| MOV | R1, #0xFA05 | ; Write value of 0xFA05 to R1, flags are not updated |
| MOVS | R10, R12 | ; Write value in R12 to R10, flags get updated |
| MOV | R3, #23 | ; Write value of 23 to R3 |
| MOV | R8, SP | ; Write value of stack pointer to R8 |
| MVNS | R2, #0xF | ; Write value of 0xFFFFFFF0 (bitwise inverse of 0xF) ; to the R2 and update flags. |

MOVT

Move Top.

Syntax

MOVT{cond} Rd, #imm16 where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

imm16 is a 16-bit immediate constant.

Operation

MOVT writes a 16-bit immediate value, *imm16*, to the top halfword, *Rd*[31:16], of its destination register. The write does not affect *Rd*[15:0].

The MOV, MOVT instruction pair enables you to generate any 3bit constant.

Restrictions

Rd must not be SP and must not be PC.

Condition flags

This instruction does not change the flags.

Examples

| | | |
|------|---------|--|
| MOVT | #0xF123 | ; Write 0xF123 to upper halfword of R3, lower halfword and APSR are unchanged. |
|------|---------|--|

REV, REV16, REVSH, and RBIT

Reverse bytes and Reverse bits.

Syntax

op{cond} Rd, Rn where:

op is any of:

| | |
|--------------|--|
| REV | Reverse byte order in a word. |
| REV16 | Reverse byte order in each halfword independently. |
| REVSH | Reverse byte order in the bottom halfword, and sign extend to 32 bits. |
| RBIT | Reverse the bit order in a 3bit word. |

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the register holding the operand.

Operation

Use these instructions to change endianness of data:

| | |
|--------------|--|
| REV | converts either: <ul style="list-style-type: none"> • 3bit big-endian data into little-endian data • 3bit little-endian data into big-endian data. |
| REV16 | converts either: <ul style="list-style-type: none"> • 16-bit big-endian data into little-endian data • 16-bit little-endian data into big-endian data. |
| REVSH | converts either: <ul style="list-style-type: none"> • 16-bit signed big-endian data into 3bit signed little-endian data • 16-bit signed little-endian data into 3bit signed big-endian data. |

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------------|--------|--|
| REV | R3, R7 | ; Reverse byte order of value in R7 and write it to R3 |
| REV16 | R0, R0 | ; Reverse byte order of each 16-bit halfword in R0 |
| REVSH | R0, R5 | ; Reverse Signed Halfword |
| REVHS | R3, R7 | ; Reverse with Higher or Same condition |
| RBIT | R7, R8 | ; Reverse bit order of value in R8 and write the result to R7. |

SADD16 and SADD8

Signed Add 16 and Signed Add 8

Syntax

op{cond}{Rd} Rn, Rm where:

op is any of:

SADD16 Performs two 16-bit signed integer additions.

SADD8 Performs four 8-bit signed integer additions.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first register holding the operand.

Rm is the second register holding the operand.

Operation

Use these instructions to perform a halfword or byte add in parallel:

The SADD16 instruction:

1. Adds each halfword from the first operand to the corresponding halfword of the second operand.
2. Writes the result in the corresponding halfwords of the destination register.

The SADD8 instruction:

1. Adds each byte of the first operand to the corresponding byte of the second operand.
2. Writes the result in the corresponding bytes of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------|------------|---|
| SADD16 | R1, R0 | ; Adds the halfwords in R0 to the corresponding halfwords of R1 and writes to corresponding halfword of R1. |
| SADD8 | R4, R0, R5 | ; Adds bytes of R0 to the corresponding byte in R5 and writes to the corresponding byte in R4. |

SHADD16 and SHADD8

Signed Halving Add 16 and Signed Halving Add 8

Syntax

op{cond}{Rd} Rn, Rm where:

op is any of:

SHADD16 Signed Halving Add 16

SHADD8 Signed Halving Add 8

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register.

Rm is the second operand register.

Operation

Use these instructions to add 16-bit and 8-bit data and then to halve the result before writing the result to the destination register:

The SHADD16 instruction:

1. Adds each halfword from the first operand to the corresponding halfword of the second operand.
2. Shuffles the result by one bit to the right, halving the data.
3. Writes the halfword results in the destination register.

The SHADDB8 instruction:

1. Adds each byte of the first operand to the corresponding byte of the second operand.
2. Shuffles the result by one bit to the right, halving the data.

-
3. Writes the byte results in the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|---------|------------|---|
| SHADD16 | R1, R0 | ; Adds halfwords in R0 to corresponding halfword of R1 and writes halved result to corresponding halfword in R1 |
| SHADD8 | R4, R0, R5 | ; Adds bytes of R0 to corresponding byte in R5 and writes halved result to corresponding byte in R4. |

SHASX and SHSAX

Signed Halving Add and Subtract with Exchange and Signed Halving Subtract and Add with Exchange.

Syntax

op{cond} {Rd}, Rn, Rm where:

| | |
|---------------|---|
| <i>op</i> | is any of: SHASX Add and Subtract with Exchange and Halving. SHSAX Subtract and Add with Exchange and Halving. |
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>Rd</i> | is the destination register. |
| <i>Rn, Rm</i> | are registers holding the first and second operands. |

Operation

The SHASX instruction:

1. Adds the top halfword of the first operand with the bottom halfword of the second operand.
2. Writes the halfword result of the addition to the top halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.
3. Subtracts the top halfword of the second operand from the bottom highword of the first operand.
4. Writes the halfword result of the division in the bottom halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.

The SHSAX instruction:

1. Subtracts the bottom halfword of the second operand from the top highword of the first operand.
2. Writes the halfword result of the subtraction to the bottom halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.
3. Adds the bottom halfword of the first operand with the top halfword of the second operand.
4. Writes the halfword result of the addition in the top halfword of the destination register, shifted by one bit to the right causing a divide by two, or halving.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|-------|------------|--|
| SHASX | R7, R4, R2 | ; Adds top halfword of R4 to bottom halfword of R2 and writes halved result to top halfword of R7 Subtracts top halfword of R2 from bottom halfword of R4 and writes halved result to bottom halfword of R7 |
| SHSAX | R0, R3, R5 | ; Subtracts bottom halfword of R5 from top halfword of R3 and writes halved result to top halfword of R0 . Adds top halfword of R5 to bottom halfword of R3 and writes halved result to bottom halfword of R0. |

SHSUB16 and SHSUB8

Signed Halving Subtract 16 and Signed Halving Subtract 8

Syntax

op{cond}{Rd,} Rn, Rm where:
op is any of:

SHSUB16 Signed Halving Subtract 16

SHSUB8 Signed Halving Subtract 8

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register.

Rm is the second operand register

Operation

Use these instructions to add 16-bit and 8-bit data and then to halve the result before writing the result to the destination register:

The SHSUB16 instruction:

1. Subtracts each halfword of the second operand from the corresponding halfwords of the first operand.
2. Shuffles the result by one bit to the right, halving the data.
3. Writes the halved halfword results in the destination register.

The SHSUBB8 instruction:

1. Subtracts each byte of the second operand from the corresponding byte of the first operand,
2. Shuffles the result by one bit to the right, halving the data,
3. Writes the corresponding signed byte results in the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|---------|------------|--|
| SHSUB16 | R1, R0 | ; Subtracts halfwords in R0 from corresponding halfword of R1 and writes to corresponding halfword of R1 |
| SHSUB8 | R4, R0, R5 | ; Subtracts bytes of R0 from corresponding byte in R5, and writes to corresponding byte in R4. |

SSUB16 and SSUB8

Signed Subtract 16 and Signed Subtract 8

Syntax

op{cond}{Rd,} Rn, Rm where:
op is any of:

SSUB16 Performs two 16-bit signed integer subtractions.

SSUB8 Performs four 8-bit signed integer subtractions.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.
Rn is the first operand register.
Rm is the second operand register.

Operation

Use these instructions to change endianness of data:

The SSUB16 instruction:

1. Subtracts each halfword from the second operand from the corresponding halfword of the first operand
2. Writes the difference result of two signed halfwords in the corresponding halfword of the destination register.

The SSUB8 instruction:

1. Subtracts each byte of the second operand from the corresponding byte of the first operand
2. Writes the difference result of four signed bytes in the corresponding byte of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------|------------|--|
| SSUB16 | R1, R0 | ; Subtracts halfwords in R0 from corresponding halfword of R1 and writes to corresponding halfword of R1 |
| SSUB8 | R4, R0, R5 | ; Subtracts bytes of R5 from corresponding byte in R0, and writes to corresponding byte of R4. |

SASX and SSAX

Signed Add and Subtract with Exchange and Signed Subtract and Add with Exchange.

Syntax

op{cond} {Rd}, Rm, Rn where:

| | |
|---------------|---|
| <i>op</i> | is any of: SASX Signed Add and Subtract with Exchange. SSAX Signed Subtract and Add with Exchange. |
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>Rd</i> | is the destination register. |
| <i>Rn, Rm</i> | are registers holding the first and second operands. |

Operation

The SASX instruction:

1. Adds the signed top halfword of the first operand with the signed bottom halfword of the second operand.
2. Writes the signed result of the addition to the top halfword of the destination register.
3. Subtracts the signed bottom halfword of the second operand from the top signed highword of the first operand.
4. Writes the signed result of the subtraction to the bottom halfword of the destination register.

The SSAX instruction:

1. Subtracts the signed bottom halfword of the second operand from the top signed highword of the first operand.
2. Writes the signed result of the addition to the bottom halfword of the destination register.

-
- 3. Adds the signed top halfword of the first operand with the signed bottom halfword of the second operand.
 - 4. Writes the signed result of the subtraction to the top halfword of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|------|------------|--|
| SASX | R0, R4, R5 | <p>; Adds top halfword of R4 to bottom halfword of R5 and</p> <p>; writes to top halfword of R0 ; Subtracts bottom halfword of R5 from top halfword of R4</p> <p>; and writes to bottom halfword of R0</p> |
| SSAX | R3, R2 | <p>; Subtracts top halfword of R2 from bottom halfword of R3</p> <p>; and writes to bottom halfword of R7</p> <p>; Adds top halfword of R3 with bottom halfword of R2 and</p> <p>; writes to top halfword of R7.</p> |

TST and TEQ

Test bits and Test Equivalence.

Syntax

TST{cond} *Rn*, *Operand2* TEQ{cond} *Rn*, *Operand2* where: *cond* is an optional condition code, see [Conditional execution](#).

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See [Flexible second operand](#) for details of the options.

Operation

These instructions test the value in a register against *Operand2*. They update the condition flags based on the result, but do not write the result to a register.

The TST instruction performs a bitwise AND operation on the value in *Rn* and the value of *Operand2*. This is the same as the ANDS instruction, except that it discards the result.

To test whether a bit of *Rn* is 0 or 1, use the TST instruction with an *Operand2* constant that has that bit set to 1 and all other bits cleared to 0.

The TEQ instruction performs a bitwise Exclusive OR operation on the value in *Rn* and the value of *Operand2*. This is the same as the EORS instruction, except that it discards the result.

Use the TEQ instruction to test if two values are equal without affecting the V or C flags.

TEQ is also useful for testing the sign of a value. After the comparison, the N flag is the logical Exclusive OR of the sign bits of the two operands.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions:

- update the N and Z flags according to the result.
- can update the C flag during the calculation of *Operand2*, see [Flexible second operand](#).
- do not affect the V flag.

Examples

| | | |
|-----|------------|---|
| TST | R0, #0x3F8 | <p>; Perform bitwise AND of R0 value to 0x3F8,</p> <p>; APSR is updated but result is discarded</p> |
|-----|------------|---|

| | | |
|-------|---------|---|
| TEQEQ | R10, R9 | ; Conditionally test if value in R10 is equal to ; value in R9, APSR is updated but result is discarded. |
|-------|---------|---|

UADD16 and UADD8

Unsigned Add 16 and Unsigned Add 8

Syntax

$op\{cond\}\{Rd\} Rn, Rm$ where:

op is any of:

UADD16 Performs two 16-bit unsigned integer additions.

UADD8 Performs four 8-bit unsigned integer additions.

$cond$ is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first register holding the operand.

Rm is the second register holding the operand.

Operation

Use these instructions to add 16- and 8-bit unsigned data:

The UADD16 instruction:

1. adds each halfword from the first operand to the corresponding halfword of the second operand.
2. writes the unsigned result in the corresponding halfwords of the destination register.

The UADD16 instruction:

1. adds each byte of the first operand to the corresponding byte of the second operand.
2. writes the unsigned result in the corresponding byte of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------|------------|---|
| UADD16 | R1, R0 | ; Adds halfwords in R0 to corresponding halfword of R1, ; writes to corresponding halfword of R1 |
| UADD8 | R4, R0, R5 | ; Adds bytes of R0 to corresponding byte in R5 and writes ; to corresponding byte in R4. |

UASX and USAX

Add and Subtract with Exchange and Subtract and Add with Exchange.

Syntax

$op\{cond\}\{Rd\}, Rn, Rm$ where:

| | |
|----------|---|
| op | is one of: UASX Add and Subtract with Exchange. USAX Subtract and Add with Exchange. |
| $cond$ | is an optional condition code, see Conditional execution . |
| Rd | is the destination register. |
| Rn, Rm | are registers holding the first and second operands. |

Operation

The UASX instruction:

1. Subtracts the top halfword of the second operand from the bottom halfword of the first operand.
2. Writes the unsigned result from the subtraction to the bottom halfword of the destination register.
3. Adds the top halfword of the first operand with the bottom halfword of the second operand.
4. Writes the unsigned result of the addition to the top halfword of the destination register.

The USAX instruction:

1. Adds the bottom halfword of the first operand with the top halfword of the second operand.
2. Writes the unsigned result of the addition to the bottom halfword of the destination register.
3. Subtracts the bottom halfword of the second operand from the top halfword of the first operand.
4. Writes the unsigned result from the subtraction to the top halfword of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|------|------------|---|
| UASX | R0, R4, R5 | ; Adds top halfword of R4 to bottom halfword of R5 and ; writes to top halfword of R0 ; Subtracts bottom halfword of R5 from top halfword of R0 ; and writes to bottom halfword of R0 |
| USAX | R7, R3, R2 | ; Subtracts top halfword of R2 from bottom halfword of R3 ; and writes to bottom halfword of R7 ; Adds top halfword of R3 to bottom halfword of R2 and ; writes to top halfword of R7. |

UHADD16 and UHADD8

Unsigned Halving Add 16 and Unsigned Halving Add 8

Syntax

op{cond}{Rd,} Rn, Rm where:

op is any of:

UHADD16 Unsigned Halving Add 16.

UHADD8 Unsigned Halving Add 8.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the register holding the first operand.

Rm is the register holding the second operand.

Operation

Use these instructions to add 16- and 8-bit data and then to halve the result before writing the result to the destination register:

The UHADD16 instruction:

1. Adds each halfword from the first operand to the corresponding halfword of the second operand.
2. Shuffles the halfword result by one bit to the right, halving the data.
3. Writes the unsigned results to the corresponding halfword in the destination register.

The UHADD8 instruction:

1. Adds each byte of the first operand to the corresponding byte of the second operand.
2. Shuffles the byte result by one bit to the right, halving the data.

-
3. Writes the unsigned results in the corresponding byte in the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|---------|------------|--|
| UHADD16 | R7, R3 | ; Adds halfwords in R7 to corresponding halfword of R3 ; and writes halved result to corresponding halfword in R7 |
| UHADD8 | R4, R0, R5 | ; Adds bytes of R0 to corresponding byte in R5 and writes ; halved result to corresponding byte in R4. |

UHASX and UHSAX

Unsigned Halving Add and Subtract with Exchange and Unsigned Halving Subtract and Add with Exchange.

Syntax

op{cond} {Rd}, Rn, Rm where:

| | |
|---------------|---|
| <i>op</i> | is one of: UHASX Add and Subtract with Exchange and Halving. UHSAX Subtract and Add with Exchange and Halving. |
| <i>cond</i> | is an optional condition code, see <i>Conditional execution</i> . |
| <i>Rd</i> | is the destination register. |
| <i>Rn, Rm</i> | are registers holding the first and second operands. |

Operation

The UHASX instruction:

1. Adds the top halfword of the first operand with the bottom halfword of the second operand.
2. Shifts the result by one bit to the right causing a divide by two, or halving.
3. Writes the halfword result of the addition to the top halfword of the destination register.
4. Subtracts the top halfword of the second operand from the bottom highword of the first operand.
5. Shifts the result by one bit to the right causing a divide by two, or halving.
6. Writes the halfword result of the division in the bottom halfword of the destination register.

The UHSAX instruction:

1. Subtracts the bottom halfword of the second operand from the top highword of the first operand.
2. Shifts the result by one bit to the right causing a divide by two, or halving.
3. Writes the halfword result of the subtraction in the top halfword of the destination register.
4. Adds the bottom halfword of the first operand with the top halfword of the second operand.
5. Shifts the result by one bit to the right causing a divide by two, or halving.
6. Writes the halfword result of the addition to the bottom halfword of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|-------|------------|---|
| UHASX | R7, R4, R2 | ; Adds top halfword of R4 with bottom halfword of R2 ; and writes halved result to top halfword of R7 ; Subtracts top halfword of R2 from bottom halfword of ; R7 and writes halved result to bottom halfword of R7 |
| UHSAX | R0, R3, R5 | ; Subtracts bottom halfword of R5 from top halfword of ; R3 and writes halved result to top halfword of R0 ; Adds top halfword of R5 to bottom halfword of R3 and ; writes halved result to bottom halfword of R0. |

UHSUB16 and UHSUB8

Unsigned Halving Subtract 16 and Unsigned Halving Subtract 8

Syntax

op{cond}{Rd,} Rn, Rm where:

op is any of:

UHSUB16 Performs two unsigned 16-bit integer additions, halves the results, and writes the results to the destination register.

UHSUB8 Performs four unsigned 8-bit integer additions, halves the results, and writes the results to the destination register.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first register holding the operand.

Rm is the second register holding the operand.

Operation

Use these instructions to add 16-bit and 8-bit data and then to halve the result before writing the result to the destination register:

The UHSUB16 instruction:

1. Subtracts each halfword of the second operand from the corresponding halfword of the first operand.
2. Shuffles each halfword result to the right by one bit, halving the data.
3. Writes each unsigned halfword result to the corresponding halfwords in the destination register.

The UHSUB8 instruction:

1. Subtracts each byte of second operand from the corresponding byte of the first operand.
2. Shuffles each byte result by one bit to the right, halving the data.
3. Writes the unsigned byte results to the corresponding byte of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|---------|------------|--|
| UHSUB16 | R1, R0 | ; Subtracts halfwords in R0 from corresponding halfword of ; R1 and writes halved result to corresponding halfword in ; R1 |
| UHSUB8 | R4, R0, R5 | ; Subtracts bytes of R5 from corresponding byte in R0 and ; writes halved result to corresponding byte in R4. |

SEL

Select Bytes. Selects each byte of its result from either its first operand or its second operand, according to the values of the GE flags.

Syntax

SEL{<c>}{<q>} {<Rd>}, <Rn>, <Rm> where:

| | |
|----------|--|
| <c>, <q> | is a standard assembler syntax fields. |
| <Rd> | is the destination register. |
| <Rn> | is the first operand register. |
| <Rm> | is the second operand register. |

Operation

The SEL instruction:

1. Reads the value of each bit of APSR.GE.
2. Depending on the value of APSR.GE, assigns the destination register the value of either the first or second operand register.

Restrictions None.

Condition flags

These instructions do not change the flags.

Examples

| | |
|--------|---|
| SADD16 | R0, R1, R2 ; Set GE bits based on result |
| SEL | R0, R0, R3 ; Select bytes from R0 or R3, based on GE. |

USAD8

Unsigned Sum of Absolute Differences

Syntax

USAD8{cond}{Rd,} Rn, Rm where: cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register.

Rm is the second operand register.

Operation

The USAD8 instruction:

1. Subtracts each byte of the second operand register from the corresponding byte of the first operand register.
2. Adds the absolute values of the differences together.
3. Writes the result to the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | |
|-------|--|
| USAD8 | R1, R4, R0 ; Subtracts each byte in R0 from corresponding byte of R4 |
| | ; adds the differences and writes to R1 |

| | | |
|-------|--------|--|
| USAD8 | R0, R5 | ; Subtracts bytes of R5 from corresponding byte in R0, ; adds the differences and writes to R0. |
|-------|--------|--|

USADA8

Unsigned Sum of Absolute Differences and Accumulate

Syntax

USADA8{cond}{Rd;} Rn, Rm, Ra where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register.

Rm is the second operand register.

Ra is the register that contains the accumulation value.

Operation

The USADA8 instruction:

1. Subtracts each byte of the second operand register from the corresponding byte of the first operand register.
2. Adds the unsigned absolute differences together.
3. Adds the accumulation value to the sum of the absolute differences.
4. Writes the result to the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------|----------------|---|
| USADA8 | R1, R0, R6 | ; Subtracts bytes in R0 from corresponding halfword of R1 ; adds differences, adds value of R6, writes to R1 |
| USADA8 | R4, R0, R5, R2 | ; Subtracts bytes of R5 from corresponding byte in R0 ; adds differences, adds value of R2 writes to R4. |

USUB16 and USUB8

Unsigned Subtract 16 and Unsigned Subtract 8

Syntax

op{cond}{Rd;} Rn, Rm where:

op is any of:

USUB16 Unsigned Subtract 16.

USUB8 Unsigned Subtract 8.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register.

Rm is the second operand register.

Operation

Use these instructions to subtract 16-bit and 8-bit data before writing the result to the destination register:

The USUB16 instruction:

1. Subtracts each halfword from the second operand register from the corresponding halfword of the first operand register.
2. Writes the unsigned result in the corresponding halfwords of the destination register.

The USUB8 instruction:

1. Subtracts each byte of the second operand register from the corresponding byte of the first operand register.
2. Writes the unsigned byte result in the corresponding byte of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|--------|------------|---|
| USUB16 | R1, R0 | ; Subtracts halfwords in R0 from corresponding halfword of R1 ; and writes to corresponding halfword in R1 |
| USUB8 | R4, R0, R5 | ; Subtracts bytes of R5 from corresponding byte in R0 and ; writes to the corresponding byte in R4. |

1.3.5 Multiply and divide instructions

Table 32 shows the multiply and divide instructions:

| Mnemonic | Brief description | See |
|--------------------|--|--|
| MLA | Multiply with Accumulate, 3bit result | MUL, MLA, and MLS |
| MLS | Multiply and Subtract, 3bit result | MUL, MLA, and MLS |
| MUL | Multiply, 3bit result | MUL, MLA, and MLS |
| SDIV | Signed Divide | SDIV and UDIV |
| SMLA[B,T] | Signed Multiply Accumulate (halfwords) | SMLA and SMLAW |
| SMLAD, SMLADX | Signed Multiply Accumulate Dual | SMLAD |
| SMLAL | Signed Multiply with Accumulate (32x32+64), 64-bit result | UMULL, UMLAL, SMULL, and SMLAL |
| SMLAL[B,T] | Signed Multiply Accumulate Long (halfwords) | SMLAL and SMLAD |
| SMLALD, SMLALDX | Signed Multiply Accumulate Long Dual | SMLAL and SMLAD |
| SMLAW[B,T] | Signed Multiply Accumulate (word by halfword) | SMLA and SMLAW |
| SMLSD | Signed Multiply Subtract Dual | SMLSD and SMLS |
| SMLS | Signed Multiply Subtract Long Dual | SMLSD and SMLS |
| SMMLA | Signed Most Significant Word Multiply Accumulate | SMMLA and SMMLS |
| SMMLS, SMMLSR | Signed Most Significant Word Multiply Subtract | SMMLA and SMMLS |
| SMUAD, SMUADX | Signed Dual Multiply Add | SMUAD and SMUSD |
| SMUL[B,T] | Signed Multiply (word by halfword) | SMUL and SMULW |

| Mnemonic | Brief description | See |
|-------------------|--|--|
| SMMUL, SMMULR | Signed Most Significant Word Multiply | SMMUL |
| SMULL | Signed Multiply (32x32), 64-bit result | UMULL, UMLAL, SMULL, and SMLAL |
| SMULWB, SMULWT | Signed Multiply (word by halfword) | SMUL and SMULW |
| SMUSD, SMUSDX | Signed Dual Multiply Subtract | SMUAD and SMUSD |
| UDIV | Unsigned Divide | SDIV and UDIV |
| UMAAL | Unsigned Multiply Accumulate Accumulate Long (32x32+32+32), 64-bit result | UMULL, UMAAL, UMLAL |
| UMLAL | Unsigned Multiply with Accumulate (32x32+64), 64-bit result | UMULL, UMLAL, SMULL, and SMLAL |
| UMULL | Unsigned Multiply (32x32), 64-bit result | UMULL, UMLAL, SMULL, and SMLAL |

34 . Multiply and divide instructions

MUL, MLA, and MLS

Multiply, Multiply with Accumulate, and Multiply with Subtract, using 3bit operands, and producing a 3bit result.

Syntax

`MUL{S}{cond} {Rd}, Rn, Rm ; Multiply`

`MLA{cond} Rd, Rn, Rm, Ra ; Multiply with accumulate` `MLS{cond} Rd, Rn, Rm, Ra ; Multiply with subtract` where: *cond* is an optional condition code, see [Conditional execution](#).

S is an optional suffix. If *S* is specified, the condition code flags are updated on the result of the operation, see [Conditional execution](#).

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn, *Rm* are registers holding the values to be multiplied.

Ra is a register holding the value to be added or subtracted from.

Operation

The MUL instruction multiplies the values from *Rn* and *Rm*, and places the least significant 32 bits of the result in *Rd*.

The MLA instruction multiplies the values from *Rn* and *Rm*, adds the value from *Ra*, and places the least significant 32 bits of the result in *Rd*.

The MLS instruction multiplies the values from *Rn* and *Rm*, subtracts the product from the value from *Ra*, and places the least significant 32 bits of the result in *Rd*.

The results of these instructions do not depend on whether the operands are signed or unsigned.

Restrictions

In these instructions, do not use SP and do not use PC.

If you use the *S* suffix with the MUL instruction:

- *Rd*, *Rn*, and *Rm* must all be in the range R0 to R7
- *Rd* must be the same as *Rm*
- you must not use the *cond* suffix.

Condition flags

If *S* is specified, the MUL instruction:

- updates the N and Z flags according to the result
- does not affect the C and V flags.

Examples

| | | |
|--------------|-----------------|--|
| MUL | R10, R2, R5 | ; Multiply, R10 = R2 x R5 |
| MLA | R10, R2, R1, R5 | ; Multiply with accumulate, R10 = (R2 x R1) + R5 |
| MULS | R0, R2, R2 | ; Multiply with flag update, R0 = R2 x R2 |
| MULLT | R2, R3, R2 | ; Conditionally multiply, R2 = R3 x R2 |
| MLS | R4, R5, R6, R7 | ; Multiply with subtract, R4 = R7 - (R5 x R6) |

UMULL, UMAAL, UMLAL

Unsigned Long Multiply, with optional Accumulate, using 3bit operands and producing a 64-bit result.

Syntax

op{cond} RdLo, RdHi, Rn, Rm where:

| | |
|-------------------|---|
| op | is one of: UMULL Unsigned Long Multiply. UMAAL Unsigned Long Multiply with Accumulate Accumulate. UMLAL Unsigned Long Multiply, with Accumulate. |
| cond | is an optional condition code, see <i>Conditional execution</i> . |
| RdHi, RdLo | are the destination registers. For UMAAL, UMLAL and UMLAL they also hold the accumulating value. |
| Rn, Rm | are registers holding the first and second operands. |

Operation

These instructions interpret the values from *Rn* and *Rm* as unsigned 3bit integers. The UMULL instruction:

- Multiplies the two unsigned integers in the first and second operands.
- Writes the least significant 32 bits of the result in *RdLo*.
- Writes the most significant 32 bits of the result in *RdHi*.

The UMAAL instruction:

- Multiplies the two unsigned 3bit integers in the first and second operands.
- Adds the unsigned 3bit integer in *RdHi* to the 64-bit result of the multiplication.
- Adds the unsigned 3bit integer in *RdLo* to the 64-bit result of the addition.
- Writes the top 3bits of the result to *RdHi*.
- Writes the lower 3bits of the result to *RdLo*.

The UMLAL instruction:

- multiplies the two unsigned integers in the first and second operands.
- Adds the 64-bit result to the 64-bit unsigned integer contained in *RdHi* and *RdLo*.
- Writes the result back to *RdHi* and *RdLo*.

Restrictions

In these instructions:

- do not use SP and do not use PC.

- $RdHi$ and $RdLo$ must be different registers.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|--------------|----------------|--|
| UMULL | R0, R4, R5, R6 | ; Multiplies R5 and R6, writes the top 32 bits to R4 and the bottom 32 bits to R0 |
| UMAAL | R3, R6, R2, R7 | ; Multiplies R2 and R7, adds R6, adds R3, writes the top 32 bits to R6, and the bottom 32 bits to R3 |
| UMLAL | R2, R1, R3, R5 | ; Multiplies R5 and R3, adds R1:R2, writes to R1:R2 |

SMLA and SMLAW

Signed Multiply Accumulate (halfwords).

Syntax

$op\{XY\}\{cond\} Rd, Rn, Rm op\{Y\}\{cond\} Rd, Rn, Rm, Ra$ where
 op is one of:

SMLA Signed Multiply Accumulate Long (halfwords)

1. and Y specifies which half of the source registers Rn and Rm are used as the first and second multiply operand.

If X is B , then the bottom halfword, bits [15:0], of Rn is used. If X is T , then the top halfword, bits [31:16], of Rn is used.

If Y is B , then the bottom halfword, bits [15:0], of Rm is used. If Y is T , then the top halfword, bits [31:16], of Rm is used.

SMLAW Signed Multiply Accumulate (word by halfword)

1. specifies which half of the source register Rm is used as the second multiply operand.

If Y is T , then the top halfword, bits [31:16] of Rm is used.

If Y is B , then the bottom halfword, bits [15:0] of Rm is used.

$cond$ is an optional condition code, see [Conditional execution](#).

Rd is the destination register. If Rd is omitted, the destination register is Rn .

Rn, Rm are registers holding the values to be multiplied.

Ra is a register holding the value to be added or subtracted from.

Operation

The SMALBB, SMLABT, SMLATB, SMLATT instructions:

- Multiplies the specified signed halfword, top or bottom, values from Rn and Rm .
- Adds the value in Ra to the resulting 3bit product.
- Writes the result of the multiplication and addition in Rd .

The non-specified halfwords of the source registers are ignored.

The SMLAWB and SMLAWT instructions:

- Multiply the 3bit signed values in Rn with:

—The top signed halfword of Rm , T instruction suffix.

—The bottom signed halfword of Rm , B instruction suffix.

- Add the 3bit signed value in Ra to the top 32 bits of the 48-bit product
- Writes the result of the multiplication and addition in Rd .

The bottom 16 bits of the 48-bit product are ignored.

If overflow occurs during the addition of the accumulate value, the instruction sets the Q flag in the APSR. No overflow can occur during the multiplication.

Restrictions

In these instructions, do not use SP and do not use PC.

Condition flags

If an overflow is detected, the Q flag is set.

Examples

| | | |
|---------------|-----------------|---|
| SMLABB | R5, R6, R4, R1 | ; Multiplies bottom halfwords of R6 and R4, adds R1 and writes to R5 |
| SMLATB | R5, R6, R4, R1 | ; Multiplies top halfword of R6 with bottom halfword of R4, adds R1 and writes to R5 |
| SMLATT | R5, R6, R4, R1 | ; Multiplies top halfwords of R6 and R4, adds R1 and writes the sum to R5 |
| SMLABT | R5, R6, R4, R1 | ; Multiplies bottom halfword of R6 with top halfword of R4, adds R1 and writes to R5 |
| SMLABT | R4, R3, R2 | ; Multiplies bottom halfword of R4 with top halfword of R3, adds R2 and writes to R4 |
| SMLAWB | R10, R2, R5, R3 | ; Multiplies R2 with bottom halfword of R5, adds R3 to the result and writes top 3bits to R10 |
| SMLAWT | R10, R2, R1, R5 | ; Multiplies R2 with top halfword of R1, adds R5 ; and writes top 3bits to R10. |

SML

Signed Multiply Accumulate Long Dual

Syntax op{X}{cond} Rd, Rn, Rm, Ra; where:
op is one of:

SMLAD Signed Multiply Accumulate Dual

SMLADX Signed Multiply Accumulate Dual Reverse

X specifies which halfword of the source register Rn is used as the multiply operand.

If X is omitted, the multiplications are bottom × bottom and top × top. If X is present, the multiplications are bottom × top and top × bottom.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register holding the values to be multiplied.

Rm the second operand register.

Ra is the accumulate value.

Operation

The SMLAD and SMLADX instructions regard the two operands as four halfword 16-bit values. The SMLAD and SMLADX instructions:

- If X is not present, multiply the top signed halfword value in Rn with the top signed halfword of Rm and the bottom signed halfword values in Rn with the bottom signed halfword of Rm.
- Or if X is present, multiply the top signed halfword value in Rn with the bottom signed halfword of Rm and the bottom signed halfword values in Rn with the top signed halfword of Rm.
- Add both multiplication results to the signed 3bit value in Ra.
- Writes the 3bit signed result of the multiplication and addition to Rd.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|---------|-----------------|---|
| SMLAD | R10, R2, R1, R5 | ; Multiplies two halfword values in R2 with corresponding halfwords in R1, adds R5 and writes to R10 |
| SMLALDX | R0, R2, R4, R6 | ; Multiplies top halfword of R2 with bottom halfword of R4, multiplies bottom halfword of R2 with top halfword of R4, adds R6 and writes to R0. |

SMLAL and SMLALD

Signed Multiply Accumulate Long, Signed Multiply Accumulate Long (halfwords) and Signed Multiply Accumulate Long Dual.

Syntax

$op\{cond\} RdLo, RdHi, Rn, Rm op\{XY\}\{cond\} RdLo, RdHi, Rn, Rm op\{X\}\{cond\} RdLo, RdHi, Rn, Rm$ where:

| | |
|--------------|--|
| op | is one of: |
| SMLAL | Signed Multiply Accumulate Long |
| SMLAL | Signed Multiply Accumulate Long (halfwords, X and Y) X and Y specify which halfword of the source registers Rn and Rm are used as the first and second multiply operand: If X is B, then the bottom halfword, bits [15:0], of Rn is used. If X is T, then the top halfword, bits [31:16], of Rn is used. If Y is B, then the bottom halfword, bits [15:0], of Rm is used. If Y is T, then the top halfword, bits [31:16], of Rm is used. |
| SMLALD | Signed Multiply Accumulate Long Dual |
| SMLALDX | Signed Multiply Accumulate Long Dual Reversed If the X is omitted, the multiplications are bottom \times bottom and top \times top. If X is present, the multiplications are bottom \times top and top \times bottom. |
| $cond$ | is an optional condition code, see Conditional execution . |
| $RdHi, RdLo$ | are the destination registers. $RdLo$ is the lower 32 bits and $RdHi$ is the upper 32 bits of the 64-bit integer. For SMLAL, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLALD and SMLALDX, they also hold the accumulating value. |
| Rn, Rm | are registers holding the first and second operands. |

Operation

The SMLAL instruction:

- Multiplies the two's complement signed word values from Rn and Rm .
- Adds the 64-bit value in $RdLo$ and $RdHi$ to the resulting 64-bit product.
- Writes the 64-bit result of the multiplication and addition in $RdLo$ and $RdHi$.

The SMLALBB, SMLALBT, SMLALTB and SMLALTT instructions:

- Multiplies the specified signed halfword, Top or Bottom, values from Rn and Rm .
- Adds the resulting sign-extended 3bit product to the 64-bit value in $RdLo$ and $RdHi$.
- Writes the 64-bit result of the multiplication and addition in $RdLo$ and $RdHi$.

The non-specified halfwords of the source registers are ignored.

The SMLALD and SMLALDX instructions interpret the values from Rn and Rm as four halfword two's complement signed 16-bit integers. These instructions:

- if X is not present, multiply the top signed halfword value of Rn with the top signed halfword of Rm and the bottom signed halfword values of Rn with the bottom signed halfword of Rm .

- Or if *X* is present, multiply the top signed halfword value of *Rn* with the bottom signed halfword of *Rm* and the bottom signed halfword values of *Rn* with the top signed halfword of *Rm*.
- Add the two multiplication results to the signed 64-bit value in *RdLo* and *RdHi* to create the resulting 64-bit product.
- Write the 64-bit product in *RdLo* and *RdHi*.

Restrictions

In these instructions:

- do not use SP and do not use PC.
- *RdHi* and *RdLo* must be different registers.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|----------------|----------------|--|
| SMLAL | R4, R5, R3, R8 | ; Multiplies R3 and R8, adds R5:R4 and writes to R5:R4 |
| SMLALBT | R2, R1, R6, R7 | ; Multiplies bottom halfword of R6 with top halfword of R7, sign extends to 3bit, adds R1:R2 and writes to R1:R2 |
| SMLALTB | R2, R1, R6, R7 | ; Multiplies top halfword of R6 with bottom halfword of R7, sign extends to 3bit, adds R1:R2 and writes to R1:R2 |
| SMLALD | R6, R8, R5, R1 | ; Multiplies top halfwords in R5 and R1 and bottom halfwords of R5 and R1, adds R8:R6 and writes to R8:R6 |
| SMLALDX | R6, R8, R5, R1 | ; Multiplies top halfword in R5 with bottom halfword of R1, and bottom halfword of R5 with top halfword of R1, adds R8:R6 and writes to R8:R6. |

SMLSD and SMLS LD

Signed Multiply Subtract Dual and Signed Multiply Subtract Long Dual

Syntax

op{X}{cond} Rd, Rn, Rm, Ra where:

op is one of:

SMLSD Signed Multiply Subtract Dual. SMLSDX Signed Multiply Subtract Dual Reversed SMLS LD Signed Multiply Subtract Long Dual.

SMLS LDX Signed Multiply Subtract Long Dual Reversed.

If *X* is present, the multiplications are bottom × top and top × bottom.

If the *X* is omitted, the multiplications are bottom × bottom and top × top.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn, *Rm* are registers holding the first and second operands.

Ra is the register holding the accumulate value.

Operation

The SMLSD instruction interprets the values from the first and second operands as four signed halfwords. This instruction:

- Optionally rotates the halfwords of the second operand.
- Performs two signed 16 × 16-bit halfword multiplications.
- Subtracts the result of the upper halfword multiplication from the result of the lower halfword multiplication.
- Adds the signed accumulate value to the result of the subtraction.
- Writes the result of the addition to the destination register.

The SMLSLD instruction interprets the values from *Rn* and *Rm* as four signed halfwords. This instruction:

- Optionally rotates the halfwords of the second operand.
- Performs two signed 16 × 16-bit halfword multiplications.
- Subtracts the result of the upper halfword multiplication from the result of the lower halfword multiplication.
- Adds the 64-bit value in *RdHi* and *RdLo* to the result of the subtraction.
- Writes the 64-bit result of the addition to the *RdHi* and *RdLo*.

Restrictions

In these instructions:

- Do not use SP and do not use PC.

Condition flags

This instruction sets the Q flag if the accumulate operation overflows. Overflow cannot occur during the multiplications or subtraction.

For the Thumb instruction set, these instructions do not affect the condition code flags.

Examples

| | | |
|----------------|-----------------------|---|
| SMLSD | <i>R0, R4, R5, R6</i> | ; Multiplies bottom halfword of R4 with bottom halfword of R5, multiplies top halfword of R4 with top halfword of R5, subtracts second from first, adds R6, writes to R0 |
| SMLSDX | <i>R1, R3, R2, R0</i> | ; Multiplies bottom halfword of R3 with top halfword of R2, multiplies top halfword of R3 with bottom halfword of R2, subtracts second from first, adds R0, writes to R1 |
| SMLSLD | <i>R3, R6, R2, R7</i> | ; Multiplies bottom halfword of R6 with bottom halfword of R2, multiplies top halfword of R6 with top halfword of R2, subtracts second from first, adds R6:R3, writes to R6:R3 |
| SMLSLDX | <i>R3, R6, R2, R7</i> | ; Multiplies bottom halfword of R6 with top halfword of R2, multiplies top halfword of R6 with bottom halfword of R2, subtracts second from ; first, adds R6:R3, writes to R6:R3. |

SMMLA and SMMLS

Signed Most Significant Word Multiply Accumulate and Signed Most Significant Word Multiply Subtract

Syntax

op{R}{cond} Rd, Rn, Rm, Ra where:

op is one of:

SMMLA Signed Most Significant Word Multiply Accumulate.

SMMLS Signed Most Significant Word Multiply Subtract.

R is a rounding error flag. If *R* is specified, the result is rounded instead of being truncated. In this case the constant 0x80000000 is added to the product before the high word is extracted.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn, *Rm* are registers holding the first and second multiply operands.

Ra is the register holding the accumulate value.

Operation

The SMMLA instruction interprets the values from *Rn* and *Rm* as signed 3bit words.

The SMMLA instruction:

- Multiplies the values in *Rn* and *Rm*.
- Optionally rounds the result by adding 0x80000000.
- Extracts the most significant 32 bits of the result.

- Adds the value of Ra to the signed extracted value.
- Writes the result of the addition in Rd .

The SMMLS instruction interprets the values from Rn and Rm as signed 3bit words.
The SMMLS instruction:

- Multiplies the values in Rn and Rm .
- Optionally rounds the result by adding 0x80000000.
- Extracts the most significant 32 bits of the result.
- Subtracts the extracted value of the result from the value in Ra .
- Writes the result of the subtraction in Rd .

Restrictions

In these instructions:

- Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|---------------|----------------|---|
| SMMLA | R0, R4, R5, R6 | ; Multiplies R4 and R5, extracts top 32 bits, adds R6, truncates and writes to R0 |
| SMMLAR | R6, R2, R1, R4 | ; Multiplies R2 and R1, extracts top 32 bits, adds R4, rounds and writes to R6 |
| SMMLSR | R3, R6, R2, R7 | ; Multiplies R6 and R2, extracts top 32 bits, subtracts R7, rounds and writes to R3 |
| SMMLS | R4, R5, R3, R8 | ; Multiplies R5 and R3, extracts top 32 bits, ; subtracts R8, truncates and writes to R4. |

SMMUL

Signed Most Significant Word Multiply

Syntax

op{R}{cond} Rd, Rn, Rm where:

op is one of:

SMMUL Signed Most Significant Word Multiply

R is a rounding error flag. If R is specified, the result is rounded instead of being truncated. In this case the constant 0x80000000 is added to the product before the high word is extracted.

$cond$ is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn , Rm are registers holding the first and second operands.

Operation

The SMMUL instruction interprets the values from Rn and Rm as two's complement 3bit signed integers. The SMMUL instruction:

- Multiplies the values from Rn and Rm .
- Optionally rounds the result, otherwise truncates the result.
- Writes the most significant 32 bits of the result in Rd .

Restrictions

In this instruction:

- do not use SP and do not use PC.

Condition flags

This instruction does not affect the condition code flags.

Examples

| | | |
|---------------|------------|--|
| SMULL | R0, R4, R5 | ; Multiplies R4 and R5, truncates top 32 bits and writes to R0 |
| SMULLR | R6, R2 | ; Multiplies R6 and R2, rounds the top 32 bits and writes to R6. |

SMUAD and SMUSD

Signed Dual Multiply Add and Signed Dual Multiply Subtract

Syntax

op{X}{cond} Rd, Rn, Rm where:

op is one of:

SMUAD Signed Dual Multiply Add.

SMUADX Signed Dual Multiply Add Reversed.

SMUSD Signed Dual Multiply Subtract.

SMUSDX Signed Dual Multiply Subtract Reversed.

If *X* is present, the multiplications are bottom × top and top × bottom.

If the *X* is omitted, the multiplications are bottom × bottom and top × top.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn, *Rm* are registers holding the first and the second operands.

Operation

The SMUAD instruction interprets the values from the first and second operands as two signed halfwords in each operand. This instruction:

- Optionally rotates the halfwords of the second operand.
- Performs two signed 16×16 -bit multiplications.
- Adds the two multiplication results together.
- Writes the result of the addition to the destination register.

The SMUSD instruction interprets the values from the first and second operands as two's complement signed integers. This instruction:

- Optionally rotates the halfwords of the second operand.
- Performs two signed 16×16 -bit multiplications.
- Subtracts the result of the top halfword multiplication from the result of the bottom halfword multiplication.
- Writes the result of the subtraction to the destination register.

Restrictions

In these instructions:

- Do not use SP and do not use PC.

Condition flags

Sets the Q flag if the addition overflows. The multiplications cannot overflow.

Examples

| | | |
|-----------------|------------|---|
| SM UAD | R0, R4, R5 | ;Multiplies bottom halfword of R4 with the bottom halfword of R5, adds multiplication of top halfword of R4 with top halfword of R5, write to R0 |
| SM UAD X | R3, R7, R4 | ; Multiplies bottom halfword of R7 with top halfword of R4, adds multiplication of top halfword of R7 with bottom halfword of R4, writes to R3 |
| SM USD | R3, R6, R2 | ; Multiplies bottom halfword of R4 with bottom halfword of R6, subtracts multiplication of top halfword of R6 with top halfword of R3, writes to R3 |

| | | |
|-------------------------|------------|--|
| SM USD X | R4, R5, R3 | ; Multiplies bottom halfword of R5 with top halfword of ; R3, subtracts multiplication of top halfword of R5 with bottom halfword of R3, writes to R4. |
|-------------------------|------------|--|

SMUL and SMULW

Signed Multiply (halfwords) and Signed Multiply (word by halfword)

Syntax

$op\{XY\}\{cond\} Rd, Rn, Rm$ $op\{Y\}\{cond\} Rd, Rn, Rm$ For SMULXY only:

op is one of:

SMUL $\{XY\}$ Signed Multiply (halfwords)

1. and Y specify which halfword of the source registers Rn and Rm is used as the first and second multiply operand.

If X is B, then the bottom halfword, bits [15:0] of Rn is used.

If X is T, then the top halfword, bits [31:16] of Rn is used. If Y is B, then the bottom halfword, bits [15:0], of Rm is used.

If Y is T, then the top halfword, bits [31:16], of Rm is used.

SMULW $\{Y\}$ Signed Multiply (word by halfword)

1. specifies which halfword of the source register Rm is used as the second multiply operand.

If Y is B, then the bottom halfword (bits [15:0]) of Rm is used. If Y is T, then the top halfword (bits [31:16]) of Rm is used.

$cond$ is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn, Rm are registers holding the first and second operands.

Operation

The SMULBB, SMULTB, SMULBT and SMULTT instructions interprets the values from Rn and Rm as four signed 16-bit integers. These instructions:

- Multiplies the specified signed halfword, Top or Bottom, values from Rn and Rm .
- Writes the 3bit result of the multiplication in Rd .

The SMULWT and SMULWB instructions interprets the values from Rn as a 3bit signed integer and Rm as two halfword 16-bit signed integers. These instructions:

- Multiplies the first operand and the top, T suffix, or the bottom, B suffix, halfword of the second operand.
- Writes the signed most significant 32 bits of the 48-bit result in the destination register.

Restrictions

In these instructions:

- Do not use SP and do not use PC.
- $RdHi$ and $RdLo$ must be different registers.

Examples

| | | |
|---------------|------------|--|
| SMULBT | R0, R4, R5 | ; Multiplies the bottom halfword of R4 with the top halfword of R5, multiplies results and writes to R0 |
| SMULBB | R0, R4, R5 | ; Multiplies the bottom halfword of R4 with the bottom halfword of R5, multiplies results and writes to R0 |
| SMULTT | R0, R4, R5 | ; Multiplies the top halfword of R4 with the top halfword of R5, multiplies results and writes to R0 |
| SMULTB | R0, R4, R5 | ; Multiplies the top halfword of R4 with the bottom halfword of R5, multiplies results and writes to R0 |

| | | |
|---------------|------------|--|
| SMULWT | R4, R5, R3 | ; Multiplies R5 with the top halfword of R3, extracts top 32 bits and writes to R4 |
| SMULWB | R4, R5, R3 | ; Multiplies R5 with the bottom halfword of R3, extracts top 32 bits and writes to R4. |

UMULL, UMLAL, SMULL, and SMLAL

Signed and Unsigned Long Multiply, with optional Accumulate, using 3bit operands and producing a 64-bit result.

Syntax

op{cond} RdLo, RdHi, Rn, Rm where:

| | |
|-------------------|--|
| <i>op</i> | is one of: UMULL Unsigned Long Multiply. UMLAL Unsigned Long Multiply, with Accumulate. SMULL Signed Long Multiply. SMLAL Signed Long Multiply, with Accumulate. |
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>RdHi, RdLo</i> | are the destination registers. For UMLAL and SMLAL they also hold the accumulating value. |
| <i>Rn, Rm</i> | are registers holding the operands. |

Operation

The UMULL instruction interprets the values from *Rn* and *Rm* as unsigned integers. It multiplies these integers and places the least significant 32 bits of the result in *RdLo*, and the most significant 32 bits of the result in *RdHi*.

The UMLAL instruction interprets the values from *Rn* and *Rm* as unsigned integers. It multiplies these integers, adds the 64-bit result to the 64-bit unsigned integer contained in *RdHi* and *RdLo*, and writes the result back to *RdHi* and *RdLo*.

The SMULL instruction interprets the values from *Rn* and *Rm* as two's complement signed integers. It multiplies these integers and places the least significant 32 bits of the result in *RdLo*, and the most significant 32 bits of the result in *RdHi*.

The SMLAL instruction interprets the values from *Rn* and *Rm* as two's complement signed integers. It multiplies these integers, adds the 64-bit result to the 64-bit signed integer contained in *RdHi* and *RdLo*, and writes the result back to *RdHi* and *RdLo*.

Restrictions

In these instructions:

- do not use SP and do not use PC
- *RdHi* and *RdLo* must be different registers.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | |
|-------|--|
| UMULL | R0, R4, R5, R6; Unsigned (R4,R0) = R5 x R6 |
| SMLAL | R4, R5, R3, R8; Signed (R5,R4) = (R5,R4) + R3 x R8 |

SDIV and UDIV

Signed Divide and Unsigned Divide.

Syntax

SDIV{cond} {Rd} Rn, Rm UDIV{cond} {Rd} Rn, Rm where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn is the register holding the value to be divided.

Rm is a register holding the divisor.

Operation

SDIV performs a signed integer division of the value in Rn by the value in Rm .

UDIV performs an unsigned integer division of the value in Rn by the value in Rm .

For both instructions, if the value in Rn is not divisible by the value in Rm , the result is rounded towards zero.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|------|--------------|-------------------------------|
| SDIV | $R0, R2, R4$ | Signed divide, $R0 = R2/R4$ |
| UDIV | $R8, R8, R1$ | Unsigned divide, $R8 = R8/R1$ |

1.3.6 Saturating instructions

Table 33 shows the saturating instructions:

Mnemonic

| Mnemonic | Brief description | See |
|----------|--|---|
| SSAT | Signed Saturate | SSAT and USAT |
| SSAT16 | Signed Saturate Halfword | SSAT16 and USAT |
| USAT | Unsigned Saturate | SSAT and USAT |
| USAT16 | Unsigned Saturate Halfword | SSAT16 and USAT16 |
| QADD | Saturating Add | QADD and QSUB |
| QSUB | Saturating Subtract | QADD and QSUB |
| QSUB16 | Saturating Subtract 16 | QADD and QSUB |
| QASX | Saturating Add and Subtract with Exchange | QASX and QSAX |
| QSAX | Saturating Subtract and Add with Exchange | QASX and QSAX |
| QDADD | Saturating Double and Add | QDADD and QDSUB |
| QDSUB | Saturating Double and Subtract | QDADD and QDSUB |
| UQADD16 | Unsigned Saturating Add 16 | UQADD and UQSUB |
| UQADD8 | Unsigned Saturating Add 8 | UQADD and UQSUB |
| UQASX | Unsigned Saturating Add and Subtract with Exchange | UQASX and UQSAX |
| UQSAX | Unsigned Saturating Subtract and Add with Exchange | UQASX and UQSAX |
| UQSUB16 | Unsigned Saturating Subtract 16 | UQADD and UQSUB |
| UQSUB8 | Unsigned Saturating Subtract 8 | UQADD and UQSUB |

35 . Table 33 Saturating instructions

For signed n -bit saturation, this means that:

- if the value to be saturated is less than -2^{n-1} , the result returned is -2^{n-1} • if the value to be saturated is greater than $2^{n-1}-1$, the result returned is $2^{n-1}-1$
- otherwise, the result returned is the same as the value to be saturated.

For unsigned n -bit saturation, this means that:

- if the value to be saturated is less than 0, the result returned is 0
- if the value to be saturated is greater than 2^n-1 , the result returned is 2^n-1
- otherwise, the result returned is the same as the value to be saturated.

If the returned result is different from the value to be saturated, it is called *saturation*. If saturation occurs, the instruction sets the Q flag to 1 in the APSR. Otherwise, it leaves the Q flag unchanged. To clear the Q flag to 0, you must use the MSR instruction, see [MSR](#).

To read the state of the Q flag, use the MRS instruction

SSAT and USAT

Signed Saturate and Unsigned Saturate to any bit position, with optional shift before saturating.

Syntax

op{cond} Rd, #n, Rm {, shift #s} where:

| | |
|-----------------|--|
| <i>op</i> | is one of: SSAT Saturates a signed value to a signed range. USAT Saturates a signed value to an unsigned range. |
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>Rd</i> | is the destination register. |
| <i>n</i> | specifies the bit position to saturate to: <ul style="list-style-type: none"> • <i>n</i> ranges from 1 to 32 for SSAT • <i>n</i> ranges from 0 to 31 for USAT. |
| <i>Rm</i> | is the register containing the value to saturate. |
| <i>shift #s</i> | is an optional shift applied to <i>Rm</i> before saturating. It must be one of the following: ASR # <i>s</i> where <i>s</i> is in the range 1 to 31 LSL # <i>s</i> where <i>s</i> is in the range 0 to 31. |

Operation

These instructions saturate to a signed or unsigned n -bit value.

The SSAT instruction applies the specified shift, then saturates to the signed range $-2^{n-1} \leq x \leq 2^{n-1}-1$.

The USAT instruction applies the specified shift, then saturates to the unsigned range $0 \leq x \leq 2^n-1$.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.
If saturation occurs, these instructions set the Q flag to 1.

Examples

| | | |
|---------------|----------------------|--|
| SSAT | R7, #16, R7, LSL #4; | Logical shift left value in R7 by 4, then ; saturate it as a signed 16-bit value and ; write it back to R7 |
| USATNE | R0, #7, R5 | ; Conditionally saturate value in R5 as an ; unsigned 7 bit value and write it to R0. |

SSAT16 and USAT16

Signed Saturate and Unsigned Saturate to any bit position for two halfwords.

Syntax

op{cond} Rd, #n, Rm where:

op is one of:

SSAT16 Saturates a signed halfword value to a signed range.

USAT16 Saturates a signed halfword value to an unsigned range.

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

n specifies the bit position to saturate to:

- *n* ranges from 1 to 16 for SSAT.
- *n* ranges from 0 to 15 for USAT.

Rm is the register containing the value to saturate.

Operation

The SSAT16 instruction:

1. Saturates two signed 16-bit halfword values of the register with the value to saturate from selected by the bit position in *n*.
2. Writes the results as two signed 16-bit halfwords to the destination register.

The USAT16 instruction:

1. Saturates two unsigned 16-bit halfword values of the register with the value to saturate from selected by the bit position in *n*.
2. Writes the results as two unsigned halfwords in the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

If saturation occurs, these instructions set the Q flag to 1.

Examples

| | | |
|-----------------|-------------|---|
| SSAT16 | R7, #9, R2 | ; Saturates the top and bottom highwords of R2 ; as 9-bit values, writes to corresponding halfword ; of R7 |
| USAT16NE | R0, #13, R5 | ; Conditionally saturates the top and bottom ; halfwords of R5 as 13-bit values, writes to ; corresponding halfword of R0. |

QADD and QSUB

Saturating Add and Saturating Subtract, signed.

Syntax

`op{cond} {Rd}, Rn, Rm op{cond} {Rd}, Rn, Rm` where:
`op` is one of:

| | |
|---------------|--|
| QADD | Saturating 3bit add. |
| QADD8 | Saturating four 8-bit integer additions. |
| QADD16 | Saturating two 16-bit integer additions. |
| QSUB | Saturating 3bit subtraction. |
| QSUB8 | Saturating four 8-bit integer subtraction. |
| QSUB16 | Saturating two 16-bit integer subtraction. |

`cond` is an optional condition code, see [Conditional execution](#).

`Rd` is the destination register.

`Rn, Rm` are registers holding the first and second operands.

Operation

These instructions add or subtract two, four or eight values from the first and second operands and then writes a signed saturated value in the destination register.

The QADD and QSUB instructions apply the specified add or subtract, and then saturate the result to the signed range $-2^{n-1} \leq x \leq 2^{n-1}-1$, where x is given by the number of bits applied in the instruction, 32, 16 or 8.

If the returned result is different from the value to be saturated, it is called *saturation*. If saturation occurs, the QADD and QSUB instructions set the Q flag to 1 in the APSR. Otherwise, it leaves the Q flag unchanged. The 8-bit and 16-bit QADD and QSUB instructions always leave the Q flag unchanged.

To clear the Q flag to 0, you must use the MSR instruction, see [MSR](#).

To read the state of the Q flag, use the MRS instruction, see [MRS](#).

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

If saturation occurs, these instructions set the Q flag to 1.

Examples

| | |
|---------------|--|
| QADD16 | R7, R4, R2 ; Adds halfwords of R4 with corresponding halfword of ; R2, saturates to 16 bits and writes to corresponding ; halfword of R7 |
| QADD8 | R3, R1, R6 ; Adds bytes of R1 to the corresponding bytes of R6, ; saturates to 8 bits and writes to corresponding byte of ; R3 |
| QSUB16 | R4, R2, R3 ; Subtracts halfwords of R3 from corresponding halfword ; of R2, saturates to 16 bits, writes to corresponding ; halfword of R4 |

| | | |
|--------------|------------|---|
| QSUB8 | R4, R2, R5 | ; Subtracts bytes of R5 from the corresponding byte in ; R2, saturates to 8 bits, writes to corresponding byte of ; R4. |
|--------------|------------|---|

QASX and QSAX

Saturating Add and Subtract with Exchange and Saturating Subtract and Add with Exchange, signed.

Syntax

op{cond} {Rd}, Rm, Rn where:

| | |
|---------------|---|
| op | is one of: QASX Add and Subtract with Exchange and Saturate. QSAX Subtract and Add with Exchange and Saturate. |
| cond | is an optional condition code, see Conditional execution . |
| Rd | is the destination register. |
| Rn, Rm | are registers holding the first and second operands. |

Operation

The QASX instruction:

1. Adds the top halfword of the source operand with the bottom halfword of the second operand.
2. Subtracts the top halfword of the second operand from the bottom highword of the first operand.
3. Saturates the result of the subtraction and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where x equals 16, to the bottom halfword of the destination register.
4. Saturates the results of the sum and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where x equals 16, to the top halfword of the destination register.

The QSAX instruction:

1. Subtracts the bottom halfword of the second operand from the top highword of the first operand.
2. Adds the bottom halfword of the source operand with the top halfword of the second operand.
3. Saturates the results of the sum and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where x equals 16, to the bottom halfword of the destination register.

$-2^{15} \leq x \leq 2^{15} - 1$, where x equals 16, to the top halfword of the destination register.

1. Saturates the result of the subtraction and writes a 16-bit signed integer in the range $-2^{15} \leq x \leq 2^{15} - 1$, where x equals 16, to the top halfword of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|-------------|------------|--|
| QASX | R7, R4, R2 | ; Adds top halfword of R4 to bottom halfword of R2, ; saturates to 16 bits, writes to top halfword of R7 ; Subtracts top highword of R2 from bottom halfword of ; R4, saturates to 16 bits and writes to bottom halfword ; of R7 |
| QSAX | R0, R3, R5 | ; Subtracts bottom halfword of R5 from top halfword of ; R3, saturates to 16 bits, writes to top halfword of R0 ; Adds bottom halfword of R3 to top halfword of R5, ; saturates to 16 bits, writes to bottom halfword of R0. |

QDADD and QDSUB

Saturating Double and Add and Saturating Double and Subtract, signed.

Syntax

op{cond} {Rd}, Rm, Rn where:

| | |
|---------------|---|
| op | is one of: QDADD Saturating Double and Add. QDSUB Saturating Double and Subtract. |
| cond | is an optional condition code, see Conditional execution . |
| Rd | is the destination register. |
| Rm, Rn | are registers holding the first and second operands. |

Operation

The QDADD instruction:

- Doubles the second operand value.
- Adds the result of the doubling to the signed saturated value in the first operand.
- Writes the result to the destination register.

The QDSUB instruction:

- Doubles the second operand value.
- Subtracts the doubled value from the signed saturated value in the first operand.
- Writes the result to the destination register.

Both the doubling and the addition or subtraction have their results saturated to the 3bit signed integer range $-2^{31} \leq x \leq 2^{31} - 1$. If saturation occurs in either operation, it sets the Q flag in the APSR.

Restrictions

Do not use SP and do not use PC.

Condition flags

If saturation occurs, these instructions set the Q flag to 1.

Examples

| | | |
|--------------|------------|---|
| QADDR | R7, R4, R2 | ; Doubles and saturates R4 to 32 bits, adds R2, ; saturates to 32 bits, writes to R7 |
| QDSUB | R0, R3, R5 | ; Subtracts R3 doubled and saturated to 32 bits ; from R5, saturates to 32 bits, writes to R0. |

UQASX and UQSAX

Saturating Add and Subtract with Exchange and Saturating Subtract and Add with Exchange, unsigned.

Syntax

op{cond} {Rd}, Rm, Rn where:

| | |
|-------------------------|---|
| type | is one of: UQASX Add and Subtract with Exchange and Saturate. UQSAX Subtract and Add with Exchange and Saturate. |
| cond | is an optional condition code, see <i>Conditional execution</i> . |
| Rd | is the destination register. |
| Rn, Rm | are registers holding the first and second operands. |

Operation

The UQASX instruction:

1. Adds the bottom halfword of the source operand with the top halfword of the second operand.
2. Subtracts the bottom halfword of the second operand from the top highword of the first operand.
3. Saturates the results of the sum and writes a 16-bit unsigned integer in the range $0 \leq x \leq 2^{16} - 1$, where x equals 16, to the top halfword of the destination register.
4. Saturates the result of the subtraction and writes a 16-bit unsigned integer in the range $0 \leq x \leq 2^{16} - 1$, where x equals 16, to the bottom halfword of the destination register.

The UQSAX instruction:

1. Subtracts the bottom halfword of the second operand from the top highword of the first operand.
2. Adds the bottom halfword of the first operand with the top halfword of the second operand.
3. Saturates the result of the subtraction and writes a 16-bit unsigned integer in the range $0 \leq x \leq 2^{16} - 1$, where x equals 16, to the top halfword of the destination register.
4. Saturates the results of the addition and writes a 16-bit unsigned integer in the range $0 \leq x \leq 2^{16} - 1$, where x equals 16, to the bottom halfword of the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|--------------|------------|--|
| UQASX | R7, R4, R2 | ; Adds top halfword of R4 with bottom halfword of R2, ; saturates to 16 bits, writes to top halfword of R7 ; Subtracts top halfword of R2 from bottom halfword of ; R4, saturates to 16 bits, writes to bottom halfword of R7 |
| UQSAX | R0, R3, R5 | ; Subtracts bottom halfword of R5 from top halfword of R3, ; saturates to 16 bits, writes to top halfword of R0 ; Adds bottom halfword of R4 to top halfword of R5 ; saturates to 16 bits, writes to bottom halfword of R0. |

UQADD and UQSUB

Saturating Add and Saturating Subtract Unsigned.

Syntax

$op\{cond\} \{Rd\}, Rn, Rm op\{cond\} \{Rd\}, Rn, Rm$ where:
 op is one of:

UQADD8 Saturating four unsigned 8-bit integer additions.
UQADD16 Saturating two unsigned 16-bit integer additions.
UDSUB8 Saturating four unsigned 8-bit integer subtractions.
UQSUB16 Saturating two unsigned 16-bit integer subtractions.
 $cond$ is an optional condition code, see [Conditional execution](#).
 Rd is the destination register.
 Rn, Rm are registers holding the first and second operands.

Operation

These instructions add or subtract two or four values and then writes an unsigned saturated value in the destination register.

The UQADD16 instruction:

- Adds the respective top and bottom halfwords of the first and second operands.
- Saturates the result of the additions for each halfword in the destination register to the unsigned range $0 \leq x \leq 2^{16}-1$, where x is 16.

The UQADD8 instruction:

- Adds each respective byte of the first and second operands.
- Saturates the result of the addition for each byte in the destination register to the unsigned range $0 \leq x \leq 2^8-1$, where x is 8.

The UQSUB16 instruction:

- Subtracts both halfwords of the second operand from the respective halfwords of the first operand.
- Saturates the result of the differences in the destination register to the unsigned range $0 \leq x \leq 2^{16}-1$, where x is 16.

The UQSUB8 instructions:

- Subtracts the respective bytes of the second operand from the respective bytes of the first operand.
- Saturates the results of the differences for each byte in the destination register to the unsigned range $0 \leq x \leq 2^8-1$, where x is 8.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the condition code flags.

Examples

| | | |
|----------------|------------|---|
| UQADD16 | R7, R4, R2 | Adds halfwords in R4 to corresponding halfword in R2, ; saturates to 16 bits, writes to corresponding halfword ; of R7 |
| UQADD8 | R4, R2, R5 | ; Adds bytes of R2 to corresponding byte of R5, saturates ; to 8 bits, writes to corresponding bytes of R4 |
| UQSUB16 | R6, R3, R0 | Subtracts halfwords in R0 from corresponding halfword ; in R3, saturates to 16 bits, writes to corresponding ; halfword in R6 |
| UQSUB8 | R1, R5, R6 | ; Subtracts bytes in R6 from corresponding byte of R5, ; saturates to 8 bits, writes to corresponding byte of R1. |

1.3.7 Packing and unpacking instructions

Table 34 shows the instructions that operate on packing and unpacking data:

| Mnemonic | Brief description | See |
|----------|---------------------------------------|---------------------------------|
| PKH | Pack Halfword | PKHBT and PKHTB |
| SXTAB | Extend 8 bits to 32 and add | SXTA and UXTA |
| SXTAB16 | Dual extend 8 bits to 16 and add | SXTA and UXTA |
| SXTAH | Extend 16 bits to 32 and add | SXTA and UXTA |
| SXTB | Sign extend a byte | SXT and UXT |
| SXTB16 | Dual extend 8 bits to 16 and add | SXT and UXT |
| SXTH | Sign extend a halfword | SXT and UXT |
| UXTAB | Extend 8 bits to 32 and add | SXTA and UXTA |
| UXTAB16 | Dual extend 8 bits to 16 and add | SXTA and UXTA |
| UXTAH | Extend 16 bits to 32 and add | SXTA and UXTA |
| UXTB | Zero extend a byte | SXT and UXT |
| UXTB16 | Dual zero extend 8 bits to 16 and add | SXT and UXT |
| UXTH | Zero extend a halfword | SXT and UXT |

36 .Packing and Unpacking Instructions

PKHBT and PKHTB

Pack Halfword

Syntax

$op\{cond\} \{Rd\}, Rn, Rm \{, LSL \#imm\} op\{cond\} \{Rd\}, Rn, Rm \{, ASR \#imm\}$ where:

op is one of:

PKHBT Pack Halfword, bottom and top with shift.

PKHTB Pack Halfword, top and bottom with shift.

$cond$ is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register.

Rm is the second operand register holding the value to be optionally shifted.

imm is the shift length. The type of shift length depends on the instruction:

For PKHBT:

LSL a left shift with a shift length from 1 to 31, 0 means no shift.

For PKHTB:

ASR an arithmetic shift right with a shift length from 1 to 32, a shift of 3bits is encoded as 0b00000.

Operation

The PKHBT instruction:

1. Writes the value of the bottom halfword of the first operand to the bottom halfword of the destination register.
2. If shifted, the shifted value of the second operand is written to the top halfword of the destination register.

The PKHTB instruction:

1. Writes the value of the top halfword of the first operand to the top halfword of the destination register.
2. If shifted, the shifted value of the second operand is written to the bottom halfword of the destination register.

Restrictions

Rd must not be SP and must not be PC.

Condition flags

This instruction does not change the flags.

Examples

| | |
|--------------|--|
| PKHBT | R3, R4, R5 LSL #0 ; Writes bottom halfword of R4 to bottom halfword of ; R3, writes top halfword of R5, unshifted, to top ; halfword of R3 |
| PKHTB | R4, R0, R2 ASR #1; Writes R2 shifted right by 1 bit to bottom halfword ; of R4, and writes top halfword of R0 to top ; halfword of R4. |

SXT and UXT

Sign extend and Zero extend.

Syntax

$op\{cond\} \{Rd\}, Rm \{, ROR \#n\} op\{cond\} \{Rd\}, Rm \{, ROR \#n\}$ where:

op is one of:

| | |
|-------------|--|
| SXTB | Sign extends an 8-bit value to a 3bit value. |
| SXTH | Sign extends a 16-bit value to a 3bit value. |

| | |
|---------------|---|
| SXTB16 | Sign extends two 8-bit values to two 16-bit values. |
| UXTB | Zero extends an 8-bit value to a 3bit value. |
| UXTH | Zero extends a 16-bit value to a 3bit value. |
| UXTB16 | Zero extends two 8-bit values to two 16-bit values. |

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rm is the register holding the value to extend.

ROR #*n* is one of:

ROR #8 Value from *Rm* is rotated right 8 bits.
ROR #16 Value from *Rm* is rotated right 16 bits.
ROR #24 Value from *Rm* is rotated right 24 bits.
If ROR #*n* is omitted, no rotation is performed.

Operation

These instructions do the following:

1. Rotate the value from *Rm* right by 0, 8, 16 or 24 bits.
2. Extract bits from the resulting value:
 - SXTB extracts bits[7:0] and sign extends to 32 bits.
 - UXTB extracts bits[7:0] and zero extends to 32 bits.
 - SXTH extracts bits[15:0] and sign extends to 32 bits.
 - UXTH extracts bits[15:0] and zero extends to 32 bits.
 - SXTB16 extracts bits[7:0] and sign extends to 16 bits, and extracts bits [23:16] and sign extends to 16 bits.
 - UXTB16 extracts bits[7:0] and zero extends to 16 bits, and extracts bits [23:16] and zero extends to 16 bits.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the flags.

Examples

| | |
|-------------|--|
| SXTH | R4, R6, ROR #16; Rotates R6 right by 16 bits, obtains bottom halfword of ; of result, sign extends to 32 bits and writes to R4 |
| UXTB | R3, R10 ; Extracts lowest byte of value in R10, zero extends, and ; writes to R3 |

SXTA and UXTA

Signed and Unsigned Extend and Add

Syntax

*op{cond} {Rd,} Rn, Rm {, ROR #*n*} op{cond} {Rd,} Rn, Rm {, ROR #*n*}* where:
op is one of:

| | |
|----------------|---|
| SXTAB | Sign extends an 8-bit value to a 3bit value and add. |
| SXTAH | Sign extends a 16-bit value to a 3bit value and add. |
| SXTAB16 | Sign extends two 8-bit values to two 16-bit values and add. |
| UXTAB | Zero extends an 8-bit value to a 3bit value and add. |
| UXTAH | Zero extends a 16-bit value to a 3bit value and add. |
| UXTAB16 | Zero extends two 8-bit values to two 16-bit values and add. |

cond is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the first operand register.

Rm is the register holding the value to rotate and extend.

ROR #*n* is one of:

ROR #8 Value from *Rm* is rotated right 8 bits.

ROR #16 Value from *Rm* is rotated right 16 bits.

ROR #24 Value from *Rm* is rotated right 24 bits.

If ROR #*n* is omitted, no rotation is performed.

Operation

These instructions do the following:

1. Rotate the value from *Rm* right by 0, 8, 16 or 24 bits.
2. Extract bits from the resulting value:
 - SXTAB extracts bits[7:0] from *Rm* and sign extends to 32 bits.
 - UXTAB extracts bits[7:0] from *Rm* and zero extends to 32 bits.
 - SXTAH extracts bits[15:0] from *Rm* and sign extends to 32 bits.
 - UXTAH extracts bits[15:0] from *Rm* and zero extends to 32 bits.
 - SXTAB16 extracts bits[7:0] from *Rm* and sign extends to 16 bits, and extracts bits [23:16] from *Rm* and sign extends to 16 bits.
 - UXTAB16 extracts bits[7:0] from *Rm* and zero extends to 16 bits, and extracts bits [23:16] from *Rm* and zero extends to 16 bits.
3. Adds the signed or zero extended value to the word or corresponding halfword of *Rn* and writes the result in *Rd*.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the flags.

Examples

| | |
|--------------|---|
| SXTAH | R4, R8, R6, ROR #16; Rotates R6 right by 16 bits, obtains bottom ; halfword, sign extends to 32 bits, adds R8, and ; writes to R4 |
| UXTAB | R3, R4, R10 ; Extracts bottom byte of R10 and zero extends to 32 ; bits, adds R4, and writes to R3. |

1.3.8 Bitfield instructions

Table 35 shows the instructions that operate on adjacent sets of bits in registers or bitfields:

| Mnemonic | Brief description | See |
|----------|----------------------------|-------------------------------|
| BFC | Bit Field Clear | BFC and BFI |
| BFI | Bit Field Insert | BFC and BFI |
| SBFX | Signed Bit Field Extract | SBFX and UBFX |
| SXTB | Sign extend a byte | SXT and UXT |
| SXTH | Sign extend a halfword | SXT and UXT |
| UBFX | Unsigned Bit Field Extract | SBFX and UBFX |
| UXTB | Zero extend a byte | SXT and UXT |
| UXTH | Zero extend a halfword | SXT and UXT |

37 . Table 2-35 Packing and Unpacking Instructions

BFC and BFI

Bit Field Clear and Bit Field Insert.

Syntax

`BFC{cond} Rd, #lsb, #width BFI{cond} Rd, Rn, #lsb, #width` where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the source register.

lsb is the position of the least significant bit of the bitfield. *lsb* must be in the range 0 to 31.

width is the width of the bitfield and must be in the range 1 to 32–*lsb*.

Operation

BFC clears a bitfield in a register. It clears *width* bits in *Rd*, starting at the low bit position *lsb*. Other bits in *Rd* are unchanged.

BFI copies a bitfield into one register from another register. It replaces *width* bits in *Rd* starting at the low bit position *lsb*, with *width* bits from *Rn* starting at bit[0]. Other bits in *Rd* are unchanged.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the flags.

Examples

| | | |
|-----|------------------|--|
| BFC | R4, #8, #12 | ; Clear bit 8 to bit 19 (12 bits) of R4 to 0 |
| BFI | R9, R2, #8, #12; | Replace bit 8 to bit 19 (12 bits) of R9 with ; bit 0 to bit 11 from R2. |

SBFX and UBFX

Signed Bit Field Extract and Unsigned Bit Field Extract.

Syntax

`SBFX{cond} Rd, Rn, #lsb, #width`

`UBFX{cond} Rd, Rn, #lsb, #width` where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

Rn is the source register.

lsb is the position of the least significant bit of the bitfield. *lsb* must be in the range 0 to 31.

width is the width of the bitfield and must be in the range 1 to 32–*lsb*.

Operation

SBFX extracts a bitfield from one register, sign extends it to 32 bits, and writes the result to the destination register.
UBFX extracts a bitfield from one register, zero extends it to 32 bits, and writes the result to the destination register.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the flags.

Examples

| | |
|------|---|
| SBFX | R0, R1, #20, #4 ; Extract bit 20 to bit 23 (4 bits) from R1 and sign ; extend to 32 bits and then write the result to R0. |
| UBFX | R8, R11, #9, #10; Extract bit 9 to bit 18 (10 bits) from R11 and zero ; extend to 32 bits and then write the result to R8. |

SXT and UXT

Sign extend and Zero extend.

Syntax

`SXTextend{cond} {Rd}, Rm {, ROR #n}`

`UXTextend{cond} {Rd}, Rm {, ROR #n}` where:

| | |
|---------------|--|
| <i>extend</i> | is one of: B Extends an 8-bit value to a 3bit value. H Extends a 16-bit value to a 3bit value. |
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>Rd</i> | is the destination register. |
| <i>Rm</i> | is the register holding the value to extend. |
| <i>ROR #n</i> | is one of: ROR #8 Value from <i>Rm</i> is rotated right 8 bits. ROR #16 Value from <i>Rm</i> is rotated right 16 bits. ROR #24 Value from <i>Rm</i> is rotated right 24 bits. |

If ROR #n is omitted, no rotation is performed.

Operation

These instructions do the following:

1. Rotate the value from *Rm* right by 0, 8, 16 or 24 bits.
2. Extract bits from the resulting value:
 - SXTB extracts bits[7:0] and sign extends to 32 bits.
 - UXTB extracts bits[7:0] and zero extends to 32 bits.
 - SXTH extracts bits[15:0] and sign extends to 32 bits.
 - UXTH extracts bits[15:0] and zero extends to 32 bits.

Restrictions

Do not use SP and do not use PC.

Condition flags

These instructions do not affect the flags.

Examples

| | | |
|------|------------------|--|
| SXTH | R4, R6, ROR #16; | Rotate R6 right by 16 bits, then obtain the lower halfword of the result and then sign extend to 32 bits and write the result to R4. |
| UXTB | R3, R10 | ; Extract lowest byte of the value in R10 and zero ; extend it, and write the result to R3. |

1.3.9 Branch and control instructions

Table 35 shows the branch and control instructions:

| Mnemonic | Brief description | See |
|----------|--------------------------------|--------------------|
| B | Branch | B, BL, BX, and BLX |
| BL | Branch with Link | B, BL, BX, and BLX |
| BLX | Branch indirect with Link | B, BL, BX, and BLX |
| BX | Branch indirect | B, BL, BX, and BLX |
| CBNZ | Compare and Branch if Non Zero | CBZ and CBNZ |
| CBZ | Compare and Branch if Zero | CBZ and CBNZ |
| IT | If-Then | IT |
| TBB | Table Branch Byte | TBB and TBH |
| TBH | Table Branch Halfword | TBB and TBH |

38 .35 Branch and Control Instructions

B, BL, BX, and BLX

Branch instructions.

Syntax

B{cond} label

BL{cond} label

BX{cond} Rm BLX{cond} Rm where:

B is branch (immediate).

BL is branch with link (immediate).

BX is branch indirect (register). BLX is branch indirect with link (register).

cond is an optional condition code, see [Conditional execution](#). label is a PC-relative expression. See [PC-relative expressions](#).

Rm is a register that indicates an address to branch to. Bit[0] of the value in Rm must be 1, but the address to branch to is created by changing bit[0] to 0.

Operation

All these instructions cause a branch to *label*, or to the address indicated in *Rm*. In addition:

- The BL and BLX instructions write the address of the next instruction to LR (the link register, R14).
- The BX and BLX instructions result in a UsageFault exception if bit[0] of *Rm* is 0.

Bcond label is the only conditional instruction that can be either inside or outside an IT block. All other branch instructions must be conditional inside an IT block, and must be unconditional outside the IT block, see [IT](#).

Table 36 shows the ranges for the various branch instructions.

| Instruction | Branch range |
|---------------------------------------|-----------------------|
| B label | -16 MB to +16 MB |
| <i>Bcond</i> label (outside IT block) | -1 MB to +1 MB |
| <i>Bcond</i> label (inside IT block) | -16 MB to +16 MB |
| BL{cond} label | -16 MB to +16 MB |
| BX{cond} Rm | Any value in register |
| BLX{cond} Rm | Any value in register |

39 .36 Branch Ranges

Note:

You might have to use the .W suffix to get the maximum branch range. See [Instruction width selection](#).

Restrictions

The restrictions are:

- do not use PC in the BLX instruction
- for BX and BLX, bit[0] of *Rm* must be 1 for correct execution but a branch occurs to the target address created by changing bit[0] to 0
- when any of these instructions is inside an IT block, it must be the last instruction of the IT block.

Note: *Bcond* is the only conditional instruction that is not required to be inside an IT block. However, it has a longer branch range when it is inside an IT block.

Condition flags

These instructions do not change the flags.

Examples

| | | |
|-------|--------|---|
| B | loopA | ; Branch to loopA |
| BLE | ng | ;Conditionally branch to label ng |
| B.W | target | ;Branch to target within 16MB range |
| BEQ | target | ; Conditionally branch to target |
| BEQ.W | target | ; Conditionally branch to target within 1MB |
| BL | funC | ; Branch with link (Call) to function funC, return address ; stored in LR |
| BX | LR | ; Return from function call |
| BXNE | R0 | ; Conditionally branch to address stored in R0 |

| | | |
|-----|----|---|
| BLX | R0 | ; Branch with link and exchange (Call) to a address stored ; in R0. |
|-----|----|---|

CBZ and CBNZ

Compare and Branch on Zero, Compare and Branch on Non-Zero.

Syntax

CBZ *Rn, label* CBNZ *Rn, label* where:

Rn is the register holding the operand. *label* is the branch destination.

Operation

Use the CBZ or CBNZ instructions to avoid changing the condition code flags and to reduce the number of instructions.

CBZ *Rn, label* does not change condition flags but is otherwise equivalent to:

CMPR*n, #0 BEQ*label

CBNZ *Rn, label* does not change condition flags but is otherwise equivalent to:

CMPR*n, #0 BNE*label

Restrictions

The restrictions are:

- *Rn* must be in the range of R0 to R7
- the branch destination must be within 4 to 130 bytes after the instruction
- these instructions must not be used inside an IT block.

Condition flags

These instructions do not change the flags.

Examples

| | |
|------|--|
| CBZ | R5, target; Forward branch if R5 is zero |
| CBNZ | R0, target; Forward branch if R0 is not zero |

IT

If-Then condition instruction.

Syntax

IT{*x\y\z*} *cond* where: *x* specifies the condition switch for the second instruction in the IT block. *y* specifies the condition switch for the third instruction in the IT block. *z* specifies the condition switch for the fourth instruction in the IT block. *cond* specifies the condition for the first instruction in the IT block.

The condition switch for the second, third and fourth instruction in the IT block can be either:

T Then. Applies the condition *cond* to the instruction.

E Else. Applies the inverse condition of *cond* to the instruction.

Note: It is possible to use AL (the *always* condition) for *cond* in an IT instruction. If this is done, all of the instructions in the IT block must be unconditional, and each of *x*, *y*, and *z* must be T or omitted but not E.

Operation

The IT instruction makes up to four following instructions conditional. The conditions can be all the same, or some of them can be the logical inverse of the others. The conditional instructions following the IT instruction form the *IT block*.

The instructions in the IT block, including any branches, must specify the condition in the {*cond*} part of their syntax.

Note: Your assembler might be able to generate the required IT instructions for conditional instructions automatically, so that you do not have to write them yourself. See your assembler documentation for details.

A BKPT instruction in an IT block is always executed, even if its condition fails.

Exceptions can be taken between an IT instruction and the corresponding IT block, or within an IT block. Such an exception results in entry to the appropriate exception handler, with suitable return information in LR and stacked PSR.

Instructions designed for use for exception returns can be used as normal to return from the exception, and

execution of the IT block resumes correctly. This is the only way that a PC-modifying instruction is permitted to branch to an instruction in an IT block.

Restrictions

The following instructions are not permitted in an IT block:

- IT
- CBZ and CBNZ
- CPSID and CPSIE.

Other restrictions when using an IT block are:

- a branch or any instruction that modifies the PC must either be outside an IT block or must be the last instruction inside the IT block. These are:

— ADD PC, PC, Rm
— MOV PC, Rm
— B, BL, BX, BLX
— any LDM, LDR, or POP instruction that writes to the PC
— TBB and TBH

- do not branch to any instruction inside an IT block, except when returning from an exception handler
- all conditional instructions except *Bcond* must be inside an IT block. *Bcond* can be either outside or inside an IT block but has a larger branch range if it is inside one
- each instruction inside the IT block must specify a condition code suffix that is either the same or logical inverse as for the other instructions in the block.



Note

Your assembler might place extra restrictions on the use of IT blocks, such as prohibiting the use of assembler directives within them.

Condition flags

This instruction does not change the flags.

Example

| | | |
|--------|-------------|--|
| ITTE | NE | ; Next 3 instructions are conditional |
| ANDNE | R0, R0, R1 | ; ANDNE does not update condition flags |
| ADDSNE | R2, R2, #1 | ; ADDSNE updates condition flags |
| MOVEQ | R2, R3 | ; Conditional move |
| CMP | R0, #9 | ; Convert R0 hex value (0 to 15) into ASCII ; ('0-9' 'A-F') |
| ITE | GT | ; Next 2 instructions are conditional |
| ADDGT | R1, R0, #55 | ; Convert 0xA -> 'A' |
| ADDLE | R1, R0, #48 | ; Convert 0x0 -> '0' |
| IT | GT | ; IT block with only one conditional instruction |
| ADDGT | R1, R1, #1 | ; Increment R1 conditionally |
| ITTEE | EQ | ; Next 4 instructions are conditional |
| MOVEQ | R0, R1 | ; Conditional move |
| ADDEQ | R2, R2, #10 | ; Conditional add |
| ANDNE | R3, R3, #1 | ; Conditional AND |
| BNE.W | dloop | ; Branch instruction can only be used in the last instruction of an IT block |
| IT | NE | ; Next instruction is conditional |
| ADD | R0, R0, R1 | ; Syntax error: no condition code used in IT block |

TBB and TBH

Table Branch Byte and Table Branch Halfword.

Syntax

TBB [*Rn, Rm*] TBH [*Rn, Rm, LSL #1*] where:

Rn is the register containing the address of the table of branch lengths.

If *Rn* is PC, then the address of the table is the address of the byte immediately following the TBB or TBH instruction.

Rm is the index register. This contains an index into the table. For halfword tables, LSL #1 doubles the value in *Rm* to form the right offset into the table.

Operation

These instructions cause a PC-relative forward branch using a table of single byte offsets for TBB, or halfword offsets for TBH. *Rn* provides a pointer to the table, and *Rm* supplies an index into the table. For TBB the branch offset is twice the unsigned value of the byte returned from the table, and for TBH the branch offset is twice the unsigned value of the halfword returned from the table. The branch occurs to the address at that offset from the address of the byte immediately after the TBB or TBH instruction.

Restrictions

The restrictions are:

- *Rn* must not be SP
- *Rm* must not be SP and must not be PC
- when any of these instructions is used inside an IT block, it must be the last instruction of the IT block.

Condition flags

These instructions do not change the flags.

Examples

```
ADR.W R0, BranchTable_Byte
TBB [R0, R1] ; R1 is the index, R0 is the base address of the branch table
```

Case1

; an instruction sequence follows

Case2

; an instruction sequence follows

Case3

; an instruction sequence follows

```
BranchTable_Byte
  DCB 0 ; Case1 offset calculation
  DCB ((CaseA-Case1)/2) ; Case2 offset calculation
  DCB ((Case3-Case1)/2) ; Case3 offset calculation
  TBH [PC, R1, LSL #1] ; R1 is the index, PC is used as base of the branch table
```

BranchTable_H

```
  DCI ((CaseA - BranchTable_H)/2) ; CaseA offset calculation
  DCI ((CaseB - BranchTable_H)/2) ; CaseB offset calculation
  DCI ((CaseC - BranchTable_H)/2) ; CaseC offset calculation
```

CaseA

; an instruction sequence follows

CaseB

; an instruction sequence follows

CaseC

; an instruction sequence follows

1.3.10 Floating-point instructions

Table 15 shows the floating-point instructions.

Note

These instructions are only available if the FPU is included, and enabled, in the system. See *Enabling the FPU* on page 4-51 for information about enabling the floating-point unit.

| Mnemonic | Brief description | See |
|----------|--|---|
| VABS | Floating-point Absolute | VABS |
| VADD | Floating-point Add | VADD |
| VCMP | Compare two floating-point registers, or one floating-point register and zero | VCMP , VCMPE |
| VCMPE | Compare two floating-point registers, or one floating-point register and zero with Invalid Operation check | VCMP , VCMPE |
| VCVT | Convert between floating-point and integer | VCVT , VCVTR between floating-point and integer |
| VCVT | Convert between floating-point and fixed point | VCVT between floating-point and fixed-point |
| VCVTR | Convert between floating-point and integer with rounding | VCVT , VCVTR between floating-point and integer |
| VCVTB | Converts half-precision value to single-precision | VCVTB , VCVTT |
| VCVTT | Converts single-precision register to half-precision | VCVTB , VCVTT |
| VDIV | Floating-point Divide | VDIV |
| VFMA | Floating-point Fused Multiply Accumulate | VFMA , VFMS |
| VFNMA | Floating-point Fused Negate Multiply Accumulate | VFNMA , VFNMS |
| VFMS | Floating-point Fused Multiply Subtract | VFMA , VFMS |
| VFNMS | Floating-point Fused Negate Multiply Subtract | VFNMA , VFNMS |
| VLDM | Load Multiple extension registers | VLDM |
| VLDR | Loads an extension register from memory | VLDR |
| VLMA | Floating-point Multiply Accumulate | VLMA , VLMS |
| VLMS | Floating-point Multiply Subtract | VLMA , VLMS |
| VMOV | Floating-point Move Immediate | VMOV Immediate |
| VMOV | Floating-point Move Register | VMOV Register |
| VMOV | Copy ARM core register to single precision | VMOVARM Core register to single precision |
| VMOV | Copy 2 ARM core registers to 2 single precision | VMOV Two ARM Core registers to two single precision |
| VMOV | Copies between ARM core register to scalar | VMOVARM Core register to scalar |

40 . Floating-point Instructions

| Mnemonic | Brief description | See |
|----------|---|--|
| VMOV | Copies between Scalar to ARM core register | VMOV Scalar to ARM Core register |
| VMRS | Move to ARM core register from floating-point System Register | VMRS |
| VMSR | Move to floating-point System Register from ARM Core register | VMSR |
| VMUL | Multiply floating-point | VMUL |
| VNEG | Floating-point negate | VNEG |
| VNMLA | Floating-point multiply and add | VNMLA, VNMLS, VNMUL |
| VNMLS | Floating-point multiply and subtract | VNMLA, VNMLS, VNMUL |
| VNMUL | Floating-point multiply | VNMLA, VNMLS, VNMUL |
| VPOP | Pop extension registers | VPOP |
| VPUSH | Push extension registers | VPUSH |
| VSQRT | Floating-point square root | VSQRT |
| VSTM | Store Multiple extension registers | VSTM |
| VSTR | Stores an extension register to memory | VSTR |
| VSUB | Floating-point Subtract | VSUB |

41 . Floating-Point Instructions (continued)

VABS

Floating-point Absolute.

Syntax

`VABS{cond}.F32 Sd, Sm` where: *cond* is an optional condition code, see [Conditional execution](#). *Sd, Sm* are the destination floating-point value and the operand floating-point value.

Operation

This instruction:

1. Takes the absolute value of the operand floating-point register.
2. Places the results in the destination floating-point register.

Restrictions

There are no restrictions.

Condition flags

The floating-point instruction clears the sign bit.

Examples

`VABS.F-32 S4, S6`

VADD

Floating-point Add

Syntax

`VADD{cond}.F32 {Sd,} Sn, Sm` where: *cond* is an optional condition code, see [Conditional execution](#).

Sd is the destination floating-point value.

Sn, Sm are the operand floating-point values.

Operation

This instruction:

1. Adds the values in the two floating-point operand registers.
2. Places the results in the destination floating-point register.

Restrictions

There are no restrictions.

Condition flags

This instruction does not change the flags.

Examples

`VADD.F-32 S4, S6, S7`

VCMP, VCMPE

Compares two floating-point registers, or one floating-point register and zero.

Syntax

`VCMP{E}{cond}.F32 Sd, Sm` `VCMP{E}{cond}.F32 Sd, #0.0` where: *cond* is an optional condition code, see [Conditional execution](#).

E If present, any NaN operand causes an Invalid Operation exception. Otherwise, only a signaling NaN causes the exception.

Sd is the floating-point operand to compare.

Sm is the floating-point operand that is compared with.

Operation

This instruction:

1. Compares:
 - Two floating-point registers.
 - One floating-point register and zero.
2. Writes the result to the FPSCR flags.

Restrictions

This instruction can optionally raise an Invalid Operation exception if either operand is any type of NaN. It always raises an Invalid Operation exception if either operand is a signaling NaN.

Condition flags

When this instruction writes the result to the FPSCR flags, the values are normally transferred to the ARM flags by a subsequent VMRS instruction, see [VMRS](#).

Examples

`VCMP.F-32 S4, #0.0`
`VCMP.F-32 S4, S2`

VCVT, VCVTR between floating-point and integer

Converts a value in a register from floating-point to a 3bit integer.

Syntax

`VCVT{R}{cond}.Tm.F32 Sd, Sm` `VCVT{cond}.F32.Tm Sd, Sm` where:

| | |
|---------------|---|
| <i>R</i> | If <i>R</i> is specified, the operation uses the rounding mode specified by the FPSCR. If <i>R</i> is omitted, the operation uses the Round towards Zero rounding mode. |
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>Tm</i> | is the data type for the operand. It must be one of: <ul style="list-style-type: none"> • S32 signed 3bit value. • U32 unsigned 3bit value. |
| <i>Sd, Sm</i> | are the destination register and the operand register. |

Operation

These instructions:

1. Either
 - Converts a value in a register from floating-point value to a 3bit integer.
 - Converts from a 3bit integer to floating-point value.
2. Places the result in a second register.

The floating-point to integer operation normally uses the Round towards Zero rounding mode, but can optionally use the rounding mode specified by the FPSCR.

The integer to floating-point operation uses the rounding mode specified by the FPSCR.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VCVT between floating-point and fixed-point

Converts a value in a register from floating-point to and from fixed-point.

Syntax

`VCVT{cond}.Td.F32 Sd, Sd, #fbits` `VCVT{cond}.F32.Td Sd, Sd, #fbits` where: *cond* is an optional condition code, see [Conditional execution](#).

Td is the data type for the fixed-point number. It must be one of:

- S16 signed 16-bit value.
- U16 unsigned 16-bit value.
- S32 signed 3bit value.
- U32 unsigned 3bit value.

Sd is the destination register and the operand register.

fbits is the number of fraction bits in the fixed-point number:

- If *Td* is S16 or U16, *fbits* must be in the range 0-16.
- If *Td* is S32 or U32, *fbits* must be in the range 1-32.

Operation

These instructions:

1. Either

- Converts a value in a register from floating-point to fixed-point.
- Converts a value in a register from fixed-point to floating-point.

2. Places the result in a second register.

The floating-point values are single-precision.

The fixed-point value can be 16-bit or 3bit. Conversions from fixed-point values take their operand from the low-order bits of the source register and ignore any remaining bits.

Signed conversions to fixed-point values sign-extend the result value to the destination register width.

Unsigned conversions to fixed-point values zero-extend the result value to the destination register width.

The floating-point to fixed-point operation uses the Round towards Zero rounding mode. The fixed-point to floating-point operation uses the Round to Nearest rounding mode.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VCVTB, VCVTT

Converts between a half-precision value and a single-precision value.

Syntax

VCVT{y}{cond}.F32.F16 *Sd, Sm*

VCVT{y}{cond}.F16.F32 *Sd, Sm* where:

y Specifies which half of the operand register *Sm* or destination register *Sd* is used for the operand or destination:

- If *y* is B, then the bottom half, bits [15:0], of *Sm* or *Sd* is used.
- If *y* is T, then the top half, bits [31:16], of *Sm* or *Sd* is used.

cond is an optional condition code, see [Conditional execution](#).

Sd is the destination register.

Sm is the operand register.

Operation

This instruction with the .F16.32 suffix:

1. Converts the half-precision value in the top or bottom half of a single-precision register to single-precision.
2. Writes the result to a single-precision register.

This instruction with the .F32.F16 suffix:

1. Converts the value in a single-precision register to half-precision.
2. Writes the result into the top or bottom half of a single-precision register, preserving the other half of the target register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VDIV

Divides floating-point values.

Syntax

VDIV{cond}.F32 {*Sd*,} *Sn, Sm* where: *cond* is an optional condition code, see [Conditional execution](#).

Sd is the destination register.

Sn, Sm are the operand registers.

Operation

This instruction:

1. Divides one floating-point value by another floating-point value.
2. Writes the result to the floating-point destination register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VFMA, VFMS

Floating-point Fused Multiply Accumulate and Subtract.

Syntax

$\text{VFMA}\{\text{cond}\}.\text{F32 }\{Sd\} Sn, Sm$
 $\text{VFMS}\{\text{cond}\}.\text{F32 }\{Sd\} Sn, Sm$ where: *cond* is an optional condition code, see [Conditional execution](#).
Sd is the destination register.
Sn, Sm are the operand registers.

Operation

The VFMA instruction:

1. Multiplies the floating-point values in the operand registers.
2. Accumulates the results into the destination register.

The result of the multiply is not rounded before the accumulation.

The VFMS instruction:

1. Negates the first operand register.
2. Multiplies the floating-point values of the first and second operand registers.
3. Adds the products to the destination register.
4. Places the results in the destination register.

The result of the multiply is not rounded before the addition.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VFNMA, VFNMS

Floating-point Fused Negate Multiply Accumulate and Subtract.

Syntax

$\text{VFNMA}\{\text{cond}\}.\text{F32 }\{Sd\} Sn, Sm$
 $\text{VFNMS}\{\text{cond}\}.\text{F32 }\{Sd\} Sn, Sm$ where: *cond* is an optional condition code, see [Conditional execution](#).
Sd is the destination register.
Sn, Sm are the operand registers.

Operation

The VFNMA instruction:

1. Negates the first floating-point operand register.
2. Multiplies the first floating-point operand with second floating-point operand.
3. Adds the negation of the floating-point destination register to the product
4. Places the result into the destination register.

The result of the multiply is not rounded before the addition.

The VFNMS instruction:

1. Multiplies the first floating-point operand with second floating-point operand.
2. Adds the negation of the floating-point value in the destination register to the product.
3. Places the result in the destination register.

The result of the multiply is not rounded before the addition.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VLDM

Floating-point Load Multiple

Syntax

VLDM{mode}{cond}{.size} Rn[], list where:

mode is the addressing mode:

- IA Increment After. The consecutive addresses start at the address specified in Rn.
- DB Decrement Before. The consecutive addresses end just before the address specified in Rn.

cond is an optional condition code, see [Conditional execution](#). size is an optional data size specifier. Rn is the base register. The SP can be used.

/ is the command to the instruction to write a modified value back to Rn. This is required if mode == DB, and is optional if mode == IA.

list is the list of extension registers to be loaded, as a list of consecutively numbered doubleword or singleword registers, separated by commas and surrounded by brackets.

Operation

This instruction loads:

- Multiple extension registers from consecutive memory locations using an address from an ARM core register as the base address.

Restrictions

The restrictions are:

- If size is present, it must be equal to the size in bits, 32 or 64, of the registers in list.
- For the base address, the SP can be used.

In the ARM instruction set, if / is not specified the PC can be used.

- list must contain at least one register. If it contains doubleword registers, it must not contain more than 16 registers.
- If using the Decrement Before addressing mode, the write back flag, /, must be appended to the base register specification.

Condition flags

These instructions do not change the flags.

VLDR

Loads a single extension register from memory

Syntax

VLDR{cond}{.64} Dd, [Rn#imm]

VLDR{cond}{.64} Dd, label

VLDR{cond}{.64} Dd, [PC, #imm]

VLDR{cond}{.32} Sd, [Rn#, imm]

VLDR{cond}{.32} Sd, label VLDR{cond}{.32} Sd, [PC, #imm] where:

| | |
|---------------|---|
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>64, 32</i> | are the optional data size specifiers. |
| <i>Dd</i> | is the destination register for a doubleword load. |
| <i>Sd</i> | is the destination register for a singleword load. |
| <i>Rn</i> | is the base register. The SP can be used. |
| <i>imm</i> | is the + or - immediate offset used to form the address. Permitted address values are multiples of 4 in the range 0 to 1020. |
| <i>label</i> | is the label of the literal data item to be loaded. |

Operation

This instruction:

- Loads a single extension register from memory, using a base address from an ARM core register, with an optional offset.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VLMA, VLMS

Multiplies two floating-point values, and accumulates or subtracts the results.

Syntax

VLMA{*cond*}.F32 *Sd, Sn, Sm*

VLMS{*cond*}.F32 *Sd, Sn, Sm* where: *cond* is an optional condition code, see [Conditional execution](#).

Sd is the destination floating-point value.

Sn, Sm are the operand floating-point values.

Operation

The floating-point Multiply Accumulate instruction:

1. Multiplies two floating-point values.
2. Adds the results to the destination floating-point value.

The floating-point Multiply Subtract instruction:

1. Multiplies two floating-point values.
2. Subtracts the products from the destination floating-point value.
3. Places the results in the destination register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VMOV Immediate

Move floating-point Immediate

Syntax

VMOV{*cond*}.F32 *Sd, #imm* where: *cond* is an optional condition code, see [Conditional execution](#).

Sd is the branch destination. *imm* is a floating-point constant.

Operation

This instruction copies a constant value to a floating-point register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VMOV Register

Copies the contents of one register to another.

Syntax

$VMOV\{cond\}.F64 Dd, Dm$

$VMOV\{cond\}.F32 Sd, Sm$ where: *cond* is an optional condition code, see [Conditional execution](#).

Dd is the destination register, for a doubleword operation.

Dm is the source register, for a doubleword operation.

Sd is the destination register, for a singleword operation.

Sm is the source register, for a singleword operation.

Operation

This instruction copies the contents of one floating-point register to another.

Restrictions

There are no restrictions

Condition flags

These instructions do not change the flags.

VMOV Scalar to ARM Core register

Transfers one word of a doubleword floating-point register to an ARM core register.

Syntax

$VMOV\{cond\} Rt, Dn[x]$ where: *cond* is an optional condition code, see [Conditional execution](#).

Rt is the destination ARM core register.

Dn is the 64-bit doubleword register.

x Specifies which half of the doubleword register to use:

- If *x* is 0, use lower half of doubleword register
- If *x* is 1, use upper half of doubleword register.

Operation

This instruction transfers:

- one word from the upper or lower half of a doubleword floating-point register to an ARM core register.

Restrictions

Rt cannot be PC or SP.

Condition flags

These instructions do not change the flags.

VMOV ARM Core register to single precision

Transfers a single-precision register to and from an ARM core register.

Syntax

$VMOV\{cond\} Sn, Rt$ $VMOV\{cond\} Rt, Sn$ where: *cond* is an optional condition code, see [Conditional execution](#).

Sn is the single-precision floating-point register.
Rt is the ARM core register.

Operation

This instruction transfers:

- The contents of a single-precision register to an ARM core register.
- The contents of an ARM core register to a single-precision register.

Restrictions

Rt cannot be PC or SP.

Condition flags

These instructions do not change the flags.

VMOV Two ARM Core registers to two single precision

Transfers two consecutively numbered single-precision registers to and from two ARM core registers.

Syntax

VMOV{cond} Sm, Sm1, Rt, Rt2 VMOV{cond} Rt, Rt2, Sm, Sm where: *cond* is an optional condition code, see [Conditional execution](#).

Sm is the first single-precision register.

Sm1 is the second single-precision register.

This is the next single-precision register after *Sm*.

Rt is the ARM core register that *Sm* is transferred to or from.

Rt2 is the The ARM core register that *Sm1* is transferred to or from.

Operation

This instruction transfers:

- The contents of two consecutively numbered single-precision registers to two ARM core registers.
- The contents of two ARM core registers to a pair of single-precision registers.

Restrictions

The restrictions are:

- The floating-point registers must be contiguous, one after the other.
- The ARM core registers do not have to be contiguous.
- *Rt* cannot be PC or SP.

Condition flags

These instructions do not change the flags.

VMOV ARM Core register to scalar

Transfers one word to a floating-point register from an ARM core register.

Syntax

VMOV{cond}{.32} Dd[x], Rt where: *cond* is an optional condition code, see [Conditional execution](#).

.32 is an optional data size specifier.

Dd[x] is the destination, where [x] defines which half of the doubleword is transferred, as follows:

- If *x* is 0, the lower half is extracted
- If *x* is 1, the upper half is extracted.

Rt is the source ARM core register.

Operation

This instruction transfers one word to the upper or lower half of a doubleword floating-point register from an ARM core register.

Restrictions

Rt cannot be PC or SP.

Condition flags

These instructions do not change the flags.

VMRS

Move to ARM Core register from floating-point System Register.

Syntax

VMRS{cond} Rt, FPSCR VMRS{cond} APSR_nzcv, FPSCR where: *cond* is an optional condition code, see [Conditional execution](#).

Rt is the destination ARM core register. This register can be R0-R14.

APSR_nzcv Transfer floating-point flags to the APSR flags.

Operation

This instruction performs one of the following actions:

- Copies the value of the FPSCR to a general-purpose register.
- Copies the value of the FPSCR flag bits to the APSR N, Z, C, and V flags.

Restrictions

Rt cannot be PC or SP.

Condition flags

These instructions optionally change the flags: N, Z, C, V

VMSR

Move to floating-point System Register from ARM Core register.

Syntax

VMSR{cond} FPSCR, *Rt* where: *cond* is an optional condition code, see [Conditional execution](#).

Rt is the general-purpose register to be transferred to the FPSCR.

Operation

This instruction moves the value of a general-purpose register to the FPSCR. See [Floating-point Status Control Register](#) for more information.

Restrictions

The restrictions are:

- *Rt* cannot be PC or SP.

Condition flags

This instruction updates the FPSCR.

VMUL

Floating-point Multiply.

Syntax

VMUL{cond}.F32 {*Sd*,} *Sn*, *Sm* where: *cond* is an optional condition code, see [Conditional execution](#).

Sd is the destination floating-point value.

Sn, *Sm* are the operand floating-point values.

Operation

This instruction:

1. Multiplies two floating-point values.
2. Places the results in the destination register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VNEG

Floating-point Negate.

Syntax

$VNEG\{cond\}.F32 Sd, Sm$ where: $cond$ is an optional condition code, see [Conditional execution](#).

Sd is the destination floating-point value.

Sm is the operand floating-point value.

Operation

This instruction:

1. Negates a floating-point value.
2. Places the results in a second floating-point register.

The floating-point instruction inverts the sign bit.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VNMLA, VNMLS, VNMUL

Floating-point multiply with negation followed by add or subtract.

Syntax

$VNMLA\{cond\}.F32 Sd, Sn, Sm$

$VNMLS\{cond\}.F32 Sd, Sn, Sm$ $VNMUL\{cond\}.F32 \{Sd,\} Sn, Sm$ where: $cond$ is an optional condition code, see [Conditional execution](#).

Sd is the destination floating-point register.

Sn, Sm are the operand floating-point registers.

Operation

The VNMLA instruction:

1. Multiplies two floating-point register values.
2. Adds the negation of the floating-point value in the destination register to the negation of the product.
3. Writes the result back to the destination register.

The VNMLS instruction:

1. Multiplies two floating-point register values.
2. Adds the negation of the floating-point value in the destination register to the product.
3. writes the result back to the destination register.

The VNMUL instruction:

1. Multiplies together two floating-point register values.
2. Writes the negation of the result to the destination register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VPOP

Floating-point extension register Pop.

Syntax

`VPOP{cond}{.size} list` where: *cond* is an optional condition code, see [Conditional execution](#).

size is an optional data size specifier.

If present, it must be equal to the size in bits, 32 or 64, of the registers in *list*.

list is a list of extension registers to be loaded, as a list of consecutively numbered doubleword or singleword registers, separated by commas and surrounded by brackets.

Operation

This instruction loads multiple consecutive extension registers from the stack.

Restrictions

The list must contain at least one register, and not more than sixteen registers.

Condition flags

These instructions do not change the flags.

VPUSH

Floating-point extension register Push.

Syntax

`VPUSH{cond}{.size} list` where: *cond* is an optional condition code, see [Conditional execution](#).

size is an optional data size specifier.

If present, it must be equal to the size in bits, 32 or 64, of the registers in *list*.

list is a list of the extension registers to be stored, as a list of consecutively numbered doubleword or singleword registers, separated by commas and surrounded by brackets.

Operation

This instruction:

- Stores multiple consecutive extension registers to the stack.

Restrictions

The restrictions are:

- list must contain at least one register, and not more than sixteen.

Condition flags

These instructions do not change the flags.

VSQRT

Floating-point Square Root.

Syntax

`VSQRT{cond}.F32 Sd, Sm` where: *cond* is an optional condition code, see [Conditional execution](#).

Sd is the destination floating-point value.

Sm is the operand floating-point value.

Operation

This instruction:

- Calculates the square root of the value in a floating-point register.
- Writes the result to another floating-point register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

VSTM

Floating-point Store Multiple.

Syntax

`VSTM{mode}{cond}{.size} Rn[], list` where:
mode is the addressing mode:

- IA *Increment After*. The consecutive addresses start at the address specified in *Rn*. This is the default and can be omitted.
- DB *Decrement Before*. The consecutive addresses end just before the address specified in *Rn*.

cond is an optional condition code, see [Conditional execution](#).

size is an optional data size specifier.

If present, it must be equal to the size in bits, 32 or 64, of the registers in *list*.

Rn is the base register. The SP can be used.

/ is the function that causes the instruction to write a modified value back to *Rn*. Required if mode == DB.

list is a list of the extension registers to be stored, as a list of consecutively numbered doubleword or singleword registers, separated by commas and surrounded by brackets.

Operation

This instruction:

- Stores multiple extension registers to consecutive memory locations using a base address from an ARM core register.

Restrictions

The restrictions are:

- *list* must contain at least one register.

If it contains doubleword registers it must not contain more than 16 registers.

- Use of the PC as *Rn* is deprecated.

Condition flags

These instructions do not change the flags.

VSTR

Floating-point Store.

Syntax

`VSTR{cond}{.32} Sd, [Rn, #imm]`
`VSTR{cond}{.64} Dd, [Rn, #imm]` where:

| | |
|---------------|---|
| <i>cond</i> | is an optional condition code, see Conditional execution . |
| <i>32, 64</i> | are the optional data size specifiers. |
| <i>Sd</i> | is the source register for a singleword store. |
| <i>Dd</i> | is the source register for a doubleword store. |
| <i>Rn</i> | is the base register. The SP can be used. |
| <i>imm</i> | is the + or - immediate offset used to form the address. Values are multiples of 4 in the range 0-1020. <i>imm</i> can be omitted, meaning an offset of +0. |

Operation

This instruction:

- Stores a single extension register to memory, using an address from an ARM core register, with an optional offset, defined in *imm*.

Restrictions

The restrictions are:

- The use of PC for Rn is deprecated.

Condition flags

These instructions do not change the flags.

VSUB

Floating-point Subtract.

Syntax

$VSUB\{cond\}.F32 \{Sd\} Sn, Sm$ where: $cond$ is an optional condition code, see [Conditional execution](#).

Sd is the destination floating-point value.

Sn, Sm are the operand floating-point value.

Operation

This instruction:

1. Subtracts one floating-point value from another floating-point value.
2. Places the results in the destination floating-point register.

Restrictions

There are no restrictions.

Condition flags

These instructions do not change the flags.

1.3.11 Miscellaneous instructions

Table 38 shows the remaining Cortex-M4 instructions:

| Mnemonic | Brief description | See |
|----------|--|----------------------|
| BKPT | Breakpoint | BKPT |
| CPSID | Change Processor State, Disable Interrupts | CPS |
| CPSIE | Change Processor State, Enable Interrupts | CPS |
| DMB | Data Memory Barrier | DMB |
| DSB | Data Synchronization Barrier | DSB |
| ISB | Instruction Synchronization Barrier | ISB |
| MRS | Move from special register to register | MRS |
| MSR | Move from register to special register | MSR |
| NOP | No Operation | NOP |
| SEV | Send Event | SEV |
| SVC | Supervisor Call | SVC |
| WFE | Wait For Event | WFE |
| WFI | Wait For Interrupt | WFI |

42 . Miscellaneous Instructions

BKPT

Breakpoint.

Syntax

BKPT #*imm* where: *imm* is an expression evaluating to an integer in the range 0-255 (8-bit value).

Operation

The BKPT instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached.

imm is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The BKPT instruction can be placed inside an IT block, but it executes unconditionally, unaffected by the condition specified by the IT instruction.

Condition flags

This instruction does not change the flags.

Examples

| | | |
|------|------|---|
| BKPT | #0x3 | ; Breakpoint with immediate value set to 0x3 (debugger can extract the immediate value by locating it using the PC) |
|------|------|---|

CPS

Change Processor State.

Syntax

CPS*effect iflags* where:

| | |
|---------------|--|
| <i>effect</i> | is one of: IE Clears the special purpose register. ID Sets the special purpose register. |
| <i>iflags</i> | is a sequence of one or more flags: i Set or clear PRIMASK. f Set or clear FAULTMASK. |

Operation

CPS changes the PRIMASK and FAULTMASK special register values. See [Exception mask registers](#) for more information about these registers.

Restrictions

The restrictions are:

- use CPS only from privileged software, it has no effect if used in unprivileged software
- CPS cannot be conditional and so must not be used inside an IT block.

Condition flags

This instruction does not change the condition flags.

Examples

| | |
|---------|---|
| CPSID i | ; Disable interrupts and configurable fault handlers (set PRIMASK) |
| CPSID f | ; Disable interrupts and all fault handlers (set FAULTMASK) |
| CPSID i | ; Enable interrupts and configurable fault handlers (clear PRIMASK) |
| CPSID f | ; Enable interrupts and fault handlers (clear FAULTMASK) |

DMB

Data Memory Barrier.

Syntax

DMB{cond} where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

DMB acts as a data memory barrier. It ensures that all explicit memory accesses that appear, in program order, before the DMB instruction are completed before any explicit memory accesses that appear, in program order, after the DMB instruction. DMB does not affect the ordering or execution of instructions that do not access memory.

Condition flags

This instruction does not change the flags.

Examples

DMB; Data Memory Barrier

DSB

Data Synchronization Barrier.

Syntax

DSB{cond} where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

DSB acts as a special data synchronization memory barrier. Instructions that come after the DSB, in program order, do not execute until the DSB instruction completes. The DSB instruction completes when all explicit memory accesses before it complete.

Condition flags

This instruction does not change the flags.

Examples

DSB; Data Synchronisation Barrier

ISB

Instruction Synchronization Barrier.

Syntax

ISB{cond} where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

ISB acts as an instruction synchronization barrier. It flushes the pipeline of the processor, so that all instructions following the ISB are fetched from cache or memory again, after the ISB instruction has been completed.

Condition flags

This instruction does not change the flags.

Examples

ISB; Instruction Synchronisation Barrier

MRS

Move the contents of a special register to a general-purpose register.

Syntax

`MRS{cond} Rd, spec_reg` where: *cond* is an optional condition code, see [Conditional execution](#).

Rd is the destination register.

spec_reg can be any of: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK, or CONTROL.



Note

All the EPSR and IPSR fields are zero when read by the MRS instruction.

Operation

Use MRS in combination with MSR as part of a read-modify-write sequence for updating a PSR, for example to clear the Q flag.

In process swap code, the programmers model state of the process being swapped out must be saved, including relevant PSR contents. Similarly, the state of the process being swapped in must also be restored. These operations use MRS in the state-saving instruction sequence and MSR in the state-restoring instruction sequence.



Note

BASEPRI_MAX is an alias of BASEPRI when used with the MRS instruction.

See [MSR](#) on page 3-164

Restrictions

Rd must not be SP and must not be PC.

Condition flags

This instruction does not change the flags.

Examples

`MRS R0, PRIMASK; Read PRIMASK value and write it to R0`

MSR

Move the contents of a general-purpose register into the specified special register.

Syntax

`MSR{cond} spec_reg, Rn` where: *cond* is an optional condition code, see [Conditional execution](#).

Rn is the source register.

spec_reg can be any of: APSR_nzcvq, APSR_g, APSR_nzcvqg, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK, or CONTROL.

Note

You can use APSR to refer to APSR_nzcvq.

Operation

The register access operation in MSR depends on the privilege level. Unprivileged software can only access the APSR, see Table 4 on page 5. Privileged software can access all special registers.

In unprivileged software writes to unallocated or execution state bits in the PSR are ignored.

Note

When you write to BASEPRI_MAX, the instruction writes to BASEPRI only if either:

- *Rn* is non-zero and the current BASEPRI value is 0
- *Rn* is non-zero and less than the current BASEPRI value.

See [MRS](#).

Restrictions

Rn must not be SP and must not be PC.

Condition flags

This instruction updates the flags explicitly based on the value in Rn .

Examples

`MSR CONTROL, R1; Read R1 value and write it to the CONTROL register`

NOP

No Operation.

Syntax

`NOP{cond}` where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

NOP does nothing. NOP is not necessarily a time-consuming NOP. The processor might remove it from the pipeline before it reaches the execution stage.

Use NOP for padding, for example to place the following instruction on a 64-bit boundary.

Condition flags

This instruction does not change the flags.

Examples

`NOP; No operation`

SEV

Send Event

Syntax

`SEV{cond}` where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

SEV is a hint instruction that causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register to 1, see [Power management](#).

Condition flags

This instruction does not change the flags.

Examples

`SEV; Send Event`

SVC

Supervisor call

Syntax

`SVC{cond} #imm` where: *cond* is an optional condition code, see [Conditional execution](#). *imm* is an expression evaluating to an integer in the range 0-255 (8-bit value).

Operation

The SVC instruction causes the SVC exception.

imm is ignored by the processor. If required, it can be retrieved by the exception handler to determine what service is being requested.

Condition flags

This instruction does not change the flags.

Examples

| | | |
|-----|-------|--|
| SVC | #0x32 | ; Supervisor Call (SVCall handler can extract the immediate value by locating it through the stacked PC) |
|-----|-------|--|

WFE

Wait For Event.

Syntax

WFE{cond} where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

WFE is a hint instruction.

If the event register is 0, WFE suspends execution until one of the following events occurs:

- an exception, unless masked by the exception mask registers or the current priority level
- an exception enters the Pending state, if SEVONPEND in the System Control Register is set
- a Debug Entry request, if Debug is enabled
- an event signaled by a peripheral or another processor in a multiprocessor system using the SEV instruction.

If the event register is 1, WFE clears it to 0 and returns immediately.

For more information see [Power management](#).

Condition flags

This instruction does not change the flags.

Examples

| |
|---------------------|
| WFE; Wait for event |
|---------------------|

WFI

Wait for Interrupt.

Syntax

WFI{cond} where: *cond* is an optional condition code, see [Conditional execution](#).

Operation

WFI is a hint instruction that suspends execution until one of the following events occurs:

- a non-masked interrupt occurs and is taken •an interrupt masked by PRIMASK becomes pending
- a Debug Entry request.

Condition flags

This instruction does not change the flags.

Examples

| |
|-------------------------|
| WFI; Wait for interrupt |
|-------------------------|

; const-struc is an expression evaluating ; to a constant in the range 0-4095.

SDIV

op{cond} {Rd,} Rn, #imm12; ADD and SUB only

Send Event

1.4 Cortex M4 Peripherals

The address map of the *Private peripheral bus* (PPB) is:

| Address | Core peripheral | Description |
|-----------------------|--------------------------------------|--|
| 0xE000E008-0xE000E00F | System control block | Table 50 |
| 0xE000E010-0xE000E01F | System timer | Table 71 |
| 0xE000E100-0xE000E4EF | Nested Vectored Interrupt Controller | Table 40 |
| 0xE000ED00-0xE000ED3F | System control Fblock | Table 50 |
| 0xE000ED90-0xE000ED93 | MPU Type Register | Reads as zero, indicating no MPU is implemented ^a |
| 0xE000ED90-0xE000EDB8 | Memory protection unit | Table 76 |
| 0xE000EF00-0xE000EF03 | Nested Vectored Interrupt Controller | Table 40 |
| 0xE000EF30-0xE000EF44 | Floating Point Unit | Table 90 |

43 . Core peripheral register regions

Software can read the MPU Type Register at 0xE000ED90 to test for the presence of a *memory protection unit* (MPU).

In register descriptions:

- the register *type* is described as follows:

RW Read and write.

RO Read-only.

WO Write-only.

- the *required privilege* gives the privilege level required to access the register, as follows:

Privileged

Only privileged software can access the register.

Unprivileged

Both unprivileged and privileged software can access the register.

1.4.1 Nested Vectored Interrupt Controller

This section describes the NVIC and the registers it uses. The NVIC supports:

- 99 interrupts.
- A programmable priority level of 0-63 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external *Non-maskable interrupt* (NMI)
- Optional WIC, providing ultra-low power sleep mode support.

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling. The hardware implementation of the NVIC registers is:

| Address | Name | Type | Required privilege | Reset value | Description |
|-------------------------|-----------------------|------|---------------------------|-------------|---------------------------------------|
| 0xE000_E100-0xE000_E10C | NVIC_ISER0-NVIC_ISER3 | RW | Privileged | 0x0000_0000 | <i>Interrupt Set-enable Registers</i> |
| 0XE000_E180-0xE000_E18C | NVIC_ICER0-NVIC_ICER3 | RW | Privileged | 0x0000_0000 | Interrupt Clear-enable Registers |
| 0XE000_E200-0xE000_E20C | NVIC_ISPR0-NVIC_ISPR3 | RW | Privileged | 0x0000_0000 | Interrupt Set-pending Registers |
| 0XE000_E280-0xE000_E28C | NVIC_ICPR0-NVIC_ICPR3 | RW | Privileged | 0x0000_0000 | Interrupt Clear-pending Registers |
| 0xE000_E300-0xE000_E30C | NVIC_IABR0-NVIC_IABR3 | RW | Privileged | 0x0000_0000 | Interrupt Active Bit Registers |
| 0xE000_E400-0xE000_E464 | NVIC_IPR0-NVIC_IPR24 | RW | Privileged | 0x0000_0000 | Interrupt Priority Registers |
| 0xE000_EF00 | STIR | WO | Configurable ^a | 0x0000_0000 | Software Trigger Interrupt Register |

44 . NVIC register summary

See the register description for more information.

Accessing the Cortex-M4 NVIC registers using CMSIS

CMSIS functions enable software portability between different Cortex-M profile processors. To access the NVIC registers when using CMSIS, use the following functions:

| CMSIS function | Description |
|--|--|
| void NVIC_EnableIRQ(IRQn_Type IRQn) ^a | Enables an interrupt or exception. |
| void NVIC_DisableIRQ(IRQn_Type IRQn) ^a | Disables an interrupt or exception. |
| void NVIC_SetPendingIRQ(IRQn_Type IRQn) ^a | Sets the pending status of interrupt or exception to 1. |
| void NVIC_ClearPendingIRQ(IRQn_Type IRQn) ^a | Clears the pending status of interrupt or exception to 0. |
| uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn) ^a | Reads the pending status of interrupt or exception. This function returns non-zero value if the pending status is set to 1. |

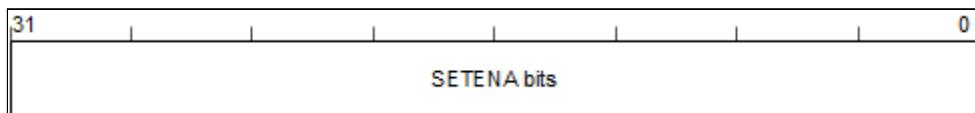
| CMSIS function | Description |
|---|---|
| void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority) ^a | Sets the priority of an interrupt or exception with configurable priority level to 1. |
| uint32_t NVIC_GetPriority(IRQn_Type IRQn) ^a | Reads the priority of an interrupt or exception with configurable priority level. This function return the current priority level. |

45 .CMSIS access NVIC functions

The input parameter IRQn is the IRQ number, see Table 16 for more information.

Interrupt Set-enable Registers

The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled. See the register summary in Table 40 for the register attributes.
The bit assignments are:



| Bits | Name | Function |
|--------|--------|---|
| [31:0] | SETENA | Interrupt set-enable bits. Write: 1. = no effect 2. = enable interrupt. Read: 3. = interrupt disabled 4. = interrupt enabled. |

46 .ISER bit assignments

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

Interrupt Clear-enable Registers

The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. See the register summary in Table 40 for the register attributes.
The bit assignments are:

31 | | | | | | | | 0
CLRENA bits

| Bits | Name | Function |
|--------|--------|---|
| [31:0] | CLRENA | <p>Interrupt clear-enable bits.</p> <p>Write:</p> <ul style="list-style-type: none"> 1. = no effect 2. = disable interrupt. <p>Read:</p> <ul style="list-style-type: none"> 3. = interrupt disabled 4. = interrupt enabled. |

47 . ICER bit assignments

Interrupt Set-pending Registers

The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. See the register summary in Table 40 for the register attributes.

The bit assignments are:

| Bits | Name | Function |
|--------|---------|---|
| [31:0] | SETPEND | <p>Interrupt set-pending bits.</p> <p>Write:</p> <ol style="list-style-type: none"> 1. = no effect 2. = changes interrupt state to pending. <p>Read:</p> <p>0 = interrupt is not pending 1 = interrupt is pending</p> |

48 . ISPR bit assignments



Note

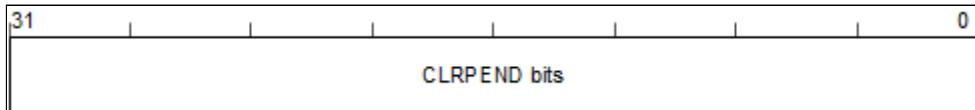
Writing 1 to the ISPR bit corresponding to:

- an interrupt that is pending has no effect
 - a disabled interrupt sets the state of that interrupt to pending.

Interrupt Clear-pending Registers

The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. See the register summary in Table 40 for the register attributes.

The bit assignments are:



| Bits | Name | Function |
|--------|---------|--|
| [31:0] | CLRPEND | Interrupt clear-pending bits. Write: 1. = no effect 2. = removes pending state an interrupt. Read: 0 = interrupt is not pending 1 = interrupt is pending. |

49 .ICPR bit assignments



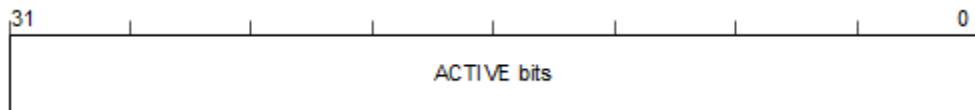
Note

Writing 1 to an ICPR bit does not affect the active state of the corresponding interrupt.

Interrupt Active Bit Registers

The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. See the register summary in Table 40 for the register attributes³

The bit assignments are:



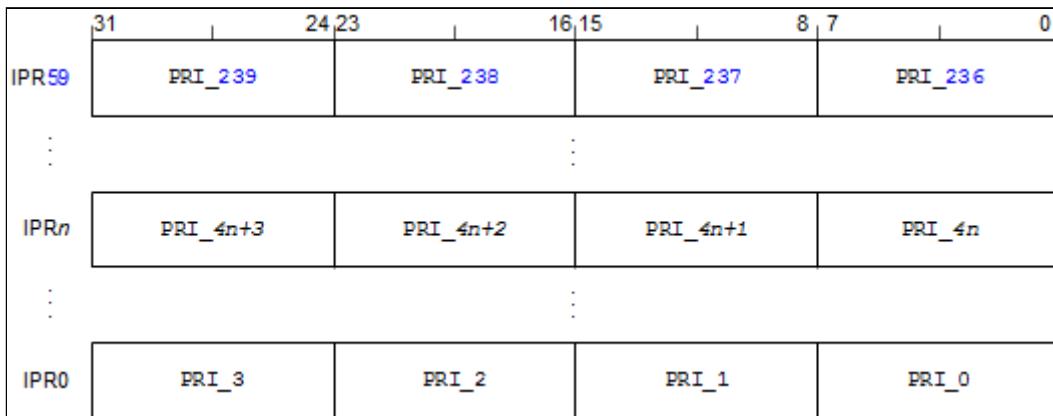
| Bits | Name | Function |
|--------|--------|--|
| [31:0] | ACTIVE | Interrupt active flags: 0 = interrupt not active 1 = interrupt active. |

50 .IABR bit assignments

A bit reads as one if the status of the corresponding interrupt is active or active and pending.

Interrupt Priority Register

The NVIC_IPR0-NVIC_IPR24 registers provide an 6-bit priority field for each interrupt. These registers are byte-accessible. See the register summary in Table 40 for their attributes. Each register holds four priority fields as shown:



| Bits | Name | Function |
|---------|-------------------------|---|
| [31:24] | Priority, byte offset 3 | Each priority field holds a priority value, 0-255. The lower the value, the greater the priority of the corresponding interrupt. The processor implements only bits[7:2] of each field, bits[1:0] read as zero and ignore writes. |
| [23:16] | Priority, byte offset 2 | |
| [15:8] | Priority, byte offset 1 | |
| [7:0] | Priority, byte offset 0 | |

51 . IPR bit assignments

See [Accessing the Cortex-M4 NVIC registers using CMSIS](#) for more information about the access to the interrupt priority array, which provides the software view of the interrupt priorities.
Find the IPR number and byte offset for interrupt m as follows:

- the corresponding IPR number, see Table 4-8 n is given by $n = m \text{ DIV } 4$
- the byte offset of the required Priority field in this register is $m \text{ MOD } 4$, where:

—byte offset 0 refers to register bits[7:0]
—byte offset 1 refers to register bits[15:8] —byte offset 2 refers to register bits[23:16]
—byte offset 3 refers to register bits[31:24].

Software Trigger Interrupt Register

Write to the STIR to generate an interrupt from software. See the register summary in Table 40 for the STIR attributes.

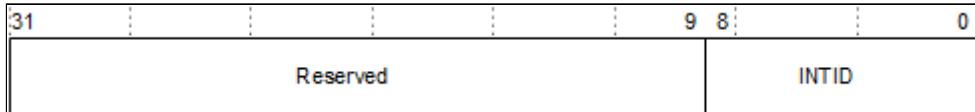
When the USERSETMPEND bit in the SCR is set to 1, unprivileged software can access the STIR, see *System Control Register* on page 4-19.



Note

Only privileged software can enable unprivileged access to the STIR.

The bit assignments are:



| Bits | Field | Function |
|--------|-------|--|
| [31:9] | - | Reserved. |
| [8:0] | INTID | Interrupt ID of the interrupt to trigger, in the range 0-239. For example, a value of 0x03 specifies interrupt IRQ3. |

52 . STIR bit assignments

Level-sensitive and pulse interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt, see [Hardware and software control of interrupts](#). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. This means that the peripheral can hold the interrupt signal asserted until it no longer requires servicing. In 9116, all interrupts are level sensitive.

Hardware and software control of interrupts

The Cortex-M4 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- the NVIC detects that the interrupt signal is HIGH and the interrupt is not active
- the NVIC detects a rising edge on the interrupt signal
- software writes to the corresponding interrupt set-pending register bit, see [Interrupt Set-pending Registers](#), or to the STIR to make an interrupt pending, see [Software Trigger Interrupt Register](#).

A pending interrupt remains pending until one of the following:

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

—For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC 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 NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.

If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.

- Software writes to the corresponding interrupt clear-pending register bit.

For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.

For a pulse interrupt, state of the interrupt changes to:
—inactive, if the state was pending
—active, if the state was active and pending.

NVIC design hints and tips

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers. See the individual register descriptions for the supported access sizes.

A interrupt can enter pending state even if it is disabled. Disabling an interrupt only prevents the processor from taking that interrupt.

Before programming VTOR to relocate the vector table, ensure the vector table entries of the new vector table are setup for fault handlers, NMI and all enabled exception like interrupts. For more information see [Vector Table Offset Register](#).

NVIC programming hints

Software uses the CPSIE I and CPSID I instructions to enable and disable interrupts. The CMSIS provides the following intrinsic functions for these instructions:

```
void __disable_irq(void) // Disable Interrupts void __enable_irq(void) // Enable Interrupts
```

In addition, the CMSIS provides a number of functions for NVIC control, including:

| CMSIS interrupt control function | Description |
|---|---|
| void NVIC_SetPriorityGrouping(uint32_t priority_grouping) | Set the priority grouping |
| void NVIC_EnableIRQ(IRQn_t IRQn) | Enable IRQn |
| void NVIC_DisableIRQ(IRQn_t IRQn) | Disable IRQn |
| uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn) | Return true (IRQ-Number) if IRQn is pending |
| void NVIC_SetPendingIRQ (IRQn_t IRQn) | Set IRQn pending |
| void NVIC_ClearPendingIRQ (IRQn_t IRQn) | Clear IRQn pending status |
| uint32_t NVIC_GetActive (IRQn_t IRQn) | Return the IRQ number of the active interrupt |
| void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority) | Set priority for IRQn |
| uint32_t NVIC_GetPriority (IRQn_t IRQn) | Read priority of IRQn |
| void NVIC_SystemReset (void) | Reset the system |

53 .CMSIS functions for NVIC control

The input parameter IRQn is the IRQ number, see Table 16. For more information about these functions see the CMSIS documentation.

1.4.2 System control block

The *System control block* (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. The system control block registers are:

| Address | Name | Type | Required privilege | Reset value | Description |
|-------------|-------------------|-----------------|--------------------|-------------|---|
| 0xE000_E008 | ACTLR | RW | Privileged | 0x0000_0000 | <i>Auxiliary Control Register</i> |
| 0xE000_ED00 | CPUID | RO | Privileged | 0x410F_C241 | <i>CPUID Base Register</i> |
| 0xE000_ED04 | ICSR | RW ^a | Privileged | 0x0000_0000 | <i>Interrupt Control and State Register</i> |
| 0xE000_ED08 | VTOR | RW | Privileged | 0x0000_0000 | <i>Vector Table Offset Register</i> |
| 0xE000_ED0C | AIRCR | RW ^a | Privileged | 0xFA05_0000 | <i>Application Interrupt and Reset Control Register</i> |
| 0xE000_ED10 | SCR | RW | Privileged | 0x0000_0000 | <i>System Control Register</i> |
| 0xE000_ED14 | CCR | RW | Privileged | 0x0000_0200 | <i>Configuration and Control Register</i> |
| 0xE000_ED18 | SHPR1 | RW | Privileged | 0x0000_0000 | <i>System Handler Priority Register 1</i> |
| 0xE000_ED1C | SHPR2 | RW | Privileged | 0x0000_0000 | <i>System Handler Priority Register 2</i> |
| 0xE000_ED20 | SHPR3 | RW | Privileged | 0x0000_0000 | <i>System Handler Priority Register 3</i> |
| 0xE000_ED24 | SHCRS | RW | Privileged | 0x0000_0000 | <i>System Handler Control and State Register</i> |
| 0xE000_ED28 | CFSR | RW | Privileged | 0x0000_0000 | <i>Configurable Fault Status Register</i> |
| 0xE000_ED28 | MMSR ^b | RW | Privileged | 0x0000_0000 | <i>MemManage Fault Status Register</i> |
| 0xE000_ED29 | BFSR ^b | RW | Privileged | 0x0000_0000 | <i>BusFault Status Register</i> |
| 0xE000_ED2A | UFSR ^b | RW | Privileged | 0x0000_0000 | <i>UsageFault Status Register</i> |
| 0xE000_ED2C | HFSR | RW | Privileged | 0x0000_0000 | <i>HardFault Status Register</i> |
| 0xE000_ED34 | MMAR | RW | Privileged | 0x0000_83AC | <i>MemManage Fault Address Register</i> |

54 . Summary of the system control block registers

| Address | Name | Type | Required privilege | Reset value | Description |
|-------------|------|------|--------------------|-------------|--|
| 0xE000_ED38 | BFAR | RW | Privileged | 0x0000_83AC | <i>BusFault Address Register</i> |
| 0xE000_ED3C | AFSR | RW | Privileged | 0x0000_0000 | <i>Auxiliary Fault Status Register</i> |

55 . Summary of the system control block registers (continued)

See the register description for more information.

A subregister of the CFSR.

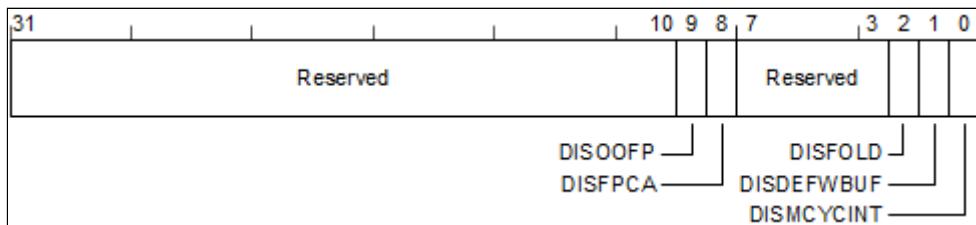
Auxiliary Control Register

The ACTLR provides disable bits for the following processor functions:

- IT folding
- write buffer use for accesses to the default memory map
- interruption of multi-cycle instructions.

By default this register is set to provide optimum performance from the Cortex-M4 processor, and does not normally require modification.

See the register summary in Table 52 for the ACTLR attributes. The bit assignments are:



| Bits | Name | Function |
|---------|------------|--|
| [31:10] | - | Reserved |
| [9] | DISOOFP | Disables floating point instructions completing out of order with respect to integer instructions. |
| [8] | DISFPCA | Disable automatic update of CONTROL.FPCA. |
| [7:3] | - | Reserved |
| [2] | DISFOLD | When set to 1, disables IT folding. see <i>About IT folding</i> for more information. |
| [1] | DISDEFWBUF | When set to 1, disables write buffer use during default memory map accesses. This causes all BusFaults to be precise BusFaults but decreases performance because any store to memory must complete before the processor can execute the next instruction. *Note*This bit only affects write buffers implemented in the Cortex-M4 processor. |
| [0] | DISMCYCINT | When set to 1, disables interruption of load multiple and store multiple instructions. This increases the interrupt latency of the processor because any LDM or STM must complete before the processor can stack the current state and enter the interrupt handler. |

56 .ACTLR bit assignments

About IT folding

In some situations, the processor can start executing the first instruction in an IT block while it is still executing the IT instruction. This behavior is called IT folding, and improves performance. However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit to 1 before executing the task, to disable IT folding.

CPUID Base Register

The CPUID register contains the processor part number, version, and implementation information. See the register summary in Table 53 for its attributes. The bit assignments are:

| | | | | | | | | | |
|-------------|---------|----------|-------|-------|--------|--|----------|-----|---|
| 31 | | 24 23 | 20 19 | 16 15 | | | | 4 3 | 0 |
| Implementer | Variant | Constant | | | PartNo | | Revision | | |

| Bits | Name | Function |
|-------------|-------------|--|
| [31:24] | Implementer | Implementer code: 0x41 = ARM |
| [23:20] | Variant | Variant number, the r value in the $rnpn$ product revision identifier: 0x0 = Revision 0 |
| [19:16] | Constant | Reads as 0xF |
| [15:4] | PartNo | Part number of the processor: 0xC24 = Cortex-M4 |
| [3:0] | Revision | Revision number, the p value in the $rnpn$ product revision identifier: 0x0 = Patch 0 |

57 .CPUID register bit assignments

Interrupt Control and State Register

The ICSR:

- provides:

—a set-pending bit for the *Non-Maskable Interrupt* (NMI) exception —set-pending and clear-pending bits for the PendSV and SysTick exceptions

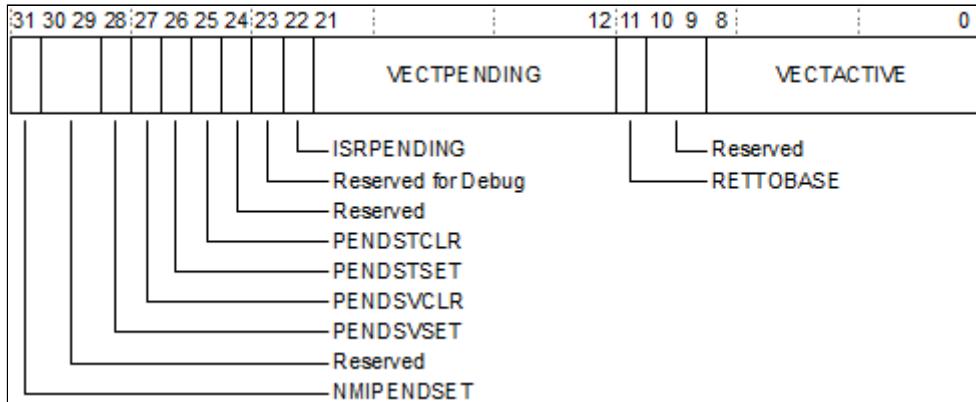
- indicates:

—the exception number of the exception being processed

—whether there are preempted active exceptions

—the exception number of the highest priority pending exception —whether any interrupts are pending.

See the register summary in Table 4-12, and the Type descriptions in Table 54, for the ICSR attributes. The bit assignments are:



| Bits | Name | Type | Function |
|---------|------------|------|---|
| [31] | NMIPENDSET | RW | <p>NMI set-pending bit. Write:</p> <ul style="list-style-type: none"> 1. = no effect 2. = changes NMI exception state to pending. <p>Read: 0 = NMI exception is not pending 1 = NMI exception is pending. Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it registers a write of 1 to this bit, and entering the handler clears this bit to 0. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.</p> |
| [30:29] | - | - | Reserved. |
| [28] | PENDSVSET | RW | <p>PendSV set-pending bit. Write:</p> <ul style="list-style-type: none"> 1. = no effect 2. = changes PendSV exception state to pending. <p>Read: 0 = PendSV exception is not pending 1 = PendSV exception is pending. Writing 1 to this bit is the only way to set the PendSV exception state to pending.</p> |
| [27] | PENDSVCLR | WO | <p>PendSV clear-pending bit. Write:</p> <ul style="list-style-type: none"> 1. = no effect 2. = removes the pending state from the PendSV exception. |
| [26] | PENDSTSET | RW | <p>SysTick exception set-pending bit. Write:</p> <ul style="list-style-type: none"> 1. = no effect 2. = changes SysTick exception state to pending. <p>Read: 0 = SysTick exception is not pending 1 = SysTick exception is pending.</p> |

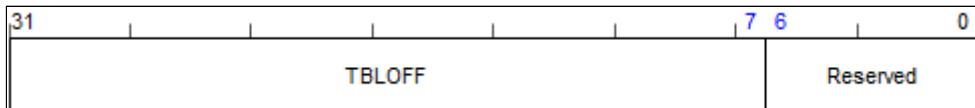
58 . ICSR bit assignments

When you write to the ICSR, the effect is Unpredictable if you:

- write 1 to the PENDSVSET bit and write 1 to the PENDSVCLR bit
- write 1 to the PENDSTSET bit and write 1 to the PENDSTCLR bit.

Vector Table Offset Register

The VTOR indicates the offset of the vector table base address from memory address 0x00000000. See the register summary in Table 51 for its attributes. The bit assignments are:



See the configurable information that follows the register description table below for information about the configuration of the boundary between TBLOFF field and the [6:0] field.

| Bits | Name | Function |
|--------|--------|--|
| [31:7] | TBLOFF | <p>Vector table base offset field. It contains bits[29:7] of the offset of the table base from the bottom of the memory map. See the configurable information that follow this table for information about the configuration of this field and the [6:0] field.</p> <p>Note</p> <p>B it[29] determines whether the vector table is in the code or SRAM memory region:</p> <ul style="list-style-type: none"> • 0 = code • 1 = SRAM. <p>Bit[29] is sometimes called the TBLBASE bit.</p> |
| [6:0] | - | Reserved. |

59 .VTOR bit assignments

When setting TBLOFF, you must align the offset to the number of exception entries in the vector table. The default programming done for TBLOFF is 0x0 if the Vector table is located in the RAM and it is 0x08012000 if the Vector table is located in the Flash. If the Vector table has to located in a custom location, ensure that the TBLOFF is aligned on a 128-word boundary.



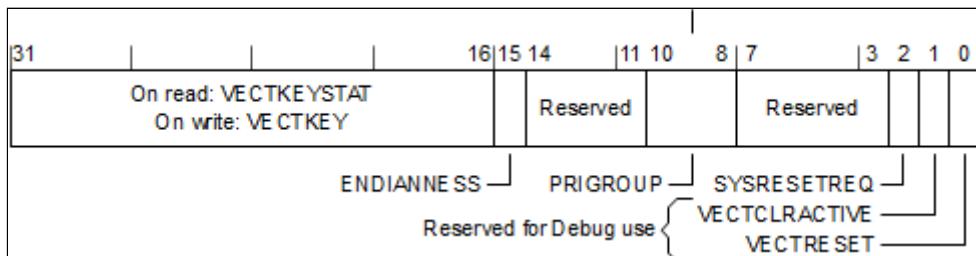
Note

Table alignment requirements mean that bits[6:0] of the table offset are always zero.

Application Interrupt and Reset Control Register

The AIRCR provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. See the register summary in Table 51 and Table 56 for its attributes.

To write to this register, you must write 0x5FA to the VECTKEY field, otherwise the processor ignores the write. The bit assignments are:



| Bits | Name | Type | Function |
|---------|-------------------------------------|------|--|
| [31:16] | Write: VECTKEYSTAT Read: VECTKEY | RW | Register key: Reads as 0xFA05 On writes, write 0x5FA to VECTKEY, otherwise the write is ignored. |
| [15] | ENDIANNESS | RO | Data endianness bit: 0 = Little-endian 1 = Big-endian. |
| [14:11] | - | - | Reserved |
| [10:8] | PRIGROUP | R/W | <ul style="list-style-type: none"> Interrupt priority grouping field. This field determines the split of group priority from subpriority, see Binary point. |
| [7:3] | | - | Reserved. |
| [2] | SYSRESETREQ | WO | <p>System reset request:</p> <ol style="list-style-type: none"> = no system reset request = asserts a signal to the outer system that requests a reset. <p>This is intended to force a large system reset of all major components except for debug. This bit reads as 0.</p> |
| [1] | VECTCLRACTIVE | WO | Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable. |
| [0] | VECTRESET | WO | Reserved for Debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable. |

60 .AIRCR bit assignments

Binary point

The PRIGROUP field indicates the position of the binary point that splits the PRI_n fields in the Interrupt Priority Registers into separate *group priority* and *subpriority* fields. Table 57 shows how the PRIGROUP value controls this split. MCU supports 64 interrupt priorities levels.

| | Interrupt priority level value, PRI_M[7:0] | | | Number of Group priorities | Subpriorities |
|----------|--|---------------------|------------------|----------------------------|---------------|
| PRIGROUP | Binary point ^a | Group priority bits | Subpriority bits | | |
| 0b000 | bxxxxxx.y | [7:1] | [0] | 128 | 2 |
| 0b001 | bxxxxx.yy | [7:2] | [1:0] | 64 | 4 |
| 0b010 | bxxxx.yyy | [7:3] | [2:0] | 32 | 8 |
| 0b011 | bxxxx.yyyy | [7:4] | [3:0] | 16 | 16 |

| | Interrupt priority level value, PRI_M[7:0] | | | Number of Group priorities | Subpriorities |
|----------|---|----------------------------|-------------------------|-----------------------------------|----------------------|
| PRIGROUP | Binary point^a | Group priority bits | Subpriority bits | | |
| 0b100 | bxxx.yyyyy | [7:5] | [4:0] | 8 | 32 |
| 0b101 | bxx.yyyyy | [7:6] | [5:0] | 4 | 64 |

61 . Priority grouping

PRI_n[7:0] field showing the binary point. x denotes a group priority field bit, and y denotes a subpriority field bit.

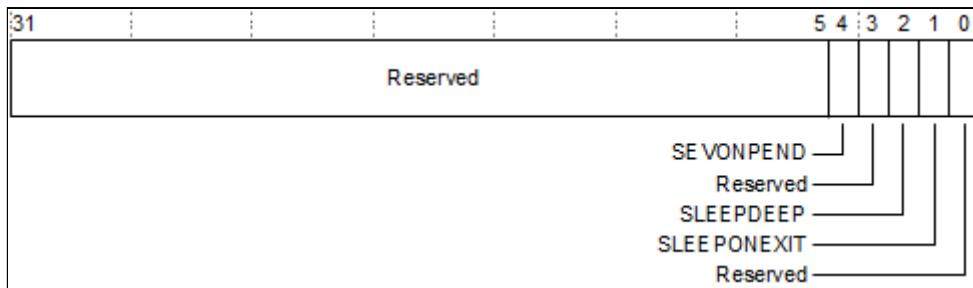


Note

Determining preemption of an exception uses only the group priority field, see [Interrupt priority grouping](#).

System Control Register

The SCR controls features of entry to and exit from low power state. See the register summary in Table 58 for its attributes. The bit assignments are:



| Bits | Name | Function |
|--------|------|-----------|
| [31:5] | - | Reserved. |

| Bits | Name | Function |
|------|-------------|---|
| [4] | SEVONPEND | <p>Send Event on Pending bit:</p> <ol style="list-style-type: none"> 1. = only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded 2. = enabled events and all interrupts, including disabled interrupts, can wakeup the processor. <p>When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE.</p> <p>The processor also wakes up on execution of an SEV instruction or an external event.</p> |
| [3] | - | Reserved. |
| [2] | SLEEPDEEP | Controls whether the processor uses sleep or deep sleep as its low power mode: 0 = sleep 1 = deep sleep. |
| [1] | SLEEPONEXIT | Indicates sleep-on-exit when returning from Handler mode to Thread mode: <ol style="list-style-type: none"> 1. = do not sleep when returning to Thread mode. 2. = enter sleep, or deep sleep, on return from an ISR. <p>Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.</p> |
| [0] | - | Reserved. |

62 . SCR bit assignments

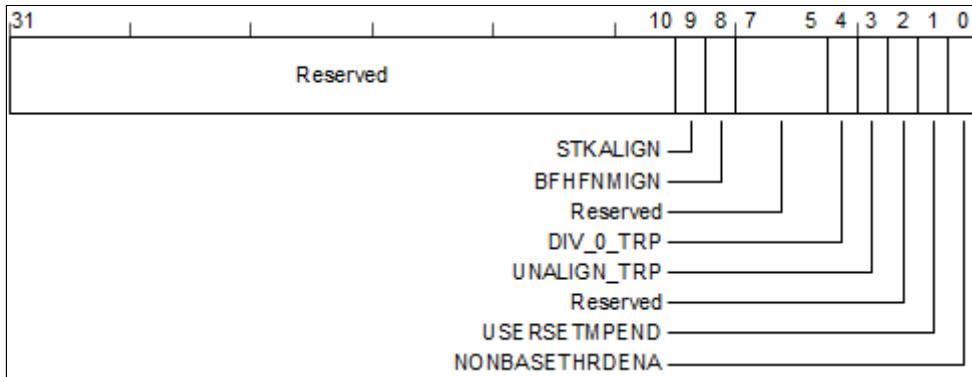
Configuration and Control Register

The CCR controls entry to Thread mode and enables:

- the handlers for NMI, hard fault and faults escalated by FAULTMASK to ignore BusFaults
- trapping of divide by zero and unaligned accesses
- access to the STIR by unprivileged software, see [Software Trigger Interrupt Register](#).

See the register summary in Table 59 for the CCR attributes.

The bit assignments are:



| Bits | Name | Function |
|---------|--------------|--|
| [31:10] | - | Reserved. |
| [9] | STKALIGN | Indicates stack alignment on exception entry: 1. = 4-byte aligned 2. = 8-byte aligned. On exception entry, the processor uses bit[9] of the stacked PSR to indicate the stack alignment. On return from the exception it uses this stacked bit to restore the correct stack alignment. |
| [8] | BFHFNIGN | Enables handlers with priority -1 or -2 to ignore data BusFaults caused by load and store instructions. This applies to the hard fault, NMI, and FAULTMASK escalated handlers: 1. = data bus faults caused by load and store instructions cause a lock-up 2. = handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions. Set this bit to 1 only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them. |
| [7:5] | - | Reserved. |
| [4] | DIV_0_TRP | Enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0: 0 = do not trap divide by 0 1 = trap divide by 0. When this bit is set to 0, a divide by zero returns a quotient of 0. |
| [3] | UNALIGN_TRP | Enables unaligned access traps: 0 = do not trap unaligned halfword and word accesses 1 = trap unaligned halfword and word accesses. If this bit is set to 1, an unaligned access generates a UsageFault. Unaligned LDM, STM, LDRD, and STRD instructions always fault irrespective of whether UNALIGN_TRP is set to 1. |
| [2] | - | Reserved. |
| [1] | USERSETPEND | Enables unprivileged software access to the STIR, see <i>Software Trigger Interrupt Register</i> on page 4-8: 1. = disable 2. = enable. |
| [0] | NONBASETHRDA | Indicates how the processor enters Thread mode: 1. = processor can enter Thread mode only when no exception is active. 2. = processor can enter Thread mode from any level under the control of an EXC_RETURN value, see Exception return . |

63 . CCR bit assignments

System Handler Priority Registers

The SHPR1-SHPR3 registers set the priority level, 0 to 63 of the exception handlers that have configurable priority. SHPR1-SHPR3 are byte accessible. See the register summary in Table 60 for their attributes.

The system fault handlers and the priority field and register for each handler are:

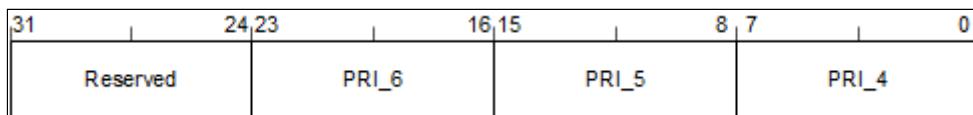
Table 60 System fault handler priority fields

HandlerFieldRegister description

MemManagePRI_4_System Handler Priority Register 1_on
page4-22BusFaultPRI_5UsageFaultPRI_6SVCallPRI_11_System Handler Priority Register 2_on
page4-22PendSVPRI_14_System Handler Priority Register 2_on page4-22SysTickPRI_15

Each PRI_N field is 8 bits wide, but the processor implements only bits[7:2] of each field, and bits[1:0] read as zero and ignore writes.

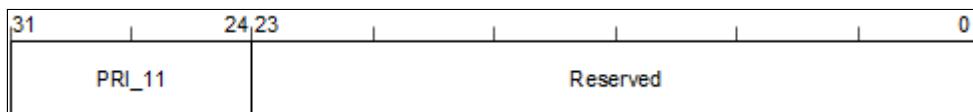
System Handler Priority Register 1 The bit assignments are:



| Bits | Name | Function |
|---------|-------|--|
| [31:24] | PRI_7 | Reserved |
| [23:16] | PRI_6 | Priority of system handler 6, UsageFault |
| [15:8] | PRI_5 | Priority of system handler 5, BusFault |
| [7:0] | PRI_4 | Priority of system handler 4, MemManage |

64 . SHPR1 register bit assignments

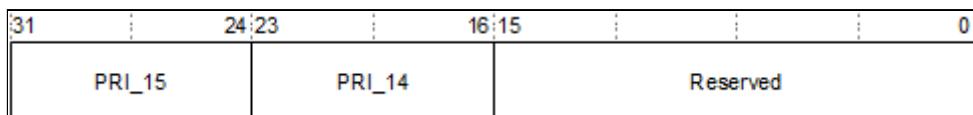
System Handler Priority Register 2 The bit assignments are:



| Bits | Name | Function |
|---------|--------|---------------------------------------|
| [31:24] | PRI_11 | Priority of system handler 11, SVCall |
| [23:0] | - | Reserved |

65 . SHPR2 register bit assignments

System Handler Priority Register 3 The bit assignments are:



| Bits | Name | Function |
|---------|--------|--|
| [31:24] | PRI_15 | Priority of system handler 15, SysTick exception |
| [23:16] | PRI_14 | Priority of system handler 14, PendSV |
| [15:0] | - | Reserved |

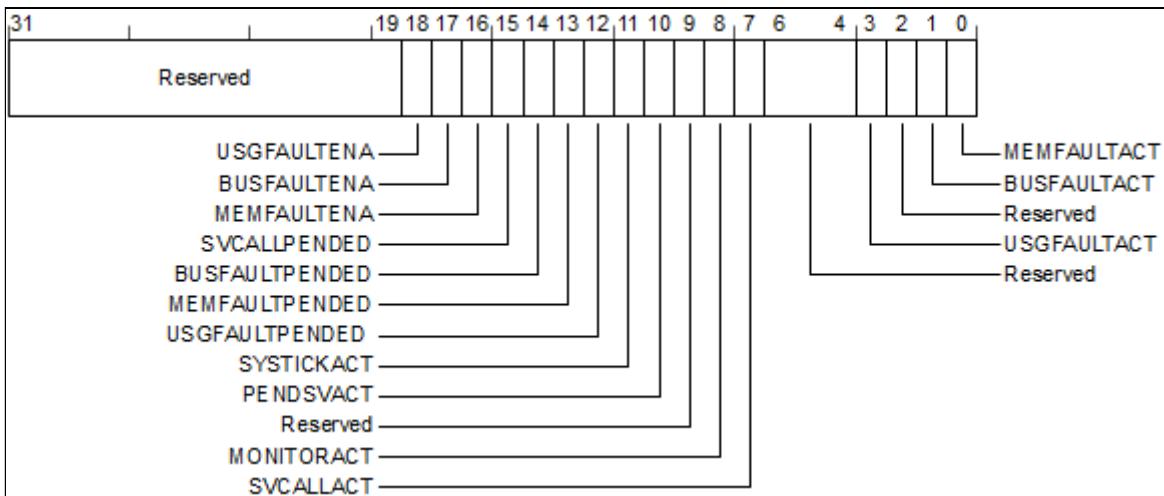
66 . SHPR3 register bit assignments

System Handler Control and State Register

The SHCSR enables the system handlers, and indicates:

- the pending status of the BusFault, MemManage fault, and SVC exceptions
- the active status of the system handlers.

See the register summary in Table 4-12 for the SHCSR attributes. The bit assignments are:



| Bits | Name | Function |
|---------|----------------|---|
| [31:19] | - | Reserved |
| [18] | USGFAULTENA | UsageFault enable bit, set to 1 to enable ^a |
| [17] | BUSFAULTENA | BusFault enable bit, set to 1 to enable ^a |
| [16] | MEMFAULTENA | MemManage enable bit, set to 1 to enable ^a |
| [15] | SVCALLPENDED | SVCall pending bit, reads as 1 if exception is pending ^b |
| [14] | BUSFAULTPENDED | BusFault exception pending bit, reads as 1 if exception is pending ^b |
| [13] | MEMFAULTPENDED | MemManage exception pending bit, reads as 1 if exception is pending ^b |
| [12] | USGFAULTPENDED | UsageFault exception pending bit, reads as 1 if exception is pending ^b |
| [11] | SYSTICKACT | SysTick exception active bit, reads as 1 if exception is active ^c |

| Bits | Name | Function |
|-------|-------------|--|
| [10] | PENDSVACT | PendSV exception active bit, reads as 1 if exception is active |
| [9] | - | Reserved |
| [8] | MONITORACT | Debug monitor active bit, reads as 1 if Debug monitor is active |
| [7] | SVCALLACT | SVCall active bit, reads as 1 if SVC call is active |
| [6:4] | - | Reserved |
| [3] | USGFAULTACT | UsageFault exception active bit, reads as 1 if exception is active |
| [2] | - | Reserved |
| [1] | BUSFAULTACT | BusFault exception active bit, reads as 1 if exception is active |
| [0] | MEMFAULTACT | MemManage exception active bit, reads as 1 if exception is active |

67 . SHCSR bit assignments

1. Enable bits, set to 1 to enable the exception, or set to 0 to disable the exception.
2. Pending bits, read as 1 if the exception is pending, or as 0 if it is not pending. You can write to these bits to change the pending status of the exceptions.
3. Active bits, read as 1 if the exception is active, or as 0 if it is not active. You can write to these bits to change the active status of the exceptions, but see the Caution in this section.

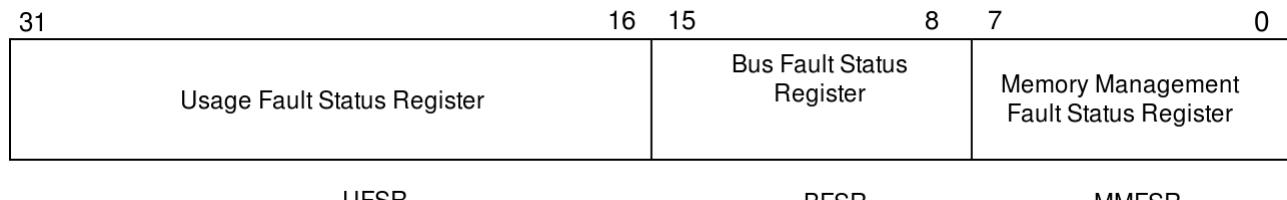
If you disable a system handler and the corresponding fault occurs, the processor treats the fault as a hard fault. You can write to this register to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

Caution

- Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.
- After you have enabled the system handlers, if you have to change the value of a bit in this register you must use a read-modify-write procedure to ensure that you change only the required bit.

Configurable Fault Status Register

The CFSR indicates the cause of a MemManage fault, BusFault, or UsageFault. See the register summary in Table 50 for its attributes. The bit assignments are:



The following subsections describe the subregisters that make up the CFSR:

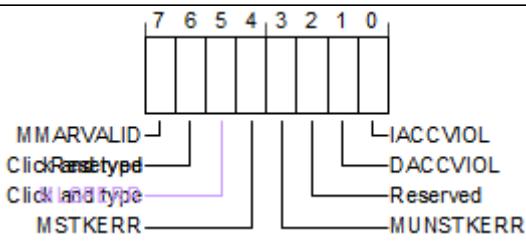
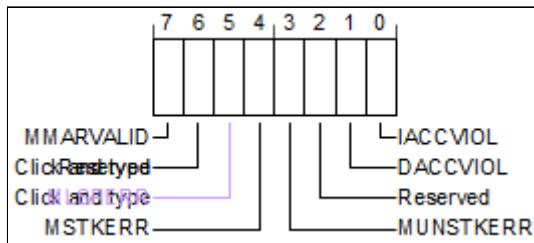
- **MemManage Fault Status Register**
- BusFault Status Register
- UsageFault Status Register

The CFSR is byte accessible. You can access the CFSR or its subregisters as follows:

- access the complete CFSR with a word access to 0xE000ED28
- access the MMFSR with a byte access to 0xE000ED28
- access the MMFSR and BCSR with a halfword access to 0xE000ED28
- access the BCSR with a byte access to 0xE000ED29
- access the UCSR with a halfword access to 0xE000ED2A.

MemManage Fault Status Register

The flags in the MMFSR indicate the cause of memory access faults. The bit assignments are:



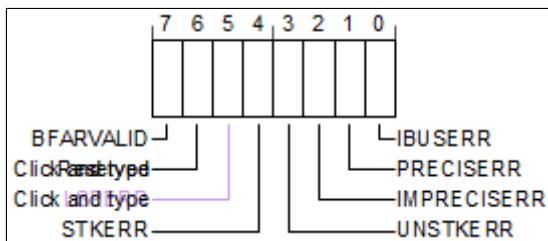
| B it s | Nam e | Function |
|--------------|------------|---|
| [7] | MMARV ALID | MemManage Fault Address Register (MMFAR) valid flag: 0 = value in MMAR is not a valid fault address 1 = MMAR holds a valid fault address. If a MemManage fault occurs and is escalated to a HardFault because of priority, the HardFault handler must set this bit to 0. This prevents problems on return to a stacked active MemManage fault handler whose MMAR value has been overwritten. |
| [6] | - | Reserved. |
| [5] | MLSPE R | 1. = No MemManage fault occurred during floating-point lazy state preservation 2. = A MemManage fault occurred during floating-point lazy state preservation |
| [4] | MSTKERR | MemManage fault on stacking for exception entry: 1. = no stacking fault 2. = stacking for an exception entry has caused one or more access violations. When this bit is 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor has not written a fault address to the MMAR. |
| [3] | MUNST KERR | MemManage fault on unstacking for a return from exception: 1. = no unstacking fault 2. = unstack for an exception return has caused one or more access violations. This fault is chained to the handler. This means that when this bit is 1, the original return stack is still present. The processor has not adjusted the SP from the failing return, and has not performed a new save. The processor has not written a fault address to the MMAR. |
| [2] | - | Reserved |

| Bits | Name | Function |
|------|----------|---|
| [1] | DACCVIOL | Data access violation flag: 1. = no data access violation fault 2. = the processor attempted a load or store at a location that does not permit the operation. When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has loaded the MMAR with the address of the attempted access. |
| [0] | IACCVIOL | Instruction access violation flag: 1. = no instruction access violation fault 2. = the processor attempted an instruction fetch from a location that does not permit execution. This fault occurs on any access to an XN region, even when the MPU is disabled. When this bit is 1, the PC value stacked for the exception return points to the faulting instruction. The processor has not written a fault address to the MMAR. |

68 .MMFSR bit assignments

BusFault Status Register

The flags in the BFSR indicate the cause of a bus access fault. The bit assignments are:



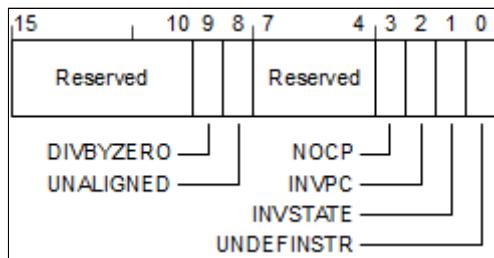
| Bits | Name | Function |
|------|-----------|--|
| [7] | BFARVALID | <i>BusFault Address Register (BFAR) valid flag:</i> 0 = value in BFAR is not a valid fault address 1 = BFAR holds a valid fault address. The processor sets this bit to 1 after a BusFault where the address is known. Other faults can set this bit to 0, such as a MemManage fault occurring later. If a BusFault occurs and is escalated to a hard fault because of priority, the hard fault handler must set this bit to 0. This prevents problems if returning to a stacked active BusFault handler whose BFAR value has been overwritten. |
| [6] | - | Reserved. |
| [5] | LSPERR | 1. = No bus fault occurred during floating-point lazy state preservation. 2. = A bus fault occurred during floating-point lazy state preservation |

| Bits | Name | Function |
|------|-------------|--|
| [4] | STKERR | BusFault on stacking for exception entry: 1. = no stacking fault 2. = stacking for an exception entry has caused one or more BusFaults. When the processor sets this bit to 1, the SP is still adjusted but the values in the context area on the stack might be incorrect. The processor does not write a fault address to the BFAR. |
| [3] | UNSTKERR | BusFault on unstacking for a return from exception: 1. = no unstacking fault 2. = unstack for an exception return has caused one or more BusFaults. This fault is chained to the handler. This means that when the processor sets this bit to 1, the original return stack is still present. The processor does not adjust the SP from the failing return, does not perform a new save, and does not write a fault address to the BFAR. |
| [2] | IMPRECISERR | Imprecise data bus error: 1. = no imprecise data bus error 2. = a data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error. When the processor sets this bit to 1, it does not write a fault address to the BFAR. This is an asynchronous fault. Therefore, if it is detected when the priority of the current process is higher than the BusFault priority, the BusFault becomes pending and becomes active only when the processor returns from all higher priority processes. If a precise fault occurs before the processor enters the handler for the imprecise BusFault, the handler detects both IMPRECISERR set to 1 and one of the precise fault status bits set to 1. |
| [1] | PRECISERR | Precise data bus error: 1. = no precise data bus error 2. = a data bus error has occurred, and the PC value stacked for the exception return points to the instruction that caused the fault. When the processor sets this bit to 1, it writes the faulting address to the BFAR. |
| [0] | IBUSERR | Instruction bus error: 0 = no instruction bus error 1 = instruction bus error. The processor detects the instruction bus error on prefetching an instruction, but it sets the IBUSERR flag to 1 only if it attempts to issue the faulting instruction. When the processor sets this bit to 1, it does not write a fault address to the BFAR. |

69 .BFSR bit assignments

UsageFault Status Register

The UFSR indicates the cause of a UsageFault. The bit assignments are:



| Bits | Name | Function |
|---------|-----------|---|
| [15:10] | - | Reserved. |
| [9] | DIVBYZERO | <p>Divide by zero UsageFault:</p> <ol style="list-style-type: none"> 1. = no divide by zero fault, or divide by zero trapping not enabled 2. = the processor has executed an SDIV or UDIV instruction with a divisor of 0. <p>When the processor sets this bit to 1, the PC value stacked for the exception return points to the instruction that performed the divide by zero. Enable trapping of divide by zero by setting the DIV_0_TRP bit in the CCR to 1, see Configuration and Control Register.</p> |
| [8] | UNALIGNED | <p>Unaligned access UsageFault:</p> <p>0 = no unaligned access fault, or unaligned access trapping not enabled 1 = the processor has made an unaligned memory access. Enable trapping of unaligned accesses by setting the UNALIGN_TRP bit in the CCR to 1, see Configuration and Control Register. Unaligned LDM, STM, LDRD, and STRD instructions always fault irrespective of the setting of UNALIGN_TRP.</p> |
| [7:4] | - | Reserved. |
| [3] | NOCP | No coprocessor UsageFault. The processor does not support coprocessor instructions: 0 = no UsageFault caused by attempting to access a coprocessor 1 = the processor has attempted to access a coprocessor. |
| [2] | INVPC | <p>Invalid PC load UsageFault, caused by an invalid PC load by EXC_RETURN:</p> <ol style="list-style-type: none"> 1. = no invalid PC load UsageFault 2. = the processor has attempted an illegal load of EXC_RETURN to the PC, as a result of an invalid context, or an invalid EXC_RETURN value. <p>When this bit is set to 1, the PC value stacked for the exception return points to the instruction that tried to perform the illegal load of the PC.</p> |
| [1] | INVSTATE | <p>Invalid state UsageFault:</p> <ol style="list-style-type: none"> 1. = no invalid state UsageFault 2. = the processor has attempted to execute an instruction that makes illegal use of the EPSR. <p>When this bit is set to 1, the PC value stacked for the exception return points to the instruction that attempted the illegal use of the EPSR. This bit is not set to 1 if an undefined instruction uses the EPSR.</p> |

| Bits | Name | Function |
|------|------------|--|
| [0] | UNDEFINSTR | <p>Undefined instruction UsageFault:</p> <ol style="list-style-type: none"> 1. = no undefined instruction UsageFault 2. = the processor has attempted to execute an undefined instruction. <p>When this bit is set to 1, the PC value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode.</p> |

70 .UFSR bit assignments

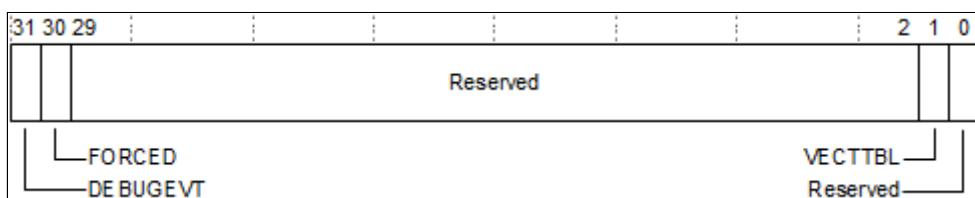

Note

The UFSR bits are sticky. This means as one or more fault occurs, the associated bits are set to 1. A bit that is set to 1 is cleared to 0 only by writing 1 to that bit, or by a reset.

HardFault Status Register

The HFSR gives information about events that activate the HardFault handler. See the register summary in Table 4-12 on page 4-11 for its attributes.

This register is read, write to clear. This means that bits in the register read normally, but writing 1 to any bit clears that bit to 0. The bit assignments are:



| Bits | Name | Function |
|--------|----------|--|
| [31] | DEBUGEVT | Reserved for Debug use. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable. |
| [30] | FORCED | Indicates a forced hard fault, generated by escalation of a fault with configurable priority that cannot be handles, either because of priority or because it is disabled: 0 = no forced HardFault 1 = forced HardFault. When this bit is set to 1, the HardFault handler must read the other fault status registers to find the cause of the fault. |
| [29:2] | - | Reserved. |
| [1] | VECTTBL | Indicates a BusFault on a vector table read during exception processing: 0 = no BusFault on vector table read 1 = BusFault on vector table read. This error is always handled by the hard fault handler. When this bit is set to 1, the PC value stacked for the exception return points to the instruction that was preempted by the exception. |
| [0] | - | Reserved. |

71 .HFSR bit assignments

Note

The HFSR bits are sticky. This means as one or more fault occurs, the associated bits are set to 1. A bit that is set to 1 is cleared to 0 only by writing 1 to that bit, or by a reset.

MemManage Fault Address Register

The MMFAR contains the address of the location that generated a MemManage fault. See the register summary in Table 50 on page 4-11 for its attributes. The bit assignments are:

| Bits | Name | Function |
|--------|---------|--|
| [31:0] | ADDRESS | When the MMARVALID bit of the MMFSR is set to 1, this field holds the address of the location that generated the MemManage fault |

72 .MMFAR bit assignments

When an unaligned access faults, the address is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size.

Flags in the MMFSR indicate the cause of the fault, and whether the value in the MMFAR is valid. See [MemManage Fault Status Register](#).

BusFault Address Register

The BFAR contains the address of the location that generated a BusFault. See the register summary in Table 50 for its attributes. The bit assignments are:

| Bits | Name | Function |
|--------|---------|--|
| [31:0] | ADDRESS | When the BFARVALID bit of the BFSR is set to 1, this field holds the address of the location that generated the BusFault |

73 .BFAR bit assignments

When an unaligned access faults the address in the BFAR is the one requested by the instruction, even if it is not the address of the fault.

Flags in the BFSR indicate the cause of the fault, and whether the value in the BFAR is valid. See [BusFault Status Register](#).

Auxiliary Fault Status Register

The AFSR contains additional system fault information. See the register summary in Table 50 for its attributes.

This register is read, write to clear. This means that bits in the register read normally, but writing 1 to any bit clears that bit to 0.

The bit assignments are:

| Bits | Name | Function |
|--------|--------|--|
| [31:0] | IMPDEF | The bits map to the AUXFAULT input signals. These inputs are hardcoded to zero |

74 .AFSR bit assignments

Each AFSR bit maps directly to an AUXFAULT input of the processor, and a single-cycle HIGH signal on the input sets the corresponding AFSR bit to one. It remains set to 1 until you write 1 to the bit to clear it to zero. When an AFSR bit is latched as one, an exception does not occur. Use an interrupt if an exception is required.

System control block design hints and tips

Ensure software uses aligned accesses of the correct size to access the system control block registers:

- except for the CFSR and SHPR1-SHPR3, it must use aligned word accesses
- for the CFSR and SHPR1-SHPR3 it can use byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to system control block registers. In a fault handler, to determine the true faulting address:

1. Read and save the MMFAR or BFAR value.
2. Read the MMARVALID bit in the MMFSR, or the BFARVALID bit in the BFSR. The MMFAR or BFAR address is valid only if this bit is 1.

Software must follow this sequence because another higher priority exception might change the MMFAR or BFAR value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the MMFAR or BFAR value.

1.4.3 System timer, SysTick

The processor has a 24-bit system timer, SysTick, that counts down from the reload value to zero, reloads, that is wraps to, the value in the SYST_RVR register on the next clock edge, then counts down on subsequent clocks.



Note

When the processor is halted for debugging the counter does not decrement.

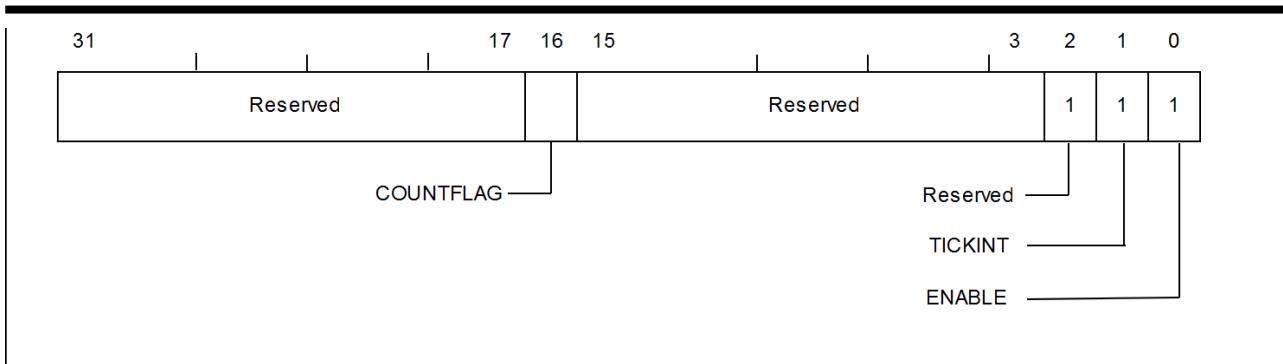
The system timer registers are:

| Address | Name | Type | Required privilege | Reset value | Description |
|-------------|----------|------|--------------------|-------------|-------------------------------------|
| 0xE000_E010 | SYST_CSR | RW | Privileged | 0x00000004 | SysTick Control and Status Register |
| 0xE000_E014 | SYST_RVR | RW | Privileged | 0x00000000 | SysTick Reload Value Register |
| 0xE000_E018 | SYST_CVR | RW | Privileged | 0x00000000 | SysTick Current Value Register |

75 .System timer registers summary

SysTick Control and Status Register

The SysTick SYST_CSR register enables the SysTick features. See the register summary in Table 71 for its attributes. The bit assignments are:



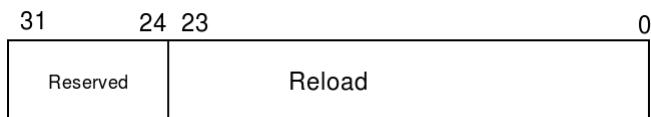
| Bits | Name | Function |
|---------|-----------|--|
| [31:17] | - | Reserved. |
| [16] | COUNTFLAG | Returns 1 if timer counted to 0 since last time this was read. |
| [15:3] | - | Reserved. |
| [2] | - | Reserved. |
| [1] | TICKINT | Enables SysTick exception request: 0 = counting down to zero does not assert the SysTick exception request 1 = counting down to zero to asserts the SysTick exception request. Software can use COUNTFLAG to determine if SysTick has ever counted to zero. |
| [0] | ENABLE | Enables the counter: 0 = counter disabled 1 = counter enabled. |

76 . SysTick SYST_CSR register bit assignments

When ENABLE is set to 1, the counter loads the RELOAD value from the SYST_RVR register and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.

SysTick Reload Value Register

The SYST_RVR register specifies the start value to load into the SYST_CVR register. See the register summary in Table 71 for its attributes. The bit assignments are:



| Bits | Name | Function |
|---------|--------|---|
| [31:24] | - | Reserved. |
| [23:0] | RELOAD | Value to load into the SYST_CVR register when the counter is enabled and when it reaches 0, see <i>Calculating the RELOAD value</i> . |

77 . SYST_RVR register bit assignments

Calculating the RELOAD value

The RELOAD value can be any value in the range 0x00000001-0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.

The RELOAD value is calculated according to its use. For example, to generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. If the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.

SysTick Current Value Register

The SYST_CVR register contains the current value of the SysTick counter. See the register summary in Table 71 on page 4-33 for its attributes. The bit assignments are:

| 31 | 24 23 | 0 |
|----------|---------|---|
| Reserved | CURRENT | |

| Bits | Name | Function |
|---------|---------|--|
| [31:24] | - | Reserved. |
| [23:0] | CURRENT | Reads return the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the SYST_CSR COUNTFLAG bit to 0. |

78 . SYST_CVR register bit assignments

SysTick design hints and tips

The SysTick counter runs on the processor clock. If this clock signal is stopped in sleep modes, the SysTick counter stops.

Ensure software uses aligned word accesses to access the SysTick registers.

The SysTick counter reload and current value are undefined at reset, the correct initialization sequence for the SysTick counter is:

1. Program reload value
2. Clear current value
3. Program Control and Status register

1.4.4 Memory protection unit

This section describes the *Memory protection unit* (MPU).

The MPU divides the memory map into a number of regions, and defines the location, size, access permissions, and memory attributes of each region. It supports:

- independent attribute settings for each region
- overlapping regions
- export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M4 MPU defines:

- eight separate memory regions, 0-7
- a background region.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps

region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M4 MPU memory map is unified. This means instruction accesses and data accesses have same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a MemManage fault. This causes a fault exception, and might cause termination of the process in an OS environment.

In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types, see [Memory regions, types and attributes](#).

Table 76 shows the possible MPU region attributes. These include Shareability and cache behavior attributes that are not relevant to most microcontroller implementations. See [MPU configuration for a microcontroller](#) for guidelines for programming such an implementation.

| Memory type | Shareability | Other attributes | Description |
|-------------------|--------------|---|--|
| Strongly- ordered | - | - | All accesses to Strongly-ordered memory occur in program order. All Strongly-ordered regions are assumed to be shared. |
| Device | Shared | - | Memory-mapped peripherals that several processors share. |
| | Non-shared | - | Memory-mapped peripherals that only a single processor uses. |
| Normal | Shared | Non-cacheable Write-through Cacheable Write-back Cacheable | Normal memory that is shared between several processors. |
| | Non-shared | Non-cacheable Write-through Cacheable Write-back Cacheable | Normal memory that only a single processor uses. |

79 . Memory attributes summary

Use the MPU registers to define the MPU regions and their attributes. The MPU registers are:

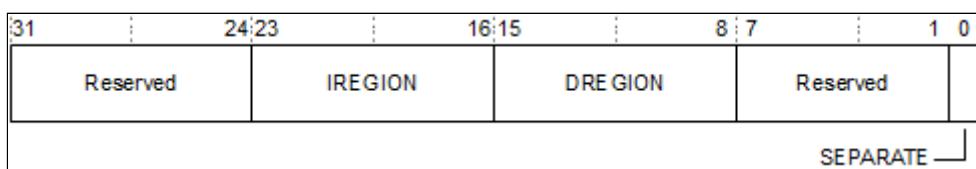
| Address | Name | Type | Required privilege | Reset value | Description |
|-------------|----------|------|--------------------|-------------|--------------------------------------|
| 0xE000_ED90 | MPU_TYPE | RO | Privileged | 0x0000_0800 | MPU Type Register |
| 0xE000_ED94 | MPU_CTRL | RW | Privileged | 0x0000_0000 | MPU Control Register |

| Address | Name | Type | Required privilege | Reset value | Description |
|-------------|-------------|------|--------------------|-------------|---|
| 0xE000_ED98 | MPU_RNR | RW | Privileged | 0x0000_0000 | MPU Region Number Register |
| 0xE000_ED9C | MPU_RBAR | RW | Privileged | 0x0000_0000 | MPU Region Base Address Register |
| 0xE000_EDA0 | MPU_RASR | RW | Privileged | 0x0000_0000 | MPU Region Attribute and Size Register |
| 0xE000_EDA4 | MPU_RBAR_A1 | RW | Privileged | 0x0000_0000 | Alias of RBAR, see MPU Region Base Address Register |
| 0xE000_EDA8 | MPU_RASR_A1 | RW | Privileged | 0x0000_0000 | Alias of RASR, see MPU Region Attribute and Size Register |
| 0xE000_EDAC | MPU_RBAR_A2 | RW | Privileged | 0x0000_0000 | Alias of RBAR, see MPU Region Base Address Register |
| 0xE000_EDB0 | MPU_RASR_A2 | RW | Privileged | 0x0000_0000 | Alias of RASR, see MPU Region Attribute and Size Register |
| 0xE000_EDB4 | MPU_RBAR_A3 | RW | Privileged | 0x0000_0000 | Alias of RBAR, see MPU Region Base Address Register |
| 0xE000_EDB8 | MPU_RASR_A3 | RW | Privileged | 0x0000_0000 | Alias of RASR, see MPU Region Attribute and Size Register |

80 . MPU registers summary

1.4.5 MPU Type Register

The MPU_TYPE register indicates whether the MPU is present, and if so, how many regions it supports. See the register summary in Table 77 for its attributes. The bit assignments are:



| Bits | Name | Function |
|---------|------|-----------|
| [31:24] | - | Reserved. |

| Bits | Name | Function |
|---------|----------|--|
| [23:16] | IREGION | Indicates the number of supported MPU instruction regions. Always contains 0x00. The MPU memory map is unified and is described by the DREGION field. |
| [15:8] | DREGION | Indicates the number of supported MPU data regions: 0x08 = Eight MPU regions. |
| [7:1] | - | Reserved. |
| [0] | SEPARATE | Indicates support for unified or separate instruction and date memory maps: 0 = unified. |

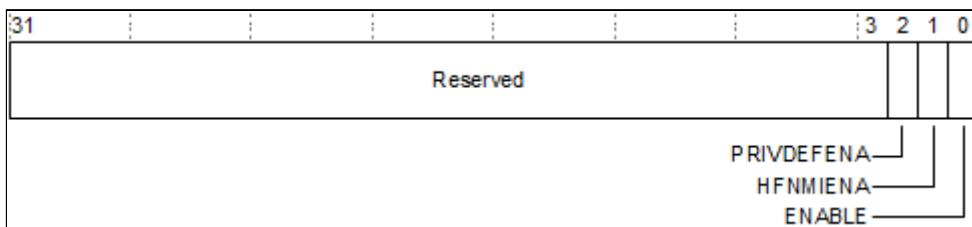
81 .TYPE register bit assignments

MPU Control Register

The MPU_CTRL register:

- enables the MPU
- enables the default memory map background region
- enables use of the MPU when in the hard fault, *Non-maskable Interrupt* (NMI), and FAULTMASK escalated handlers.

See the register summary in Table 77 for the MPU_CTRL attributes. The bit assignments are:



| Bits | Name | Function |
|--------|------------|--|
| [31:3] | - | Reserved. |
| [2] | PRIVDEFENA | Enables privileged software access to the default memory map: <ol style="list-style-type: none"> = If the MPU is enabled, disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault. = If the MPU is enabled, enables use of the default memory map as a background region for privileged software accesses. When enabled, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map. If the MPU is disabled, the processor ignores this bit. |

| Bits | Name | Function |
|------|----------|---|
| [1] | HFNMIENA | Enables the operation of MPU during hard fault, NMI, and FAULTMASK handlers. When the MPU is enabled: <ol style="list-style-type: none"> 1. = MPU is disabled during hard fault, NMI, and FAULTMASK handlers, regardless of the value of the ENABLE bit 2. = the MPU is enabled during hard fault, NMI, and FAULTMASK handlers. When the MPU is disabled, if this bit is set to 1 the behavior is Unpredictable. |
| [0] | ENABLE | Enables the MPU: 0 = MPU disabled 1 = MPU enabled. |

82 . MPU_CTRL register bit assignments

When ENABLE and PRIVDEFENA are both set to 1:

- For privileged accesses, the *default memory map* is as described in [Memory model](#). Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
- Any access by unprivileged software that does not address an enabled memory region causes a MemManage fault.

XN and Strongly-ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the ENABLE bit is set to 1, at least one region of the memory map must be enabled for the system to function unless the PRIVDEFENA bit is set to 1. If the PRIVDEFENA bit is set to 1 and no regions are enabled, then only privileged software can operate.

When the ENABLE bit is set to 0, the system uses the default memory map. This has the same memory attributes as if the MPU is not implemented, see Table 11. The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether PRIVDEFENA is set to 1.

Unless HFNMIENA is set to 1, the MPU is not enabled when the processor is executing the handler for an exception with priority -1 or -2. These priorities are only possible when handling a hard fault or NMI exception, or when FAULTMASK is enabled. Setting the HFNMIENA bit to 1 enables the MPU when operating with these two priorities.

1.4.6 MPU Region Number Register

The MPU_RNR selects which memory region is referenced by the MPU_RBAR and MPU_RASR registers. See the register summary in Table 77 for its attributes. The bit assignments are:



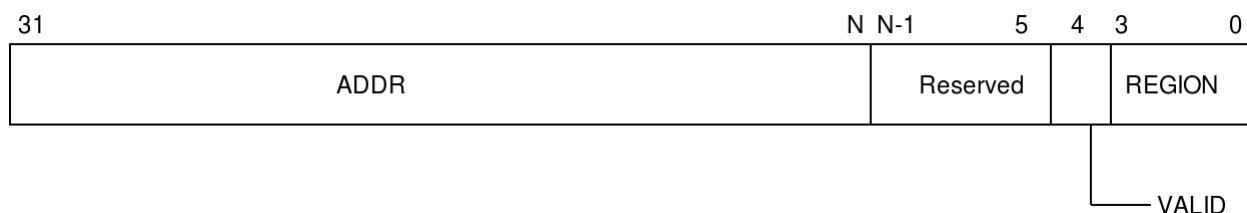
| Bits | Name | Function |
|--------|--------|--|
| [31:8] | - | Reserved. |
| [7:0] | REGION | Indicates the MPU region referenced by the MPU_RBAR and MPU_RASR registers. The MPU supports 8 memory regions, so the permitted values of this field are 0-7. |

83 . MPU_RNR bit assignments

Normally, you write the required region number to this register before accessing the MPU_RBAR or MPU_RASR. However you can change the region number by writing to the MPU RBAR with the VALID bit set to 1, see *MPU Region Base Address Register*. This write updates the value of the REGION field.

MPU Region Base Address Register

The MPU_RBAR defines the base address of the MPU region selected by the MPU_RNR, and can update the value of the MPU_RNR. See the register summary in Table 77 for its attributes.
Write MPU_RBAR with the VALID bit set to 1 to change the current region number and update the MPU_RNR. The bit assignments are:



If the region size is 32B, the ADDR field is bits [31:5] and there is no Reserved field

| Bits | Name | Function |
|-----------|--------|---|
| [31:N] | ADDR | Region base address field. The value of N depends on the region size. For more information see <i>The ADDR field</i> . |
| [(N-1):5] | - | Reserved. |
| [4] | VALID | MPU Region Number valid bit: Write: 0 = MPU_RNR not changed, and the processor: <ul style="list-style-type: none">• updates the base address for the region specified in the MPU_RNR• ignores the value of the REGION field 1 = the processor:• updates the value of the MPU_RNR to the value of the REGION field •updates the base address for the region specified in the REGION field. Always reads as zero. |
| [3:0] | REGION | MPU region field: For the behavior on writes, see the description of the VALID field. On reads, returns the current region number, as specified by the RNR. |

84 . MPU_RBAR bit assignments

The ADDR field

The ADDR field is bits[31:N] of the MPU_RBAR. The region size, as specified by the SIZE field in the MPU_RASR, defines the value of N:
 $N = \log_2(\text{Region size in bytes})$,

If the region size is configured to 4GB, in the MPU_RASR, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x00000000.

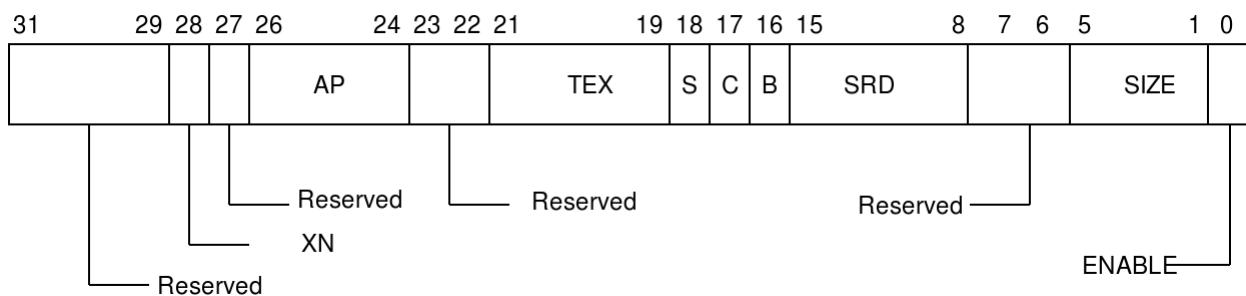
The base address is aligned to the size of the region. For example, a 64KB region must be aligned on a multiple of 64KB, for example, at 0x00010000 or 0x00020000.

MPU Region Attribute and Size Register

The MPU_RASR defines the region size and memory attributes of the MPU region specified by the MPU_RNR, and enables that region and any subregions. See the register summary in Table 82 for its attributes. MPU_RASR is accessible using word or halfword accesses:

- the most significant halfword holds the region attributes
 - the least significant halfword holds the region size and the region and subregion enable bits.

The bit assignments are:



| Bits | Name | Function |
|-----------------|-----------|---|
| [31:29] | - | Reserved. |
| [28] | XN | Instruction access disable bit: 1. = instruction fetches enabled 2. = instruction fetches disabled. |
| [27] | - | Reserved. |
| [26:24] | AP | Access permission field, see Table 4-48. |
| [23:22] | - | Reserved. |
| [21:19, 17, 16] | TEX, C, B | Memory access attributes, see Table 4-46. |
| [18] | S | Shareable bit, see Table 4-46. |
| [15:8] | SRD | Subregion disable bits. For each bit in this field: 1. = corresponding sub-region is enabled 2. = corresponding sub-region is disabled See Subregions for more information. Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, write the SRD field as 0x00. |

| Bits | Name | Function |
|-------|--------|---|
| [7:6] | - | Reserved. |
| [5:1] | SIZE | Specifies the size of the MPU protection region. The minimum permitted value is 3 (0b00010), see See <i>SIZE field values</i> for more information. |
| [0] | ENABLE | Region enable bit. |

85 .MPU_RASR bit assignments

For information about access permission, see *MPU access permission attributes*.

SIZE field values

The SIZE field defines the size of the MPU memory region specified by the RNR, as follows:

$$(\text{Region size in bytes}) = 2^{(\text{SIZE}+1)}$$

The smallest permitted region size is 32B, corresponding to a SIZE value of 4. Table 83 gives example SIZE values, with the corresponding region size and value of N in the MPU_RBAR.

| SIZE value | Region size | Value of N ^a | Note |
|--------------|-------------|-------------------------|------------------------|
| 0b00100 (4) | 32B | 5 | Minimum permitted size |
| 0b01001 (9) | 1KB | 10 | - |
| 0b10011 (19) | 1MB | 20 | - |
| 0b11101 (29) | 1GB | 30 | - |
| 0b11111 (31) | 4GB | 32 | Maximum possible size |

86 .Example SIZE field values

In the MPU_RBAR, see *MPU Region Base Address Register* on page 4-39.

MPU access permission attributes

This section describes the MPU access permission attributes. The access permission bits, TEX, C, B, S, AP, and XN, of the RASR, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault. Table 84 shows encodings for the TEX, C, B, and S access permission bits.

| TEX | C | B | S | Memory type | Shareability | Other attributes |
|-------|---|----------------|----------------|-----------------------------------|-------------------|--|
| 0b000 | 0 | 0 | x ^a | Strongly-ordered | Shareable | - |
| | | | 1 | x ^a | Device | Shareable |
| | 1 | 0 | 0 | Normal | Not Shareable | Outer and inner write-through. No write allocate. |
| | | | 1 | | Shareable | |
| | | | 1 | 0 | Normal | Outer and inner write-back. No write allocate. |
| | | | 1 | | Shareable | |
| 0b001 | 0 | 0 | 0 | Normal | Not Shareable | Outer and inner noncacheable |
| | | | 1 | | Shareable | |
| | 1 | 0 | x ^a | Reserved encoding | | - |
| | | | x ^a | Implementation defined attributes | | - |
| | | 1 | 0 | Normal | Not Shareable | Outer and inner write-back. Write and read allocate. |
| | | | 1 | | Shareable | |
| 0b010 | 0 | 0 | x ^a | Device | Not shareable | Nonshared Device. |
| | | | 1 | x ^a | Reserved encoding | |
| | 1 | x ^a | x ^a | Reserved encoding | | - |
| 0b1BB | A | A | 0 | Normal | Not Shareable | Cached memory, BB = outer policy, AA = inner policy. |
| | | | 1 | | Shareable | and BB bits.1ShareableSee Table 4-47 on page 4-43 for the encoding of the AA |

87 . TEX, C, B, and S encoding

The MPU ignores the value of this bit.

Table 85 shows the cache policy for memory attribute encodings with a TEX value is in the range 4-7.

| Encoding, AA or BB | Corresponding cache policy |
|--------------------|-------------------------------------|
| 00 | Non-cacheable |
| 01 | Write back, write and read allocate |
| 10 | Write through, no write allocate |
| 11 | Write back, no write allocate |

88 . Cache policy for memory attribute encoding

Table 4-48 shows the AP encodings that define the access permissions for privileged and unprivileged software.

| AP[2:0] | Privileged permissions | Unprivileged permissions | Description |
|---------|------------------------|--------------------------|---|
| 000 | No access | No access | All accesses generate a permission fault |
| 001 | RW | No access | Access from privileged software only |
| 010 | RW | RO | Writes by unprivileged software generate a permission fault |
| 011 | RW | RW | Full access |
| 100 | Unpredictable | Unpredictable | Reserved |
| 101 | RO | No access | Reads by privileged software only |
| 110 | RO | RO | Read only, by privileged or unprivileged software |
| 111 | RO | RO | Read only, by privileged or unprivileged software |

89 .AP encoding

MPU mismatch

When an access violates the MPU permissions, the processor generates a MemManage fault, see [Exceptions and interrupts](#). The MMFSR indicates the cause of the fault. See [MemManage Fault Status Register](#) for more information.

Updating an MPU region

To update the attributes for an MPU region, update the MPU_RNR, MPU_RBAR and MPU_RASR registers. You can program each register separately, or use a multiple-word write to program all of these registers. You can use the MPU_RBAR and MPU_RASR aliases to program up to four regions simultaneously using an STM instruction.

Updating an MPU region using separate words

Simple code to configure one region:

| | | |
|------|----------------|---|
| | | ; R1 = region number ; R2 = size/enable ; R3 = attributes ; R4 = address |
| LDR | R0,=MPU_RNR | ; 0xE000ED98, MPU region number register |
| STR | R1, [R0, #0x0] | ; Region Number |
| STR | R4, [R0, #0x4] | ; Region Base Address |
| STRH | R2, [R0, #0x8] | ; Region Size and Enable |
| STRH | R3, [R0, #0xA] | ; Region Attribute |

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

| | | |
|------|----------------|---|
| | | ; R1 = region number ; R2 = size/enable ; R3 = attributes ; R4 = address |
| LDR | R0,=MPU_RNR | ; 0xE000ED98, MPU region number register |
| STR | R1, [R0, #0x0] | ; Region Number |
| BIC | R2, R2, #1 | ; Disable |
| STRH | R2, [R0, #0x8] | ; Region Size and Enable |
| STR | R4, [R0, #0x4] | ; Region Base Address |
| STRH | R3, [R0, #0xA] | ; Region Attribute |
| ORR | R2, #1 | ; Enable |
| STRH | R2, [R0, #0x8] | ; Region Size and Enable |

Software must use memory barrier instructions:

- before MPU setup if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings
- after MPU setup if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not require any memory barrier instructions during MPU setup, because it accesses the MPU through the PPB, which is a Strongly-Ordered memory region.

For example, if you want all of the memory access behavior to take effect immediately after the programming sequence, use a DSB instruction and an ISB instruction. A DSB is required after changing MPU settings, such as at the end of context switch. An ISB is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then you do not require an ISB.

Updating an MPU region using multi-word writes

You can program directly using multi-word writes, depending on how the information is divided. Consider the following reprogramming:

| | | |
|-----|----------------|--|
| | | ; R1 = region number ; R2 = address ; R3 = size, attributes in one |
| LDR | R0,=MPU_RNR | ; 0xE000ED98, MPU region number register |
| STR | R1, [R0, #0x0] | ; Region Number |
| STR | R2, [R0, #0x4] | ; Region Base Address |
| STR | R3, [R0, #0x8] | ; Region Attribute, Size and Enable |

Use an STM instruction to optimize this:

| | | |
|-----|-------------|--|
| | | ; R1 = region number ; R2 = address ; R3 = size, attributes in one |
| LDR | R0,=MPU_RNR | ; 0xE000ED98, MPU region number register |

| | | |
|-----|-------------|--|
| STM | R0, {R1-R3} | ; Region Number, address, attribute, size and enable |
|-----|-------------|--|

You can do this in two words for pre-packed information. This means that the MPU_RBAR contains the required region number and had the VALID bit set to 1, see [MPU Region Base Address Register](#). Use this when the data is statically packed, for example in a boot loader:

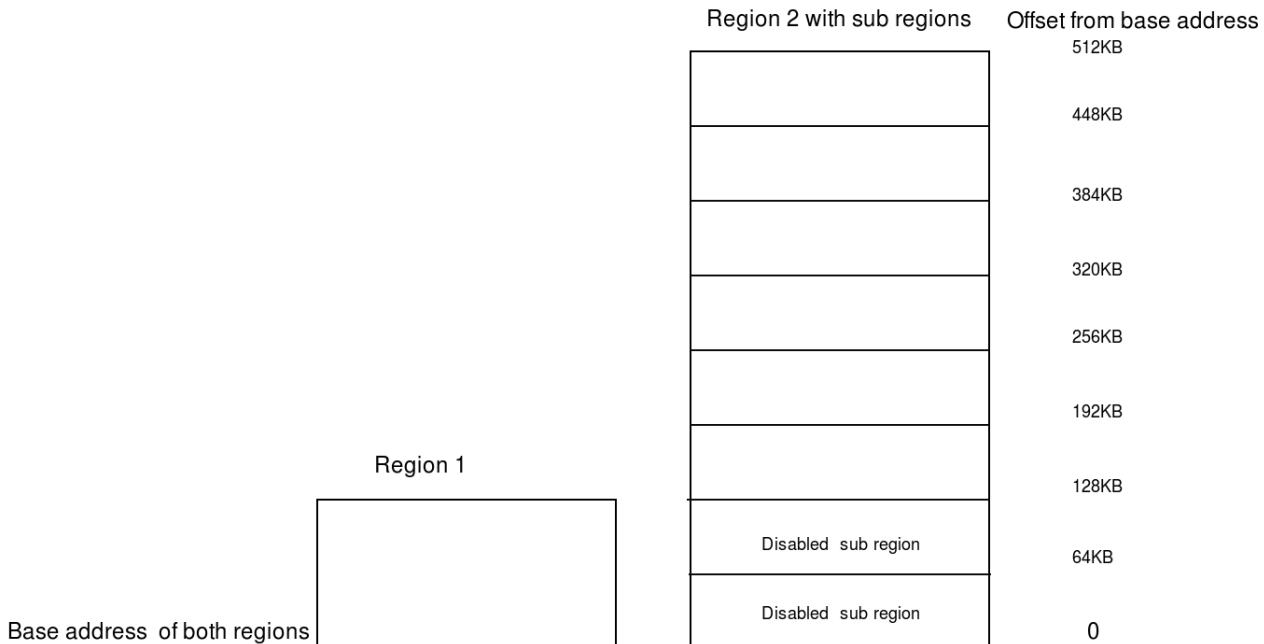
| | | |
|-----|---|---|
| | | ; R1 = address and region number in one ; R2 = size and attributes in one |
| LDR | R0, =MPU_RBAR; 0xE000ED9C, MPU Region Base register | |
| STR | R1, [R0, #0x0] | ; Region base address and ; region number combined with VALID (bit 4) set to 1 |
| STR | R2, [R0, #0x4] | ; Region Attribute, Size and Enable |

Subregions

Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the MPU_RASR to disable a subregion, see [MPU Region Attribute and Size Register](#). The least significant bit of SRD controls the first subregion, and the most significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion the MPU issues a fault. Regions of 32, 64, and 128 bytes do not support subregions, With regions of these sizes, you must set the SRD field to 0x00, otherwise the MPU behavior is Unpredictable.

Example of SRD use

Two regions with the same base address overlap. Region one is 128KB, and region two is 512KB. To ensure the attributes from region one apply to the first 128KB region, set the SRD field for region two to 0b00000011 to disable the first two subregions, as the figure shows.



MPU design hints and tips

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- except for the MPU_RASR, it must use aligned word accesses
- for the MPU_RASR it can use byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

MPU configuration for a microcontroller

Usually, a microcontroller system has only a single processor and no caches. In such a system, program the MPU as follows:

| Memory region | TEX | CBS | Memory type and attributes |
|---------------|-------|-------|--|
| Flash memory | 0b000 | 1 0 0 | Normal memory, Non-shareable, write-through |
| Internal SRAM | 0b000 | 1 0 1 | Normal memory, Shareable, write-through |
| External SRAM | 0b000 | 1 1 1 | Normal memory, Shareable, write-back, write-allocate |
| Peripherals | 0b000 | 0 1 1 | Device memory, Shareable |

90 . Memory region attributes for a microcontroller

In most microcontroller implementations, the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable. The values given are for typical situations. In special systems, such as multiprocessor designs or designs with a separate DMA engine, the shareability attribute might be important. In these cases see the recommendations of the memory device manufacturer.

1.4.7 Floating Point Unit (FPU)

The Cortex-M4F FPU implements the FPv4-SP floating-point extension.

The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

The FPU provides floating-point computation functionality that is compliant with the *ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic*, referred to as the IEEE 754 standard.

The FPU contains 32 single-precision extension registers, which you can also access as 16 doubleword registers for load, store, and move operations.

Table 88 shows the floating-point system registers in the Cortex-M4F FPU.

| Address | Name | Type | Reset | Description |
|-------------|-------|------|-------------|--|
| 0xE000_ED88 | CPACR | RW | 0x0000_0000 | <i>Coprocessor Access Control Register</i> |
| 0xE000_EF34 | FPCCR | RW | 0xC000_0000 | <i>Floating-point Context Control Register</i> |
| 0xE000_EF38 | FPCAR | RW | - | <i>Floating-point Context Address Register</i> |

| Address | Name | Type | Reset | Description |
|-------------|--------|------|-------------|--|
| - | FPSCR | RW | - | Floating-point Status Control Register |
| 0xE000_EF3C | FPDSCR | RW | 0x0000_0000 | Floating-point Default Status Control Register |

91 . Cortex-M4F floating-point system registers

The following sections describe the floating-point system registers whose implementation is specific to this processor.

1.4.8 Coprocessor Access Control Register

The CPACR register specifies the access privileges for coprocessors. See the register summary in *Cortex-M4F floating-point system registers* for its attributes. The bit assignments are:

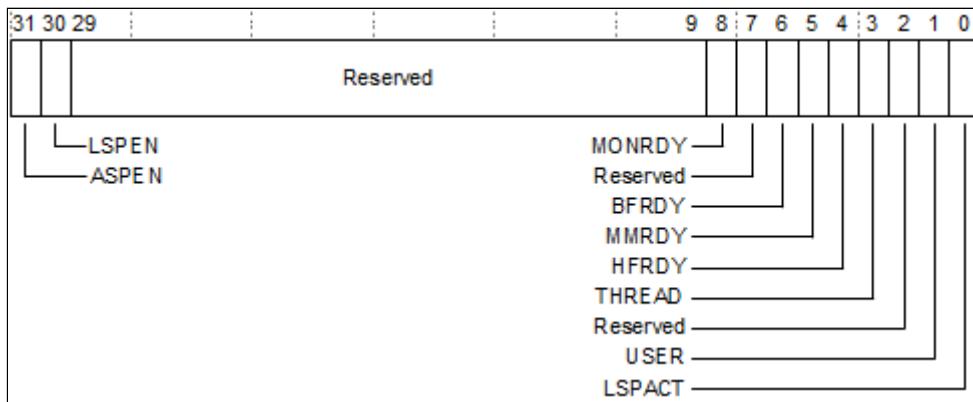
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|-------|-------|----------|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | CP 11 | CP 10 | Reserved | | | | | | | | | | | | | | | | | | | |

| Bits | Name | Function |
|-------------------------------------|--------|--|
| [31:24] | - | Reserved. Read as Zero, Write Ignore. |
| [2n+1:2n] for n values 10 and 11 | CP n | Access privileges for coprocessor n . The possible values of each field are: 0b00 = Access denied. Any attempted access generates a NOCP UsageFault. 0b01 = Privileged access only. An unprivileged access generates a NOCP fault. 0b10 = Reserved. The result of any access is Unpredictable. 0b11 = Full access. |
| [19:0] | - | Reserved. Read as Zero, Write Ignore. |

92 . CPACR register bit assignments

Floating-point Context Control Register

The FPCCR register sets or returns FPU control data. See the register summary in *Cortex-M4F floating-point system registers* for its attributes. The bit assignments are:



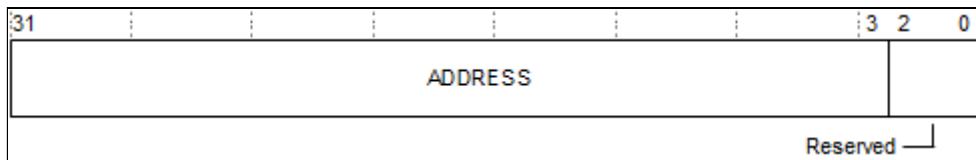
| Bits | Name | Function |
|--------|--------|---|
| [31] | ASPE N | Enables CONTROL<2> setting on execution of a floating-point instruction. This results in automatic hardware state preservation and restoration, for floating-point context, on exception entry and exit. 1. = Disable CONTROL<2> setting on execution of a floating-point instruction. 2. = Enable CONTROL<2> setting on execution of a floating-point instruction. |
| [30] | LSPEN | 1. = Disable automatic lazy state preservation for floating-point context. 2. = Enable automatic lazy state preservation for floating-point context. |
| [29:9] | - | Reserved. |
| [8] | MONRDY | 1. = DebugMonitor is disabled or priority did not permit setting MON_PEND when the floating-point stack frame was allocated. 2. = DebugMonitor is enabled and priority permits setting MON_PEND when the floating-point stack frame was allocated. |
| [7] | - | Reserved. |
| [6] | BFRDY | 1. = BusFault is disabled or priority did not permit setting the BusFault handler to the pending state when the floating-point stack frame was allocated. 2. = BusFault is enabled and priority permitted setting the BusFault handler to the pending state when the floating-point stack frame was allocated. |
| [5] | MMRDY | 1. = MemManage is disabled or priority did not permit setting the MemManage handler to the pending state when the floating-point stack frame was allocated. 2. = MemManage is enabled and priority permitted setting the MemManage handler to the pending state when the floating-point stack frame was allocated. |
| [4] | HFRDY | 1. = Priority did not permit setting the HardFault handler to the pending state when the floating-point stack frame was allocated. 2. = Priority permitted setting the HardFault handler to the pending state when the floating-point stack frame was allocated. |
| [3] | THREAD | 0 = Mode was not Thread Mode when the floating-point stack frame was allocated. 1 = Mode was Thread Mode when the floating-point stack frame was allocated. |
| [2] | - | Reserved. |

| Bits | Name | Function |
|------|--------|--|
| [1] | USER | 1. = Privilege level was not user when the floating-point stack frame was allocated. 2. = Privilege level was user when the floating-point stack frame was allocated. |
| [0] | LSPACT | 1. = Lazy state preservation is not active. 2. = Lazy state preservation is active. floating-point stack frame has been allocated but saving state to it has been deferred. |

93 .FPCCR register bit assignments

Floating-point Context Address Register

The FPCR register holds the location of the unpopulated floating-point register space allocated on an exception stack frame. See the register summary in [Cortex-M4F floating-point system registers](#) for its attributes. The bit assignments are:



| Bits | Name | Function |
|--------|---------|--|
| [31:3] | ADDRESS | The location of the unpopulated floating-point register space allocated on an exception stack frame. |
| [2:0] | - | Reserved. Read as Zero, Writes Ignored. |

94 .FPCAR register bit assignments

Floating-point Status Control Register

The FPSCR register provides all necessary User level control of the floating-point system. The bit assignments are:



| Bits | Name | Function |
|---------|-------|---|
| [31] | N | Condition code flags. Floating-point comparison operations update these flags. N Negative condition code flag. |
| [30] | Z | Z Zero condition code flag. |
| [29] | C | C Carry condition code flag. |
| [28] | V | V Overflow condition code flag. |
| [27] | - | Reserved. |
| [26] | AHP | Alternative half-precision control bit: 1. IEEE half-precision format selected. 2. Alternative half-precision format selected. |
| [25] | DN | Default NaN mode control bit: 1. NaN operands propagate through to the output of a floating-point operation. 2. Any operation involving one or more NaNs returns the Default NaN. |
| [24] | FZ | Flush-to-zero mode control bit: 1. Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. 2. Flush-to-zero mode enabled. |
| [23:22] | RMode | Rounding Mode control field. The encoding of this field is: 0b00 <i>Round to Nearest</i> (RN) mode 0b01 <i>Round towards Plus Infinity</i> (RP) mode 0b10 <i>Round towards Minus Infinity</i> (RM) mode 0b11 <i>Round towards Zero</i> (RZ) mode. The specified rounding mode is used by almost all floating-point instructions. |
| [21:8] | - | Reserved. |
| [7] | IDC | Input Denormal cumulative exception bit, see bits [4:0]. |
| [6:5] | - | Reserved. |
| [4] | IXC | Cumulative exception bits for floating-point exceptions, see also bit [7]. Each of these |
| [3] | UFC | bits is set to 1 to indicate that the corresponding exception has occurred since 0 was last written to it. |
| [2] | OFC | DZC Division by Zero cumulative exception bit. |
| [1] | DZC | IDC, bit[7] Input Denormal cumulative exception bit. IXC Inexact cumulative exception bit. UFC Underflow cumulative exception bit. OFC Overflow cumulative exception bit. DZC Division by Zero cumulative exception bit. IOC Invalid Operation cumulative exception bit. |

| Bits | Name | Function |
|------|------|----------|
| [0] | IOC | |

95 .FPSCR bit assignments

Floating-point Default Status Control Register

The FPDSCR register holds the default values for the floating-point status control data. See the register summary in [Cortex-M4F floating-point system registers](#) for its attributes. The bit assignments are:

| 31 | 27 | 26 | 25 | 24 | 23 | 21 | 0 |
|----------|----|----|----|----|----|----|----------|
| Reserved | 0 | 0 | 0 | 0 | 0 | | Reserved |

AHP DN FZ R Mode

| Bits | Name | Function |
|---------|-------|-------------------------------|
| [31:27] | - | Reserved |
| [26] | AHP | Default value for FPSCR.AHP |
| [25] | DN | Default value for FPSCR.DN |
| [24] | FZ | Default value for FPSCR.FZ |
| [23:22] | RMode | Default value for FPSCR.RMode |
| [21:0] | - | Reserved |

96 .FPDSCR register bit assignments

Enabling the FPU

The FPU is disabled from reset. You must enable it before you can use any floating-point instructions. Example 4-1 shows an example code sequence for enabling the FPU in both privileged and user modes. The processor must be in privileged mode to read from and write to the CPACR.

Example 4-1 Enabling the FPU

| | | |
|-------|----------------------|---|
| | | ; CPACR is located at address 0xE000ED88 |
| LDR.W | R0, =0xE000ED88 | ; Read CPACR |
| LDR | R1, [R0] | ; Set bits 20-23 to enable CP10 and CP11 coprocessors |
| ORR | R1, R1, #(0xF << 20) | ; Write back the modified value to the CPACR |
| STR | R1, [R0] | ; wait for store to complete |
| DSB | | ;reset pipeline now the FPU is enabled |
| ISB | | |

1.5 Instruction Cache Controller

1.5.1 General Description

The Instruction Cache Controller controls the instruction fetching from External Memory into Local Cache for access to the processor.

1.5.2 Features

- Bypass cache mode
- 32 or 128 bit line usage mode
- AHB wrap transfer mode
- Cache enable/disable mode
- 16k/8k access mode
- 4 Ways set associative, 4KB for each way
- There is a direct path for cortex I and D port to access icache.

1.5.3 Functional Description

The Cache controller controls the processor access to the External memory by loading 16 bytes of program data into the local cache every time a cache miss occurs.

The main functions of the ICC are as follows:

- Whenever there is a fetch request from the CPU, ICC reads the data and tag rams and gives grant to CPU, if controller wins the arbitration with AHB port for data and tag rams.
- In the next cycle, it checks whether it's a hit or miss.
- Whenever there is a hit, the ICC sends the instruction data required for the CPU, and gives fetch ready signal.
- Whenever there is a miss, the ICC sends a request to AHB master for getting data from the memory and copies into the local buffer. It also gives fetch miss signal to CPU. And asserts line_busy signal as soon as it gives trigger to AHB.
- When the hready for the required data is present ICC gives line ready signal.
- Line busy signal will be deasserted when AHB transfer is finished.
- CPU has to re-request for that address when line ready is given.
- ICC can serve hits under miss but not miss under miss cases.

Address mapping

| AHB address bits | | | Block accessed |
|------------------|-------|----|---|
| 31:26 | 15:14 | 13 | |
| 000001 | xx | x | Arm Cache Req Gen (accessible only through cortex I-port) |
| xxxxx0 | 00 | x | Control Registers |
| xxxxx0 | 01 | 0 | trams set1 |
| xxxxx0 | 01 | 1 | trams set2 |
| xxxxx0 | 10 | x | drams set1 |
| xxxxx0 | 11 | x | drams set2 |

| Ram | Offset Address Range | Size (in Bytes) |
|-----------|-------------------------------|-----------------|
| Dram set1 | 32'h0000_8000 – 32'h0000_BFFF | 16K |
| Dram set2 | 32'h0000_C000 – 32'h0000_FFFF | Not Present |

| Ram | Offset Address Range | Size (in Bytes) |
|-----------|-------------------------------|-----------------|
| Tram set1 | 32'h0000_4000 – 32'h0000_5FFF | 2K |
| Tram set2 | 32'h0000_6000 – 32'h0000_7FFF | Not Present |

1.5.4 Register Summary

Base Address for M4 Instance : 0x2028_0000

Base Address for TA Instance : 0x0080_0000

| Register Name | Offset | Description |
|----------------------------------|--------|---|
| RAM_CTRL_REG | 0x4 | Rams control register |
| ICACHE_CTRL_REG | 0x14 | Icache control register |
| ADDR_TRANSLATE_SEG1_CTRL_REG | 0x24 | Address Translate Value Segment Register_1 |
| ADDR_TRANSLATE_SEG2_CTRL_REG | 0x28 | Address Translate Value Segment Register_2 |
| ADDR_TRANSLATE_SEG3_CTRL_REG | 0x2C | Address Translate Value Segment Register_3 |
| ADDR_TRANSLATE_SEG4_CTRL_REG | 0x30 | Address Translate Value Segment Register_4 |
| ADDR_TRANSLATE_SEG5_CTRL_REG | 0x34 | Address Translate Value Segment Register_5 |
| ADDR_TRANSLATE_SEG6_CTRL_REG | 0x38 | Address Translate Value Segment Register_6 |
| ADDR_TRANSLATE_SEG7_CTRL_REG | 0x3C | Address Translate Value Segment Register_7 |
| ADDR_TRANSLATE_SEG8_CTRL_REG | 0x40 | Address Translate Value Segment Register_8 |
| ADDR_TRANSLATE_SEG9_CTRL_REG | 0x44 | Address Translate Value Segment Register_9 |
| ADDR_TRANSLATE_SEG10_CTRL_REG | 0x48 | Address Translate Value Segment Register_10 |
| ADDR_TRANSLATE_SEG11_CTRL_REG | 0x4C | Address Translate Value Segment Register_11 |
| ADDR_TRANSLATE_SEG12_CTRL_REG | 0x50 | Address Translate Value Segment Register_12 |
| ADDR_TRANSLATE_SEG13_CTRL_REG | 0x54 | Address Translate Value Segment Register_13 |
| ADDR_TRANSLATE_SEG14_CTRL_REG | 0x58 | Address Translate Value Segment Register_14 |
| ADDR_TRANSLATE_SEG15_CTRL_REG | 0x5C | Address Translate Value Segment Register_16 |
| THREAD_WAY_ALLOCATION_VECTOR_REG | 0xD4 | Thread way allocation vector register |

1.5.5 Register Description

RAM_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|-----|------------------------------|----------|-------------|--|
| 0 | Rams_ownership of second set | R/W | 0 | <p>This bit controls the ownership of second set of rams when number of ways is 4 and 32k memory is enabled.</p> <p>0-> controller cannot use set2. It is available on AHB.</p> <p>1-> controller gets access of dram set2 when cache is enabled</p> <p>When the number of ways is 8, the ownership of rams is always with controller.(when cache enabled).This bit is ignored</p> |

97 . Rams Ctrl Register

ICACHE_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------------------------|----------|-------------|--|
| 30:8 | Reserved | R | 0 | Reserved |
| 7 | disable_fetc_h_256bit_lb | R/W | 0 | When set, disables fetching from line buffer which is present in 256-bit mode prefetch module. This will help in saving two cycles, if a request results in miss but present in prefetch line buffer. Even if the request served by prefetch buffer, line busy, line ready signals from icc will be asserted. Impact of these assertion on TA is not clear and hence disable is added. These assertions shouldn't have any problem in M4 |
| 6 | mode_256bit_line | R/W | 0 | <p>0 -> One 128 bit line buffer is used. Four beat AHB transaction is initiated with the external memory</p> <p>1 -> Two 128 bit line buffers are used. Eight beat AHB transaction is initiated with the external memory</p> |
| 5 | lru_8ways | R/W | 0 | <p>0-> 8 ways logic is disabled in controller</p> <p>1-> 8 ways logic is disabled in controller</p> |
| 4 | icache_line_buf_invalid | R/W | 0 | <p>0-> line buffer valid for icc 'n'</p> <p>1-> line buffer invalid for icc 'n'</p> <p>This is a self clearing bit</p> |
| 3 | icache_ahb_wrap_mode | R/W | 0 | <p>0-> wrap mode is disabled for icc 'n'.</p> <p>1-> wrap mode is enabled for icc 'n'</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|---------------------|----------|-------------|---|
| 2 | mode32_128 bit_line | R/W | 0 | 0-> 128 bit mode enabled for icc 'n'. AHB requests will be 128 bit 1-> 32 bit mode is enabled for icc 'n'. AHB requests will be 32 bit The above is valid only when cache is disabled |
| 1 | bypass_cache | R/W | 0 | 0-> Fetch Requests are served through icache rams 1-> Fetch Requests are served via ahb, bypassing the cache rams The above is valid only when cache is enabled. |
| 0 | icache_enable | R/W | 0 | 0 -> Icache is not enabled for icc 'n' 1-> Icache is enabled and cache access can take place via ICC 'n' |

98 .ICache Ctrl Register

ADDR_TRANSLATE_SEG_N_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|-------|-----------------------|----------|-------------|---|
| 31:21 | Segment Address Value | R/W | 0x0 | TA - Hold the 11 bit segment translate address when icache is enabled. These 31:21 bits directly map to 31:21 bits of ahb address. M4 - For address translation functionality is disabled. In M4SS Addr_translate_value_seg1_ctrl1[22:21] are used for below purpose 21 - icache output will be registered and given to processor. This bit has to be set above 120MHz 22 - this is applicable only when above bit is set. If this bit is set, data will be served through unregistered path if there is hit to cache buffer line. To use this feature, enable_seq_access_ps_mode bit in Miscellaneous Registers0#MISC_CFG_MISC_CTRL_1 has to be set |
| 20:0 | Reserved | R | 0x0 | Reserved |

99 .Address Translate register

THREAD WAY ALLOCATION VECTOR REG

| Bit | Access | Function | Reset Value | Description |
|---------------------------------|-----------|----------|-------------|---|
| (31: num_threads*num_ways) | Reserved | R | 0x0 | Reserved |
| (num_threads* num_ways) -1:0 | Th_access | R/W | 0x0 | Thread way allocation vector: thread_alloc_vec[n] = th_access[((n+1)*num_ways-1:n*num_ways] |

100 . Thread_way_allocation register

2 MCU Bus Matrix

2.1 General Description

The High Performance MCU AHB ICM (Inter-connect Matrix) is a multi-layer interconnect implementation of the AHB protocol designed for higher performance and higher frequency systems. The ICM operates at the same frequency as the processor. A 14 Master x 12 Slave AHB ICM is used. Multilayer bus matrix enables simultaneous access of peripherals by different masters. This chapter discusses features, high level architecture and register interface details.

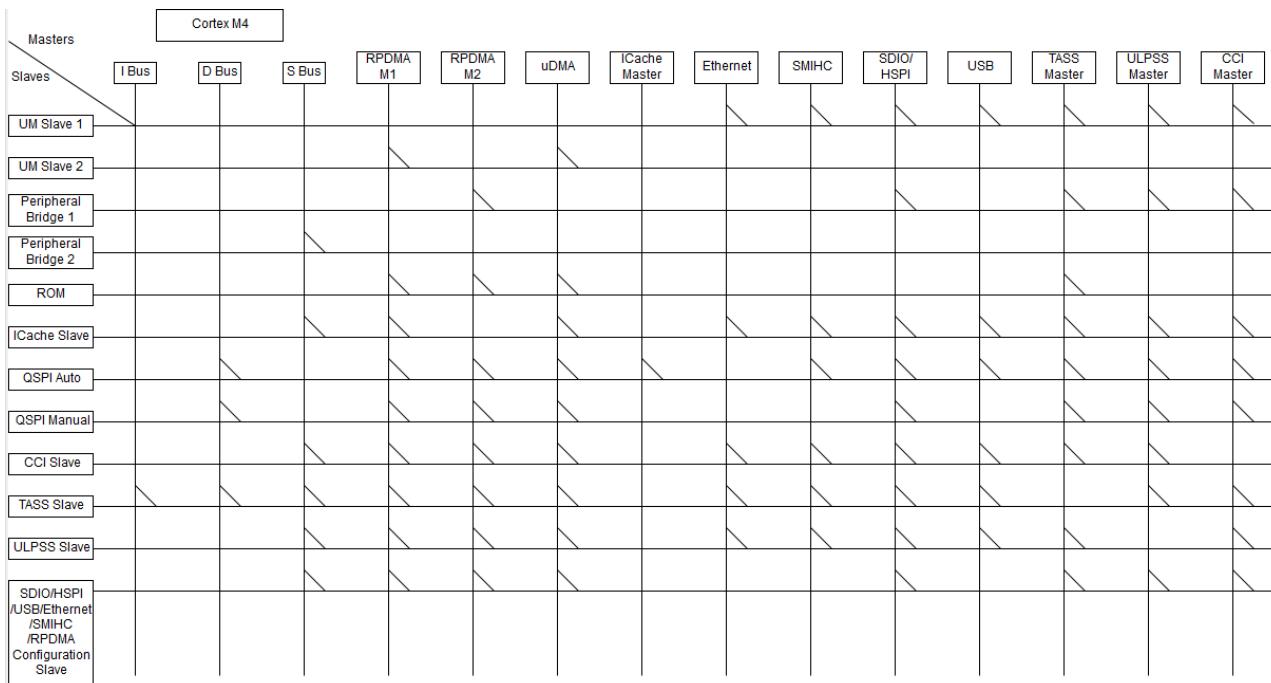
2.2 Features

- Multilayer interconnect matrix for high performance
- 14 Masters and 12 slaves
- Concurrent accesses allowed to slaves on different layers
- Operation at same frequency as Cortex M4 processor

2.3 Functional Description

2.3.1 Overview

The following diagram shows the interconnect configuration/connections possible between masters and slaves.



5 . High Performance AHB Bus Matrix Block Diagram



Note

Refer to the product-specific datasheet to identify available peripherals in your part.

2.3.2 Master and Slave Details

Following are the AHB masters connected to ICM

| Master Name | Description | Master ID |
|-------------------------|--|-----------|
| Cortex M4 I Bus | Processor's Instruction Bus | 5 |
| Cortex M4 D Bus | Processor's Data Bus | 6 |
| Cortex M4 S Bus | Processor's System Bus | 7 |
| GPDMA Master – 1 | Multi channel GPDMA has two parallel Master channels into the AHB bus matrix. Both DMA channels can independently access the bus matrix without any arbitration. | 1 |
| GPDMA Master – 2 | This helps in ensuring that the latency is minimized for latency-critical peripherals | 1 |
| uDMA Master | DMA Master for the Micro DMA | 9 |
| ICache Master | I-Cache fetches instructions from external NOR flash through this master | 4 |
| Ethernet | Ethernet controller's DMA | 3 |
| SMIH Controller | SD-MEM controller's DMA | 3 |
| SDIO/ HSPI | SDIO/ HSPI's DMA | 3 |
| USB | USB's DMA | 3 |
| NWP AHB to AHB Bridge | Used for accesses coming from NWP subsystem | 13 |
| ULPSS AHB to AHB Bridge | Used for accesses coming from ULP MCU subsystem | 11 |
| CCI Master | Used for accesses from CCI (Chip to Chip high speed parallel interface) | 11 |



Note

Master ID is used in debugging the origin of memory access which has caused memory trap

Following are the AHB slaves connected to ICM

| Slave | Description |
|-----------------------------|---|
| Unified Memory - 1 | Memory can be accessed through two slaves. DMAs(GPDMA/uDMA) use slave 2 and rest of the masters use slave 1. This helps in reducing the wait cycles when these two groups of masters are accessing different banks in the on-chip SRAM (bank size is 16K). Note that when the two masters are accessing same bank, then there will be wait states. Also note that processor buses have tightly coupled path to the memory and don't go through bus matrix |
| Unified Memory - 2 | |
| AHB to APB Bridge - 1 | All masters except the processor access the peripherals through this bridge |
| AHB to APB Bridge - 2 | Peripheral bridge port that is dedicated to the processor. This ensures minimal latency for peripheral accesses from processor |
| ROM | ROM has separate slave |
| Icache Slave | Icache controller configuration is done through this slave |
| QSPI Automode | Flash is presented as a memory mapped device and can be accessed through this slave |
| QSPI Manualmode | This channel can be used to configure the QSPI Flash controller and do manual mode writes and reads to flash |
| CCI Slave | Accesses to second chip through the CCI go through this slave |
| NWP AHB to AHB Bridge Slave | Accesses to NWP subsystem go through this slave |

| Slave | Description |
|---------------------------------|--|
| ULPSS AHB to AHB Bridge Slave | Accesses to ULP subsystem slaves go through this slave |
| SDIO/HSPI/USB/Ethernet/ | Accesses to SDIO/HSPI/USB/Ethernet/SMIHC/GPDMA configuration registers go through this slave |
| SMIHC/GPDMA Configuration Slave | |

2.3.3 Address Mapping

AHB Slaves Address Mapping

Following table has the base addresses of memories and high speed peripherals.

| | Module Name | Size | Start Address |
|------------------------|------------------------|-------------|---|
| Memories | | | |
| | LP SRAM | 128KB | 0x0000_0000 Note: Add 0x0020_0000 to the SRAM addresses for access from outside M4SS |
| | HP SRAM1 | 64KB | 0x0002_0000 |
| | HP SRAM2 | 64KB/192KB | 0x0003_0000 |
| | ROM | 64KB | 0x0030_0000 |
| | NWP ROM | 512KB | 0x0008_0000 - Tightly coupled path. For accessing NWP ROM through AHB2AHB bridge, use the address specified in NWP subsystem address mapping table |
| AHB Peripherals | | | |
| | QSPI Auto Mode | 64MB | 0x0800_0000 |
| | QSPI Manual Mode | 1MB | 0x1200_0000 |
| | SDIO/HSPI Slave | 64KB | 0x2020_0000 |
| | USB Slave | 64KB | 0x2021_0000 |
| | SMIHC Controller Slave | 128KB/128KB | 0x2022_0000/0x2026_0000 NWP can access sdmem only at 0x2026_0000 address |
| | Ethernet Slave | 128K | 0x2024_0000 |
| | Icache Slave | 64KB | 0x2028_0000 |
| | GPDMA Slave | 512KB | 0x2108_0000 |

| | | | |
|--|------------------------|--------|--|
| | ULPSS AHB Bridge Slave | 256KB | 0x2404_0000 |
| | APB Bridge | 64MB | 0x4400_0000 |
| | CCI Controller Slave | 0.75GB | 0x6000_0000 |
| | NWP AHB Bridge Slave | 512MB | 0x0080_0000 / 0x0400_0000 / 0x1000_0000 / 0x2010_0000 / 0x2040_0000 / 0x2100_0000 / 0x2200_0000 / 0x4000_0000 |

APB Peripherals Address Mapping

Following table has the base addresses of all low speed peripherals.

| Peripheral | Base Address |
|----------------------------|--------------|
| PERI-1 Power Domain | |
| UART1 | 0x4400_0000 |
| USART1 | 0x4400_0100 |
| I2C1 | 0x4401_0000 |
| SSI_MST | 0x4402_0000 |
| UDMA | 0x4403_0000 |
| PERI-2 Power Domain | |
| SSI_SLV1 | 0x4501_0000 |
| UART2 | 0x4502_0000 |
| GSPI_1 | 0x4503_0000 |
| CONFIG_TIMER | 0x4506_0000 |
| CAN1_CONTROLLER | 0x4507_0000 |
| CRC | 0x4508_0000 |
| HWRNG | 0x4509_0000 |
| Other Domains | |
| VIC | 0x4611_0000 |

| | |
|----------------------------|-------------|
| ROM_PATCH | 0x4612_0200 |
| EGPIO | 0x4613_0000 |
| CCI | 0x4617_0000 |
| REG_SPI | 0x4618_0000 |
| PMU | 0x4600_0000 |
| PAD_CFG | 0x4600_4000 |
| MISC_CFG | 0x4600_8000 |
| EFUSE | 0x4600_C000 |
| PERI-4 Power Domain | |
| SGPIO | 0x4700_0000 |
| I2C2 | 0x4704_0000 |
| I2S | 0x4705_0000 |
| QEI | 0x4706_0000 |
| PWM | 0x4707_0000 |

2.3.4 Register Summary

There are no registers in this module

2.3.5 Register Description

There are no registers in this module

3 Memory Architecture

3.1 General Description

There is on chip ROM, RAM and off chip FLASH connectivity. Sizes of ROM/RAM/FLASH will vary depending on the chip configuration.

3.2 Features

Highlights:

- Unified memory architecture - software can partition the memory between code and data usage
- Multiport - RAMs support multiport access - allowing simultaneous access from different masters(I, D, DMAs) to non overlapping regions without any cycle penalty
- ROM/RAMs are tightly coupled to the processor I/D buses to reduce the latency and power
- Supports memory protection - generates trap if unintended master accesses the memory

The Cortex-M4F processor has following memory:

- On-chip SRAM of 144K/208K/272K/400Kbytes based on the chip configuration
- Out of the above SRAM, 16Kbytes is present in the Ultra-low-power peripheral subsystem. This memory is present on the S-bus of the Cortex-M4 and is primarily used by the ULP MCU peripherals like VAD, FIM, ULP I2S, etc.
- 64Kbytes of ROM which holds the M4F peripheral drivers.
- 16Kbytes of Instruction cache enabling eXecute In Place (XIP) with external quad/octa SPI SDR/DDR flashes.
- Based on the package configuration up to 4MBytes of "in-package" Quad SPI flash is available for the M4
- eFuse of 64 bytes (available for customer applications)

The ThreadArch processor has following memory:

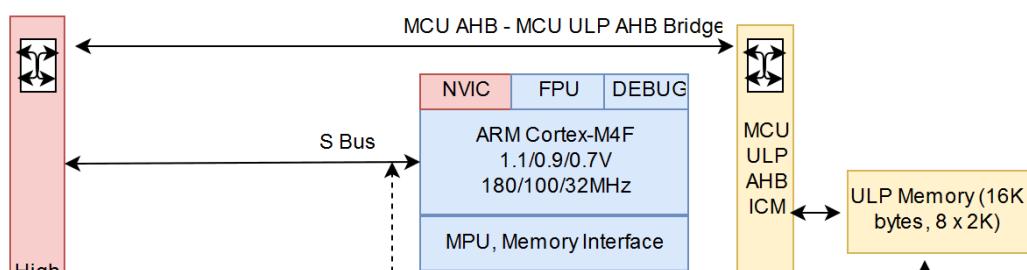
- On-chip SRAM of 384K/256K/192K/128Kbytes based on chip configuration
- 512Kbytes of ROM which holds the Secure primary bootloader, Network Stack, Wireless stacks and security functions
- 16Kbytes of Instruction cache enabling eXecute In Place (XIP) with external quad/octa SPI SDR/DDR flashes.
- Based on the package configuration up to 4MBytes of "in-package" Quad SPI flash is available for the NWP
- eFuse of 512 bytes (used to store primary boot configuration, security and calibration parameters)

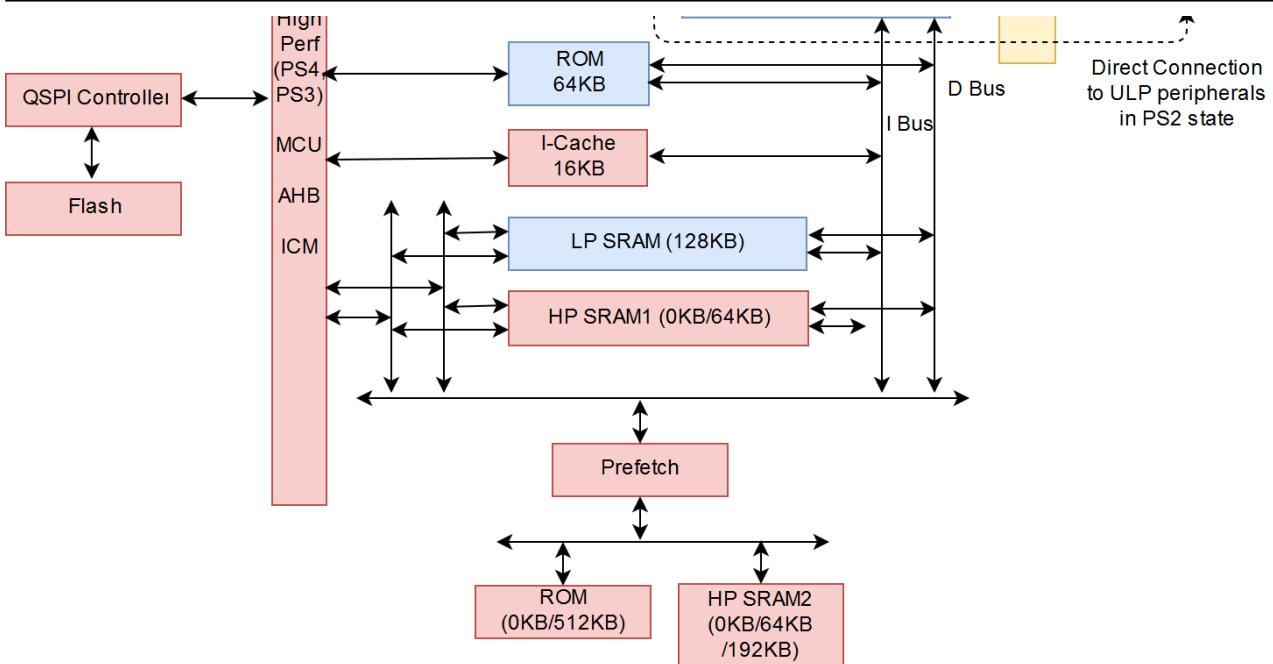
Based on the product and package configuration the following flash architectures are possible:

- No-Flash: Flash is not needed in devices where there is a host processor in the system that runs the network stack and wireless drivers.
- Single Flash used by M4: This is possible for non-Wi-Fi wireless MCUs (e.g., Wireless M4F MCU with BT 5)
- Single Flash used by TA: This is possible for Wireless MCUs where the MCU application code size is lesser than 128Kbytes (e.g., Wi-Fi + BT5 Sensor hub Wireless MCU)
- Two independent Flashes - one each for M4 and TA: This is needed for Wireless MCUs where Wi-Fi is needed and with MCU code size larger than 128KB.

3.3 Functional Description

Following diagram shows the memory architecture





Memory Architecture

3.4 Unified Memory Architecture and Multiport

The MCU have unified memory architecture. There is no hardware level partitioning between code and data sections. Software can partition according to the application requirement. Memory is divided into 16K physical banks. For best performance, it is better to partition the memory at 16K granularity. LP SRAM and HP SRAM1 support four ports. Two of the ports are connected I/D buses of Corex-M4. And other two ports are connected to ICM and meant to be used by other masters on bus like GPDMA, μDMA and high speed peripheral DMAs.

The amount of SRAM which is available for use varies between different power states(described in [Power Architecture Section](#)).

3.4.1 LP SRAM

LP SRAM is of size 128K. This can be accessed in PS4/PS3/PS2 states. This memory is available in all chip configurations. This memory supports four ports.

3.4.2 ROM

ROM is of size 64KB. This can be accessed in PS4/PS3/PS2 states. This will contain application boot code and some of the peripheral APIs. This memory is available in all chip configurations.

3.4.3 HP SRAM1

HP SRAM1 is of size 64KB. This can be accessed only in PS4/PS3 states. This memory is available only in some of the chip configurations. This memory supports four ports.

3.4.4 HP ROM

HP ROM is of size 512KB. This can be accessed only in PS4/PS3 states. This memory is available only in some of the chip configurations. This is a single port memory.

3.4.5 HP SRAM2

HP SRAM2 is of size 192KB. This can be accessed only in PS4/PS3 states. This memory is available only in some of the chip configurations. This is a single port memory. HP ROM and HP SRAM2 accesses will happen through same port.

3.4.6 ULP Memory

ULP memory is of size 16KB. This can be accessed in PS4/PS3/PS2 states. This is mainly used for sensor data collection from ULP peripherals. This is also used to feed the data for FIM accelerator. In PS2 state, processor will be accessing this memory directly without going through ICM and AHB2AHB bridge. This will reduce the cycles required to access this memory

3.4.7 ICache Memory

There is a 16KB of instruction cache memory. This can be accessed in PS4/PS3 states. This memory is also accessible through S bus at icache slave base address. When instruction cache is not used, this memory can be used as general purpose memory.

3.4.8 Flash Memory

Flash memory is accessed through QSPI controller. This can be accessed in PS4/PS2 states. All accesses to flash memory from I Bus will be routed to icache memory. If there is miss in icache memory, then icache controller will fetch it from external flash through QSPI controller in auto mode. Firmware has to configure the QSPI controller for auto mode before issuing any accesses from I Bus. D Bus accesses to flash address space will be routed to QSPI controller. Flash memory can also be accessed through S Bus by configuring QSPI controller in manual mode.

3.5 Frequency and Latency

| | Frequency | Latency(clock cycles) |
|----------|------------------|------------------------------|
| LP SRAM | 180MHz | 0 |
| HP SRAM1 | 180MHz | 0 |
| ROM | 135MHz | 0 |
| | 180MHz | 1 |
| HP SRAM2 | 100MHz | 0 |
| | 140MHz | 2 |
| HP ROM | 100MHz | 0 |
| | 140MHz | 2 |
| Icache | 100MHz | 0 |
| | 180MHz | 1 |



Note

Max frequency that is possible while using HP SRAM2 and HP ROM is 140MHz. By default all the memory paths are configured for 0 cycle latency mode. Before firmware is trying to operate at a frequency above the 0 cycle latency limit, firmware has enable higher cycle latency mode.

Refer [MCU Configuration Registers](#) for details of the high latency mode enable registers. To circumvent the higher latency in HP SRAM2 and HP ROM, there is prefetch logic in these memory paths. It is better to map code section to these memories and enable prefetching while operating these memories in higher latency mode.

3.6 Memory Protection

Trap is a memory protection technique that is used to detect unauthorized accesses to the memory. Accesses to a specific memory address or an address range can be detected. The feature does not prevent the tampering of the information present in the memory but just notifies the processor upon erroneous access. The details of the trap sources are captured in status registers. Based on this information processor has to take appropriate actions. This is supported for LP SRAM and HP SRAM1.

3.6.1 Features

- Supported for all banks of the unified memory
- Supports trap for accesses via all unified memory ports(I Bus, D Bus, DMAs)

- Supports independent trap enables per port and per bank
- Supports master ID based trap generation(refer [MCU Bus Matrix](#))
- Support to interrupt the processor on trap detection
- Trap status provided to processor via per port status registers and a common trap status register
- Provides information about the master that has caused the unauthorized access

3.6.2 Trap Generation and Handling

- Trap can be enabled per bank and per port
- As soon as unauthorized access is detected on any port for which trap has been enabled, trap interrupt is raised
- The status of the trap(as to which port it is generated from - I Bus/D Bus/DMA) is latched into trap status register
- Along with this, the per port status registers are updated with the following information
 - Present address on the port
 - Write/Read on the port
 - Grant for the port
 - Master ID accessing the port
- The status in all per port status registers is updated irrespective of trap detection on the corresponding port. After checking the grant bit of all status registers, we can get the information of which port transactions are valid. And then comparing the addresses of all granted ports with the trap address, we can identify the port which has caused the trap
- The information in trap status register and per port status registers gives idea to processor about
 - Port and master ID that has caused the trap
 - Unauthorized address
- Trap interrupt is mapped to NMI interrupt in Cortex M4
- The processor has to clear the information in the trap status register after reading information from the status registers and taking necessary actions. Until then, the trap status registers are not updated again even though a new trap occurs(irrespective of the port). Trap is serviced by clearing trap status register[3:0] bits

3.7 Programming Sequence

- Enable trap interrupt in [ENABLE_TRAP_REG](#)
- Enable trap per bank and per port via trap enable registers in IBUS_TRAP_ENABLE_REG*, DMA_WR_TRAP_ENABLE_REG* or DMA_RD_TRAP_ENABLE_REG*
- To enable trap for specific masters for DMA traps, set the respective bit in the AHB_MASTER_NUM_MASK_REG* register
- To disable trap, reset the respective bit in the AHB_MASTER_NUM_MASK_REG* register

3.8 Register Summary

Following are the list registers to enable and check the status of trap.

| Registers | Master |
|------------|---|
| IBUS* | Cortex M4 I Bus |
| DBUS* | Cortex M4 D Bus |
| DMA0* | UM Slave 1. Refer MCU Bus Matrix for masters using this this port |
| DMA1* | UM Slave 2. Refer MCU Bus Matrix for masters using this this port |
| DMA_RD/WR* | These registers are common between DMA0/DMA1/DBUS ports |

Base Address: 0x 4600_8000

| Register Name | Offset | Description |
|---------------------------------|--------|--------------------------|
| ENABLE_TRAP_REG | 0x70 | MCU Trap Enable Register |

| Register Name | Offset | Description |
|-----------------------------------|--------|---|
| IBUS_TRAP_ENABLE_REG_HP_SRA | 0x90 | Trap Enable Register for HP SRAM1 accesses from Cortex M4 I-Port M1 |
| DMA_WR_TRAP_ENABLE_REG_HP_SRAM1 | 0x94 | Trap Enable Register for HP SRAM1 Write accesses from Cortex M4 D-Port, AHB Slave 1 and 2 Ports |
| DMA_RD_TRAP_ENABLE_REG_HP_SRAM1 | 0x98 | Trap Enable Register for HP SRAM1 Read accesses from Cortex M4 D-Port, AHB Slave 1 and 2 Ports |
| AHB_MASTER_NUM_MASK_REG_H_P_SRAM1 | 0x9C | Mask Register for Enabling Trap in HP SRAM1 |
| AHB_MASTER_NUM_MASK_REG_L_P_SRAM | 0xA0 | Mask Register for Enabling Trap in LP SRAM |
| TRAP_DETECTED_REG_HP_SRAM1 | 0xA4 | Trap Detection Register for HP SRAM1 |
| IBUS_TRAP_STATUS_REG_HP_SR_AM1 | 0xA8 | Status Register for Trap on Cortex M4 I-port accessing HP SRAM1 |
| IBUS_TRAP_STATUS_REG_LP_SRA | 0xB4 | Status Register for Trap on Cortex M4 I-Port accessing LP SRAM M |
| DMA0_TRAP_STATUS_REG_HP_SR_AM1 | 0xB8 | Status Register for Trap on AHB Slave1 Port accessing HP SRAM1 |
| DMA0_TRAP_STATUS_REG_LP_SR_AM | 0xC4 | Status Register for Trap on AHB Slave1 Port accessing LP SRAM AM |
| DMA1_TRAP_STATUS_REG_HP_SR_AM1 | 0xC8 | Status Register for Trap on AHB Slave2 Port accessing HP SRAM1 |
| DMA1_TRAP_STATUS_REG_LP_SR_AM | 0xD4 | Status Register for Trap on AHB Slave2 Port accessing LP SRAM AM |
| DBUS_TRAP_STATUS_REG_HP_SR_AM1 | 0xD8 | Status Register for Trap on Cortex M4 D-Port accessing HP SRAM1 |
| DBUS_TRAP_STATUS_REG_LP_SR_AM | 0xE4 | Status Register for Trap on Cortex M4 D-Port accessing LP SRAM AM |
| IBUS_TRAP_ENABLE_REG_LP_SRA | 0x118 | Trap Enable Register for LP SRAM accesses from Cortex M4 I-Port M |
| DMA_RD_TRAP_ENABLE_REG_LP_SRAM | 0x124 | Trap Enable Register for LP SRAM Write accesses from Cortex M4 D-Port, AHB Slaves 1 and 2 Ports |
| DMA_WR_TRAP_ENABLE_REG_LP_SRAM | 0x130 | Trap Enable Register for LP SRAM Read accesses from Cortex M4 D-Port, AHB Slaves 1 and 2 Ports |
| TRAP_STATUS_HP_SRAM1 | 0x13C | Trap Status Register for HP SRAM1 |
| TRAP_STATUS_LP_SRAM | 0x148 | Trap Status Register for LP SRAM |
| TRAP_STATUS_HP_SRAM2 | 0x14C | Trap Status Register for HP SRAM2 |
| TRAP_CLEAR_HP_SRAM1 | 0x150 | Trap Clear Register for HP SRAM1 |
| TRAP_CLEAR_LP_SRAM | 0x15C | Trap Clear Register for LP SRAM |
| TRAP_CLEAR_HP_SRAM2 | 0x160 | Trap Clear Register for HP SRAM2 |
| TRAP_DETECTED_LP_SRAM | 0x164 | Trap Detection Register for LP SRAM |

| Register Name | Offset | Description |
|--|--------|--------------------------------------|
| TRAP_DETECTED_HP_SRAM2 | 0x168 | Trap Detection Register for HP SRAM2 |

101 .Memory Protection Register Table

3.9 Register Description

3.9.1 ENABLE_TRAP_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------|-------------|---|
| 31:10 | R/W | Reserved | 0x0 | Reserved |
| 9 | R/W | AHB_error_trap_enable | 0x0 | Enable the AHB error trap for cortex. <ul style="list-style-type: none"> • AHB Error Trap will not be raised to cortex • AHB Error Trap will be raised to cortex |
| 8 | R/W | apb_dummy_slave_selected | 0x0 | Hardware set this bit, When any of the AHB master is trying to access the wrong APB peripheral address. Firmware reset this bit by writing zero in this register. |
| 7:2 | R/W | Reserved | 0xF | Reserved |
| 1 | R/W | mem_error_trap_enable | 0x0 | Writing 1 to this enables Memory Error Trap to Cortex-M4F Processor |
| 0 | R/W | AHB_DUMMY_slave_selected | 0x0 | Read value of 1 indicates if any AHB master is trying to access the wrong slave address. Writing 0 to this will reset this bit |

102 .ENABLE_TRAP_REG Description

3.9.2 IBUS_TRAP_ENABLE_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------------------|-------------|--|
| 31:4 | NA | Reserved | 0x0 | Reserved |
| 3:0 | R/W | i_port_trap_enable_H_P_SRAM1 | 0x0 | Writing 1 to each bit enables Instruction-port trap for accesses to each 16K Bank of HP SRAM1. BIT(0) - Represent 1 st 16KB Bank of HP SRAM1 BIT(1) - Represent 2 nd 16KB Bank of HP SRAM1 BIT(2) - Represent 3 rd 16KB Bank of HP SRAM1 BIT(3) - Represent 4 th 16KB Bank of HP SRAM1 |

103 .IBUS_TRAP_ENABLE_REG_HP_SRAM1 Register Description

3.9.3 DMA_WR_TRAP_ENABLE_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|------|--------|--|-------------|--|
| 31:4 | NA | Reserved | 0x0 | Reserved |
| 3:0 | R/W | Dport_AHB_Slave_wr_0x0 trap_enable_HP_SRAM1 | 0x0 | <p>Writing 1 to each bit enables D-port, AHB Slaves 1 and 2 trap for write accesses to each 16K Bank of HP SRAM1</p> <p>BIT(0) - Represent 1st 16KB Bank of HP SRAM1</p> <p>BIT(1) - Represent 2nd 16KB Bank of HP SRAM1</p> <p>BIT(2) - Represent 3rd 16KB Bank of HP SRAM1</p> <p>BIT(3) - Represent 4th 16KB Bank of HP SRAM1</p> |

104 .DMA_WR_TRAP_ENABLE_REG_HP_SRAM1 Register Description

3.9.4 DMA_RD_TRAP_ENABLE_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|------|--------|--|-------------|---|
| 31:4 | NA | Reserved | 0x0 | Reserved |
| 3:0 | R/W | Dport_AHB_Slave_wr_0x0 trap_enable_HP_SRAM1 | 0x0 | <p>Writing 1 to each bit enables D-port, AHB Slaves 1 and 2 trap for read accesses to each 16K Bank of HP SRAM1</p> <p>BIT(0) - Represent 1st 16KB Bank of HP SRAM1</p> <p>BIT(1) - Represent 2nd 16KB Bank of HP SRAM1</p> <p>BIT(2) - Represent 3rd 16KB Bank of HP SRAM1</p> <p>BIT(3) - Represent 4th 16KB Bank of HP SRAM1</p> |

105 .DMA_RD_TRAP_ENABLE_REG_HP_SRAM1 Register Description

3.9.5 AHB_MASTER_NUM_MASK_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------------------|-------------|--|
| 31:16 | R | Reserved | 0x0 | Reserved |
| 15:0 | R/W | Trap_enable_per_master_HP_SRAM1 | 0x0 | <p>'1' on a particular bit position indicates that access by that master to a trap enable bank will generate trap, otherwise if it is '0' trap is not generated when it is accessing a trap enable bank.</p> <p>The masking is for UM1 transactions indicated in</p> <p>MCR_D_PORT_AND_AHB_SLAVE_WR_TRAP_ENABLE_64K_REG, MCR_D_PORT_AND_AHB_SLAVE_RD_TRAP_ENABLE_64K_REG.</p> |

106 .AHB_MASTER_NUM_MASK_REG_HP_SRAM1 Register Description

3.9.6 AHB_MASTER_NUM_MASK_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--|-------------|---|
| 31:16 | R/W | Trap enable bits per master for memory set 128KB | 0x0 | '1' on a particular bit position indicates that access by that master to a trap enable bank will generate trap, otherwise if it is '0' trap is not generated when it is accessing a trap enable bank. The masking is for UM1 transactions indicated in MCR_D_PORT_AND_AHB_SLAVE_WR_TRAP_ENABLE_128K_REG , MCR_D_PORT_AND_AHB_SLAVE_RD_TRAP_ENABLE_128K_REG |
| 15:0 | R | Reserved | 0x0 | Reserved |

107 .AHB_MASTER_NUM_MASK_REG_LP_SRAM Register Description

3.9.7 TRAP_DETECT_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------------------|-------------|--|
| 31:1 | R/W | Reserved | 0x0 | Reserved |
| 0 | R/W | Async_trap_detected_0x0 cortex | 0x0 | This will give the indication that trap has been detected by cortex. Firmware has to write this bit as '1' so that the async_trap can be cleared. |

108 .TRAP_DETECT_REG_HP_SRAM1 Register Description

3.9.8 IBUS_TRAP_STATUS_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read request on I port. |
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |
| 1 | R | Write/read | 0x0 | 0 - read request |
| 0 | R | Gnt | 0x0 | The grant signal for I port. |

109 .IBUS_TRAP_STATUS_REG_HP_SRAM1 Register Description

3.9.9 IBUS_TRAP_STATUS_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read request on I port. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------|-------------|---|
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |
| 1 | R | Write/read | 0x0 | 0 - read request |
| 0 | R | Gnt | 0x0 | The grant signal for I port. |

110 .IBUS TRAP STATUS REG LP_SRAM Register Description

3.9.10 DMA0_TRAP_STATUS_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read or write request on AHB Slave1 port. |
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |
| 1 | R | Write/read | 0x0 | 1 = write request 0 = read request |
| 0 | R | Gnt | 0x0 | The grant signal for AHB Slave1 port of unified memory. |

111 .DMA0 TRAP STATUS REG HP_SRAM1 Register Description

3.9.11 DMA0_TRAP_STATUS_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read or write request on AHB Slave1 port. |
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |
| 1 | R | Write/read | 0x0 | 1 = write request 0 = read request |
| 0 | R | Gnt | 0x0 | The grant signal for AHB Slave1 port of unified memory. |

112 .DMA0 TRAP STATUS REG LP_SRAM Register Description

3.9.12 DMA1_TRAP_STATUS_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read or write request on AHB Slave2 port. |
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 1 | R | Write/read | 0x0 | 1 = write request 0 = read request |
| 0 | R | Gnt | 0x0 | The grant signal for AHB Slave2 port of unified memory. |

113 .DMA1_TRAP_STATUS_REG_HP_SRAM1 Register Description

3.9.13 DMA1_TRAP_STATUS_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read or write request on AHB Slave2 port. |
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |
| 1 | R | Write/read | 0x0 | 1 = write request 0 = read request |
| 0 | R | Gnt | 0x0 | The grant signal for AHB Slave2 port of unified memory. |

114 .DMA1_TRAP_STATUS_REG_LP_SRAM Register Description

3.9.14 DBUS_TRAP_STATUS_REG_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read or write request on D port. |
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |
| 1 | R | Write/read | 0x0 | 1 = write request 0 = read request |
| 0 | R | Gnt | 0x0 | The grant signal for D port. |

115 .DBUS_TRAP_STATUS_REG_HP_SRAM1 Register Description

3.9.15 DBUS_TRAP_STATUS_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 31:25 | R | Reserved | 0x0 | Reserved |
| 24:6 | R | Address | 0x0 | The address for the read or write request on D port. |
| 5:2 | R | master_number | 0x0 | The number of the master which requested the transaction. |
| 1 | R | Write/read | 0x0 | 1 = write request 0 = read request |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|------------------------------|
| 0 | R | Gnt | 0x0 | The grant signal for D port. |

116 .DBUS_TRAP_STATUS_REG_LP_SRAM Register Description

3.9.16 IBUS_TRAP_ENABLE_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------------------------|-------------|---|
| 31:8 | NA | Reserved | 0x0 | Reserved |
| 7:0 | R/W | I port trap enable for 128K memory | 0x0 | I port trap enable bits per bank of 128K memory. LSB corresponds to 0th bank in the 128K memory set. A '1' in any bit position indicates that if that bank access is done through I port, a trap will be generated. |

117 .IBUS_TRAP_ENABLE_REG_LP_SRAM Register Description

3.9.17 DMA_RD_TRAP_ENABLE_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------|--------|---|-------------|---|
| 31:8 | NA | Reserved | 0x0 | Reserved |
| 7:0 | R/W | D and AHB Slave 1,2 ports write trap enable for 128K memory | 0x0 | D port and AHB Slave 1,2 ports write trap enable bits per bank of 64K memory LSB corresponds to 0th bank in the particular set. A '1' in any bit position indicates that if that bank access is done through a write on D port or AHB Slave 1 port or AHB Slave 2 port, a trap will be generated. |

118 .DMA_RD_TRAP_ENABLE_REG_LP_SRAM Register Description

3.9.18 DMA_WR_TRAP_ENABLE_REG_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------|--------|---|-------------|--|
| 31:8 | NA | Reserved | 0x0 | Reserved |
| 7:0 | R/W | D port and AHB Slave 1,2 ports read trap enable for 128K memory | 0x0 | D port and AHB Slave 1,2 ports write trap enable bits per bank of 64K memory LSB corresponds to 0th bank in the particular set. A '1' in any bit position indicates that if that bank access is done through a read on D port or AHB Slave 1 port or AHB Slave 2 port, a trap will be generated. |

119 .DMA_WR_TRAP_ENABLE_REG_LP_SRAM Register Description

3.9.19 TRAP_STATUS_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------------------|-------------|--|
| 31:5 | R | Reserved | 0x0 | Reserved |
| 3 | R | D port and AHB Slaves Write Trap | 0x0 | Async write trap is detected on any port among AHB Slave 1,2 and D ports for 64KB memory access. |
| 2 | R | D port and AHB Slaves Read Trap | 0x0 | Async read trap is detected on any port among AHB Slave 1,2 and D ports for 64KB memory access. |
| 1 | R | I Port Trap | 0x0 | Async trap is detected from cortex I port for 64KB memory access. |
| 0 | R | Reserved | 0x0 | Reserved |

120 .TRAP_STATUS_HP_SRAM1 Register Description

3.9.20 TRAP_STATUS_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------------------|-------------|---|
| 31:5 | R | Reserved | 0x0 | Reserved |
| 3 | R | D port and AHB Slaves Write Trap | 0x0 | Async write trap is detected from any port among AHB Slave 1,2 and D ports for 128KB memory access. |
| 2 | R | D port and AHB Slaves Read Trap | 0x0 | Async read trap is detected from any port among AHB Slave 1,2 and D ports for 128KB memory access. |
| 1 | R | I Port Trap | 0x0 | Async trap is detected from cortex I port for 128KB memory access. |
| 0 | R | Reserved | 0x0 | Reserved |

121 .TRAP_STATUS_LP_SRAM Register Description

3.9.21 TRAP_STATUS_HP_SRAM2

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------|-------------|--|
| 31:1 | R | Reserved | 0x0 | Reserved |
| 0 | R | ASYNC TRAP NWP | 0x0 | Async trap is detected from NWP memory (status has to be read from there). |

122 .TRAP_STATUS_HP_SRAM2 Register Description

3.9.22 TRAP_CLEAR_HP_SRAM1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------------------|-------------|--|
| 31:5 | R | Reserved | 0x0 | Reserved |
| 3 | W | D port and AHB slaves write trap | 0x0 | When set to 1, Async write trap is cleared from AHB Slave ports 1,2 and D port for 64KB memory access. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------------------|-------------|---|
| 2 | W | D port and AHB slaves read trap | 0x0 | When set to 1, Async read trap is cleared from AHB Slave ports 1,2 and D port for 64KB memory access. |
| 1 | W | I port trap | 0x0 | when set to 1, Async trap is cleared from cortex I port for 64KB memory access. |
| 0 | R | Reserved | 0x0 | Reserved |

123 . TRAP_CLEAR_HP_SRAM1 Register Description

3.9.23 TRAP_CLEAR_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------------------|-------------|---|
| 31:5 | R | Reserved | 0x0 | Reserved |
| 3 | W | D port and AHB slaves write trap | 0x0 | When set to 1, Async write trap is cleared from AHB Slave ports 1,2 and D port for 128KB memory access. |
| 2 | W | D port and AHB slaves read trap | 0x0 | When set to 1, Async read trap is cleared from AHB Slave ports 1,2 and D port for 128KB memory access. |
| 1 | W | I port trap | 0x0 | When set to 1, Async trap is cleared from cortex I port for 128KB memory access. |
| 0 | R | Reserved | 0x0 | Reserved |

124 . TRAP_CLEAR_LP_SRAM Register Description

3.9.24 TRAP_CLEAR_HP_SRAM2

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------|-------------|--|
| 31:1 | NA | Reserved | 0x0 | Reserved |
| 0 | W | ASYNC TRAP NWP | 0x0 | Async trap is cleared from NWP memory (status has to be read from NWP) |

125 . TRAP_CLEAR_HP_SRAM2 Register Description

3.9.25 TRAP_DETECTED_LP_SRAM

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------------|-------------|--|
| 31:1 | R/W | Reserved | 0x0 | Reserved |
| 0 | R/W | Async_trap_detected_cortex | 0x0 | This will give the indication that trap has been detected by cortex. Firmware has to write this bit as '1' so that the async_trap can be cleared. |

126 . TRAP_DETECTED_LP_SRAM Register Description

3.9.26 TRAP_DETECTED_HP_SRAM2

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|-------------|
| 31:1 | R/W | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------------------|-------------|--|
| 0 | R/W | Async_trap_detected_0x0 cortex | | This will give the indication that trap has been detected by cortex. Firmware has to write this bit as '1' so that the async_trap can be cleared. |

127 .TRAP_DETECTED_HP_SRAM2 Register Description

4 Clock Architecture

4.1 General Description

The Clock Architecture describes how to configure the on-chip clocks (ULP Clock Oscillators, High-Frequency PLL) and the clocks to Processor, High Speed Interfaces, and Peripherals (includes MCU HP, MCU ULP and UULP Vbat). The Clock subsystem enables the software to vary the clock source/frequency for different functionalities to achieve lower power consumption based on the application.

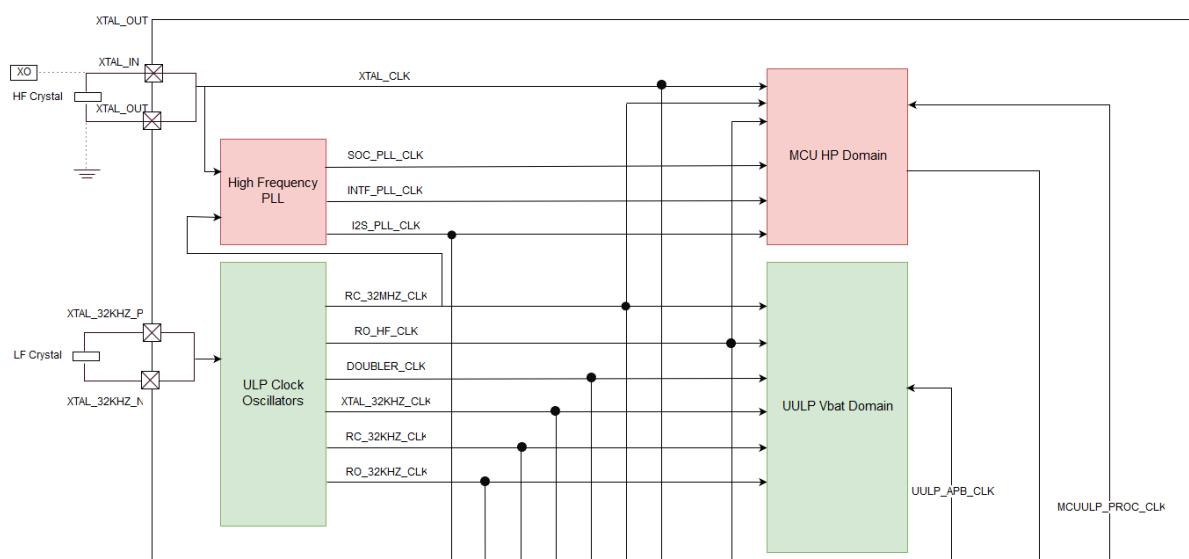
4.2 Features

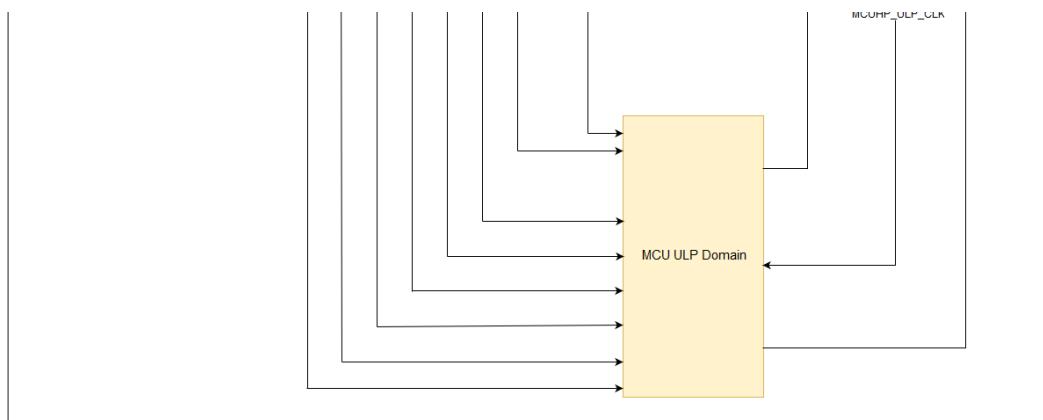
- Multiple high frequency clocks generated by PLLs
 - High Frequency Clock from 1MHz - 250MHz (SOC_PLL_CLK)
 - High Frequency Interface Clock from 1MHz - 110MHz/250MHz based on the part number. (INTF_PLL_CLK)
 - Defined frequencies for I²S Interface (I2S_PLL_CLK)
- Multiple clocks generated by ULP Clock Oscillators. These are low-power clock oscillators
 - External Crystal clock (XTAL_CLK)
 - RC 32MHz Clock (RC_32MHZ_CLK)
 - RO High-Frequency clock (RO_HF_CLK)
 - Doubler Clock (DOUBLER_CLK)
 - RC 32kHz Clock (RC_32KHZ_CLK)
 - RO 32kHz Clock (RO_32KHZ_CLK)
 - XTAL 32kHz clock (XTAL_32KHZ_CLK)
- Configurable independent division factors for varying the frequencies of different functional blocks
- Configurable independent clock gating for different functional blocks

4.3 Functional Description

4.3.1 Overview

The figure below depicts the top level overview of clocks being used for different domains.





Clocking Diagram

4.4 HF Crystal Clock

There are two possibilities to connect the external reference clock source on XTAL_IN and XTAL_OUT pins.

1. Connect Crystal between XTAL_IN and XTAL_OUT pins
2. Connect Crystal Oscillator on XTAL_IN and Ground XTAL_OUT pin (shown as a dotted line in the Figure 12)

The supported clock frequencies on XTAL_CLK are limited to the following frequencies

1. 9.6MHz
2. 13MHz
3. 19.2MHz
4. 20MHz
5. 26 MHz
6. 38.4MHz
7. 40MHz
8. 48MHz
9. 52MHz

The list below provided the different features supported for the High-Frequency Crystal Clock. Please refer to the "XTAL Clock" section of the MCU SAPI Manual for details on how to configure these features.

1. Ability to Operate on Battery Voltage or the DC-DC 1.35 Supply.
 - a. Applications like calibration of Low-Frequency RC/RO Clocks during sleep requires the HF-Crystal to be operating on Battery Voltage.

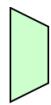
4.5 Naming Convention

| S.No | CLK_NAME | Source | Default Frequency | Frequency Range | Default State | Description |
|------|-------------|--------------------|--|---|---------------|---|
| 1 | XTAL_CLK | External | - | Defined Frequencies | Enabled | This is the external reference clock for PLLs |
| 2 | SOC_PLL_CLK | High-Frequency PLL | 90MHz/180MHz (Depends on the Part Number) | 1MHz to 110MHz/ 250MHz (Depends on the Part Number) | Enabled | High Frequency Clock from PLL |

| S.No | CLK_NAME | Source | Default Frequency | Frequency Range | Default State | Description |
|------|-----------------|-----------------------|--|---|---------------|--|
| 3 | INTF_PLL_CLK | High-Frequency PLL | 90MHz/180MHz (Depends on the Part Number) | 1MHz to 110MHz/ 250MHz (Depends on the Part Number) | Enabled | High Frequency Interface Clock from PLL |
| 4 | I2S_PLL_CLK | High-Frequency PLL | 6.144MHz | 256kHz to 24.576MHz | Enabled | I ² S Interface Clock from PLL |
| 5 | RC_32MHZ_CLK | ULP CLOCK OSCILLATORS | 32MHz | 20MHz to 40MHz | Enabled | Low power RC High Frequency clock source. |
| 6 | RO_HF_CLK | ULP CLOCK OSCILLATORS | 20MHz | 1MHz to 50MHz | Disabled | Low power RO High Frequency clock source. |
| 7 | DOUBLER_CLK | ULP CLOCK OSCILLATORS | Double the frequency of RO_HF_CLK/RC_32MHZ_CLK | | Disabled | Frequency Doubler Clock. |
| 8 | XTAL_32KHZ_C_LK | ULP CLOCK OSCILLATORS | 32KHz | Constant clock | Enabled | Low power XTAL Low Frequency clock source. |
| 9 | RC_32KHZ_CLK | ULP CLOCK OSCILLATORS | 32KHz | Constant clock | Enabled | Low power RC Low Frequency clock source. |
| 10 | RO_32KHZ_CLK | ULP CLOCK OSCILLATORS | 32KHz | 16kHz to 64kHz | Enabled | Low power RO Low Frequency clock source. |

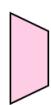
128 . Clocking Diagram Signal Descriptions

The following legend illustrates the components used in the clocking diagram's for different functional domains/ peripherals listed in this section. The text appearing below each component refers to the register bits for configuring and/or controlling the component.



CLOCK_SEL_REGISTER

Switches between any two active clock sources and the output is disabled when none of the clocks are selected. The input clock source can be disabled only after the output is switched to intended clock source.



CLOCK_SEL_REGISTER

Switches between the two inactive clock sources. The input clock sources need to be disabled before switching the clock source



DIV_FACTOR_REG
OUT_CLK_ENABLE

Divides the input clock by a factor of N. The output clock can be enabled/disabled. Division Factor of 0 doesn't perform any division on the clock



DIV_FACTOR_REG
OUT_CLK_ENABLE

Divides the input clock by a factor of 2*N. The output clock can be enabled/disabled. Division Factor of 0 doesn't perform any division on the clock



Divides the input clock by a factor of N (N needs to be a ODD number). The output clock can be enabled/disabled.

DIV_FACTOR_REG
OUT_CLK_ENABLE



Divides the input clock by a factor of $(N+0.5)$. The output clock can be enabled/disabled.

DIV_FACTOR_REG
OUT_CLK_ENABLE



Clock Gating Option. The output clock can be enabled/disabled.

CLK_ENABLE_REG

Clocking Legend

4.6 Reference Clock

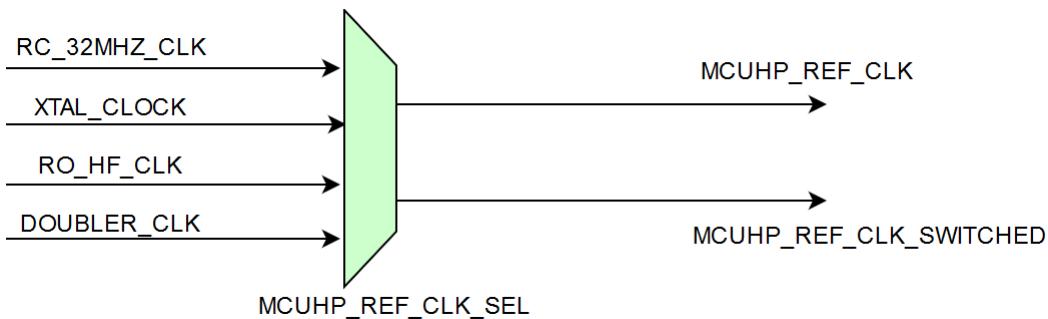
There are reference clock generated in MCU-HP and MCU ULP Domain which are reused in generation of clocks for peripherals/modules in respective domains.

The clock source selection for these reference clocks will be retained during Sleep mode and hence need not be re-configured on each wakeup.

4.6.1 MCU HP

The source for reference clock is configured through MCUHP_REF_CLK_SEL in MCU_REF_CLK_CONFIG Register.

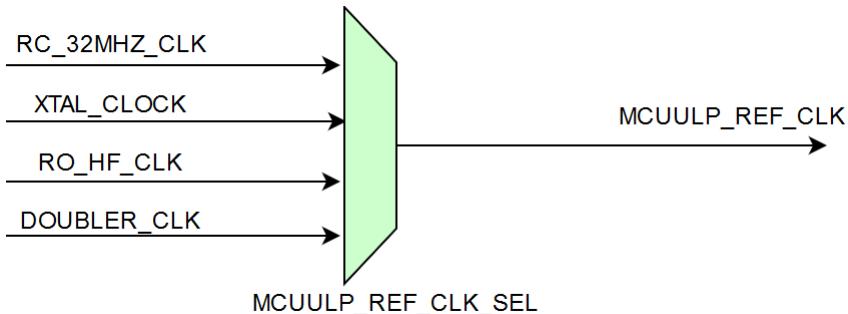
The Clock switching status can be read through MCUHP_REF_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register as described in the [MCU-HP Clock Architecture](#) section.



MCU-HP Reference Clock Generation

4.6.2 MCU ULP

The source for reference clock is configured through MCUULP_REF_CLK_SEL in MCU_REF_CLK_CONFIG Register.



MCU-ULP Reference Clock Generation

4.7 Clocking Schemes

The list below shows the organization of different clock generation mechanisms:

1. The generation of clocks from ULP Clock Oscillators are described in the [ULP Clock Oscillators](#) Section.
2. The generation of clocks from HIGH-Frequency PLL is described in the [High Frequency PLL](#) Section.
3. The generation of clocks for MCU HP Peripherals, MCU HP High SPEED Interface and the Cortex-M4F Processor are described in the [MCU HP Clock Architecture](#) Section.
4. The generation of clocks for MCU ULP Peripherals and the MCU ULP AHB are described in the [MCU ULP Clock Architecture](#) Section.
5. The generation of clocks for MCU UULP Vbat Peripherals are described in the [MCU ULP VBAT Clock Architecture](#) Section.

4.8 Register Summary

Base Address: 0x2404_8100

| Register Name | Offset | Description |
|------------------------------------|--------|--|
| MCU_REF_CLK_CONFIG | 0x1C | Reference Clock Configuration Register |

[129 . List of Registers](#)

4.9 Register Description

4.9.1 MCU_REF_CLK_CONFIG

| Bit | Access | Function | Default Value | Description |
|-----------|--------|----------|---------------|--|
| 31:1 9 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |
| 18:1 6 | | | 1 | Specifies the clock source to be used for MCU ULP Reference Clock. 0 - Output Clock is disabled 1 - RC_32MHZ_CLK 2 - Reserved 3 - XTAL_CLK 4 - Reserved 5 - RO_HF_CLK 6 - DOUBLER_CLK 7 - Reserved |
| 15:3 | - | | - | It is recommended to retain the contents by using read/modify write to this register. |

| | | | | |
|-----|-----|-------------------|---|---|
| 2:0 | R/W | MCUHP_REF_CLK_SEL | 1 | Specifies the clock source to be used for MCU HP Reference Clock. 0 - Output Clock is disabled 1 - RC_32MHZ_CLK 2 - Reserved 3 - XTAL_CLK 4 - Reserved 5 - RO_HF_CLK 6 - DOUBLER_CLK 7 - Reserved |
|-----|-----|-------------------|---|---|

130 .MCU_REF_CLK_CONFIG Description

4.10 High-Frequency PLL

4.10.1 General Description

The High-Frequency PLL is the source of the high frequency clocks used for Processor or the Peripherals.

4.10.2 Features

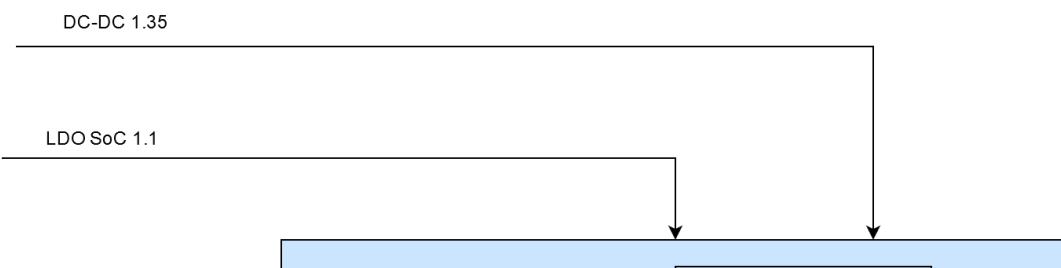
- High-Frequency Crystal clock (XTAL_CLOCK) or the Internal RC 32MHz clock (RC_32MHZ_CLK) can be used as a reference frequency for the PLLs.
- There are 3 independent PLLs present which generate the High Frequency clock sources.
 - SoC-PLL with frequency range of 1MHz - 110MHz/250MHz depending on the part number. Refer to the Ordering Information section of the Datasheet for information on part numbers.
 - Interface-PLL with frequency range of 1MHz - 110MHz/250MHz depending on the part number.
 - I2S-PLL with defined frequencies from 256KHz - 24.576MHz.
- PLL Lock time of 100us
- Each of the PLL can be powered down independently for efficient power management.
- The Output clock from each PLL can be disabled independently.

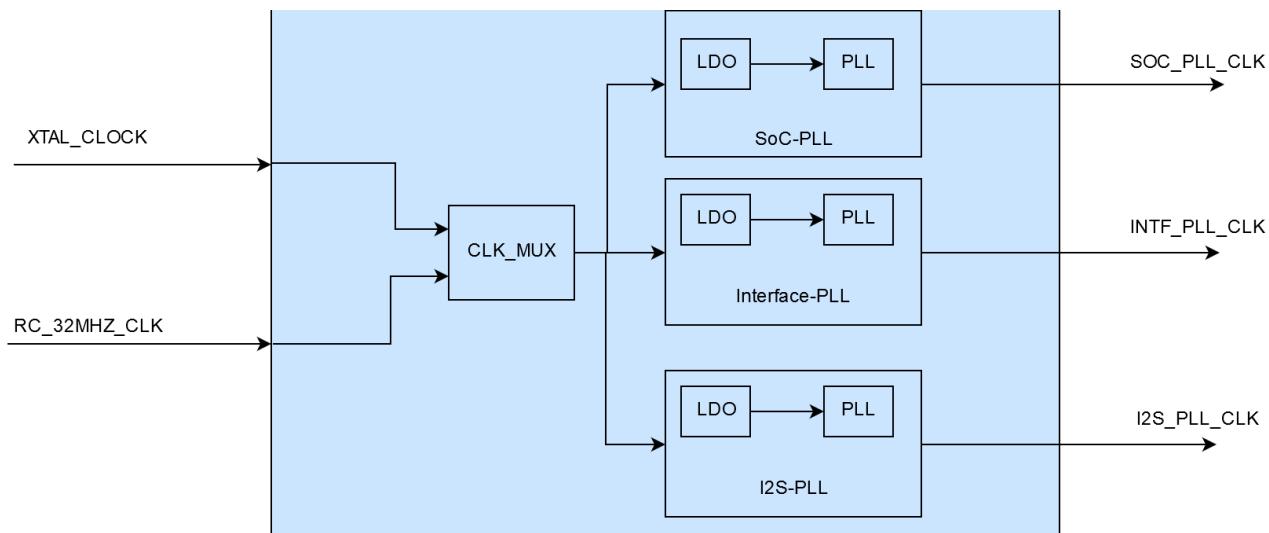
4.10.3 Functional Description

Overview

The LDO's internal to the PLL acts as an voltage source for the PLL. The DCDC 1.35 and LDO SoC 1.1 (as described in the Power Management Section) are used as the input source for these LDO's.

The Input voltage sources (DC-DC 1.35 and LDO SoC 1.1) as shown in the figure below needs to be enabled for the PLL to be active.





PLL Clocks

All PLL modules require certain locking period, during which the output may not be stable. As a result, before selecting a specific PLL output, the user program must check that the relevant lock bit is set before switching the clock.

The list below provides the names of the lock bits for SoC-PLL, Interface-PLL and I2S-PLL blocks.

- SOCPLL_LOCK in SOCPLL_STATUS_REG indicates the status for SoC-PLL
- INTFPLL_LOCK in INTFPLL_STATUS_REG indicates the status for Interface-PLL
- I2SPPLL_LOCK in I2SPPLL_STATUS_REG indicates the status for I2S-PLL

The following sections describe available configuration parameters for each PLL.

4.10.4 Input Reference Clock

There are two sources for the input reference clock to the PLLs. Either the High-Frequency Crystal Clock or the Internal RC-32MHz clock can be used as the reference clock.

The reference clock can be selected through REF_CLK_SEL in PLL_REF_CLK_CONFIG_REG Register.

4.10.5 SoC-PLL

Programming Sequence

The list below describes the programming sequence to be followed for achieving a particular output frequency.

1. Power-up the PLL through SOCPLL_PD in SOCPLL_CONFIG_REG1 Register.
2. Configure LDO Output Voltage through SOCPLL_LDO_PROG in PLL_LDO_CONFIG_REG register
 - a. If the Output intended frequency is greater than 200MHz it needs to be configured to 1.1V, else it needs to be configured to 1.05V.
3. Configure Frequency Range Selection through SOCPLL_RANGE_SEL in SOCPLL_CONFIG_REG1 Register.
4. Configure PLL Input Division Factor (NFAC) through SOCPLL_N in SOCPLL_CONFIG_REG2 Register.
 - a. NFAC should be chosen such that FIN (VCO Input Frequency) is in the range of 0.9MHz to 1.1MHz as per the below equation. FREF is the Input reference clock frequency to the PLL.

$$FIN = FREF / (NFAC + 1)$$

-
5. Configure PLL Output Division Factor (PFAC) through SOCPLL_P which is present in SOCPLL_CONFIG_REG2 Register.

- a. PFAC should be chosen from the following values - {1,3,7,15,31,127}.
- b. PFAC should be chosen such that VCOFREQ (VCO Output Frequency) is in the range of 127MHz to 250MHz as per the below equation.

$$VCOFREQ = FOUT * (PFAC + 1)$$

6. Configure PLL Multiplication Factor (MFAC) through SOCPLL_M in SOCPLL_CONFIG_REG1 Register.

- a. MFAC is derived from VCOFREQ, FIN parameters obtained in the above steps.
- b. If the intended output frequency is less than or equal to 200MHz, then MFAC is derived as per the below equation.

$$MFAC = \text{floor}(VCOFREQ/FIN) - 1$$

- c. If the intended output frequency is greater than 200MHz, then MFAC is derived as per the below equation.

$$MFAC = \text{floor}(VCOFREQ/(2 * FIN)) - 1$$

7. Configure PLL Fractional Frequency Control Word (FCW) through SOCPLL_FCW which is present in SOCPLL_CONFIG_REG3 Register.

- a. FCW is derived from MFAC, VCOFREQ, FIN parameters obtained in the above steps. This derivation of FCW is as per the below equation

$$FCW = (VCOFREQ/FIN - MFAC - 1) * 2^4$$

8. Enable the output clock through SOCPLL_CLK_EN in SOCPLL_CONFIG_REG1 Register.

9. Wait till the PLL output clock is stable by checking the SOCPLL_LOCK in SOCPLL_STATUS_REG Register.

4.10.6 Interface-PLL

Programming Sequence

The list below describes the programming sequence to be followed for achieving a particular output frequency.

1. Power-up the PLL through INTFPLL_PD in INTFPLL_CONFIG_REG1 Register.
2. Configure LDO Output Voltage through INTFPLL_LDO_PROG in PLL_LDO_CONFIG_REG register
 - a. If the Output intended frequency is greater than 200MHz it needs to be configured to 1.1V, else it needs to be configured to 1.05V.
3. Configure Frequency Range Selection through INTFPLL_RANGE_SEL in INTFPLL_CONFIG_REG1 Register.
4. Configure PLL Input Division Factor (NFAC) through INTFPLL_N in INTFPLL_CONFIG_REG2 Register.
 - a. NFAC should be chosen such that FIN (VCO Input Frequency) is in the range of 0.9MHz to 1.1MHz as per the below equation. FREF is the Input reference clock frequency to the PLL.

$$FIN = FREF/(NFAC + 1)$$

5. Configure PLL Output Division Factor (PFAC) through INTFPLL_P which is present in INTFPLL_CONFIG_REG2 Register.
 - a. PFAC should be chosen from the following values - {1,3,7,15,31,127}.
 - b. PFAC should be chosen such that VCOFREQ (VCO Output Frequency) is in the range of 127MHz to 250MHz as per the below equation.

$$VCOFREQ = FOUT * (PFAC + 1)$$

6. Configure PLL Multiplication Factor (MFAC) through INTFPLL_M in INTFPLL_CONFIG_REG1 Register.

- a. MFAC is derived from VCOFREQ, FIN parameters obtained in the above steps.
- b. If the intended output frequency is less than or equal to 200MHz, then MFAC is derived as per the below equation.

$$MFAC = \text{floor}(VCOFREQ/FIN) - 1$$

- c. If the intended output frequency is greater than 200MHz, then MFAC is derived as per the below equation.

$$MFAC = \text{floor}(VCOFREQ/(2 * FIN)) - 1$$

7. Configure PLL Fractional Frequency Control Word (FCW) through INTFPLL_FCW which is present in INTFPLL_CONFIG_REG3 Register.
 - a. FCW is derived from MFAC, VCOFREQ, FIN parameters obtained in the above steps. This derivation of FCW is as per the below equation

$$FCW = (VCOFREQ/FIN - MFAC - 1) * 2^{14}$$

8. Enable the output clock through INTFPLL_CLK_EN in INTFPLL_CONFIG_REG1 Register.
9. Wait till the PLL output clock is stable by checking the INTFPLL_LOCK in INTFPLL_STATUS_REG Register.

4.10.7 I2S-PLL

The following Output frequencies can be generated using I2S-PLL

- 256KHz
- 512KHz
- 768KHz
- 1.024MHz
- 1.4112MHz
- 2.048MHz
- 2.8224MHz
- 3.072MHz
- 4.096MHz
- 4.2336MHz
- 4.608MHz
- 5.6448MHz
- 6.144MHz
- 8.4672MHz
- 9.216MHz
- 11.2896MHz
- 12.288MHz
- 18.432MHz
- 24.576MHz

Programming Sequence

The list below describes the programming sequence to be followed for achieving a particular output frequency.

1. Power-up the PLL through I2SPPLL_PD in I2SPPLL_CONFIG_REG1 Register.
2. LDO Output Voltage has to be configured to 1.05V through I2SPPLL_LDO_PROG in PLL_LDO_CONFIG_REG Register.
3. Configure PLL Input Division Factor (NFAC) through I2SPPLL_N in I2SPPLL_CONFIG_REG2 Register.
 - a. NFAC is derived as per the below table. FREF is the Input reference clock frequency to the PLL.

| S.No | Fref (MHz) | NFAC | FIN (MHz) |
|------|------------|------|-----------|
| 1 | 9.6 | 10 | 0.96 |
| 2 | 13 | 13 | 1 |
| 3 | 16 | 16 | 1 |
| 4 | 19.2 | 20 | 0.96 |
| 5 | 26 | 26 | 1 |

| S.No | Fref (MHz) | NFAC | FIN (MHz) |
|------|------------|------|-----------|
| 6 | 32 | 32 | 1 |
| 7 | 38.4 | 40 | 0.96 |
| 8 | 40 | 40 | 1 |
| 9 | 52 | 52 | 1 |

131 . N_FAC Derivation for I2S-PLL

1. Configure PLL Multiplication Factor (MFAC), PLL Output Division Factors (PFAC1, PFAC2) and PLL Fractional Control word(FCW). These parameters are derived as per the tables below based on the FIN derived in the above steps.
 - a. MFAC can be configured through I2SPLL_M in I2SPLL_CONFIG_REG1 Register.
 - b. PFAC1, PFAC2 can be configured through I2SPLL_P1, I2SPLL_P2 in I2SPLL_CONFIG_REG2 Register.
 - c. FCW can be configured through I2SPLL_FCW which is present in I2SPLL_CONFIG_REG3 Register.

| FIN (Mhz) | MFAC | FCW | PFAC1 | PFAC2 | Output Freq (MHz) |
|-----------|------|-------|-------|-------|-------------------|
| 1 | 73 | 11928 | 2 | 0 | 24.576 |
| | | | 3 | 0 | 18.432 |
| | | | 5 | 0 | 12.288 |
| | | | 7 | 0 | 9.216 |
| | | | 11 | 0 | 6.144 |
| | | | 15 | 0 | 4.608 |
| | | | 17 | 0 | 4.096 |
| | | | 23 | 0 | 3.072 |
| | | | 17 | 1 | 2.048 |
| | | | 23 | 1 | 1.536 |
| | | | 17 | 2 | 1.024 |
| | | | 11 | 3 | 0.768 |
| | | | 17 | 3 | 0.512 |
| | | | 17 | 4 | 0.256 |
| | | | 67 | 5 | 0 |
| | | | | 7 | 0 |
| | | | | 11 | 0 |
| | | | | 15 | 0 |
| | | | | 23 | 0 |
| | | | | 23 | 1 |

132 . MFAC, FCW, PFAC1, PFAC2 Derivation for I2S-PLL with FIN = 1MHz

| Fin(Mhz) | M_Fac | FCW | P_Fac1 | P_Fac2 | Fout(Mhz) |
|-----------------|--------------|------------|---------------|---------------|------------------|
| 0.96 | 76 | 131072 | 0 | 0 | 24.576 |
| | | | 3 | 0 | 18.432 |
| | | | 5 | 0 | 12.288 |
| | | | 7 | 0 | 9.216 |
| | | | 11 | 0 | 6.144 |
| | | | 15 | 0 | 4.608 |
| | | | 17 | 0 | 4.096 |
| | | | 23 | 0 | 3.072 |
| | | | 17 | 1 | 2.048 |
| | | | 23 | 1 | 1.536 |
| | | | 17 | 2 | 1.024 |
| | | | 11 | 3 | 0.768 |
| | | | 17 | 3 | 0.512 |
| | | | 17 | 4 | 0.256 |
| | 70 | 9175 | 5 | 0 | 11.2896 |
| | | | 7 | 0 | 8.4672 |
| | | | 11 | 0 | 5.6448 |
| | | | 15 | 0 | 4.2336 |
| | | | 23 | 0 | 2.8224 |
| | | | 23 | 1 | 1.4112 |

133 . MFAC, FCW, PFAC1, PFAC2 Derivation for I2S-PLL with FIN = 0.96MHz

2. Enable the output clock through I2SPLL_CLK_EN in I2SPLL_CONFIG_REG1 Register.
3. Wait till the PLL output clock is stable by checking the I2SPLL_LOCK in I2SPLL_STATUS_REG Register.

4.10.8 PLL Programming Baud Rate

The PLL programming Baud Rate has to be configured as described below before accessing the PLL Configuration Registers. This is derived from the Processor clock as described in PLL_PROG_CTRL_REG register.

1. The maximum programming Baud rate should be 50MHz.

4.10.9 Register Summary

Base_address = 0x4618_0000

| Register Name | Offset | Description |
|------------------------|---------------|--|
| PLL_REF_CLK_CONFIG_REG | 0x04 | Reference Clock Configuration Register |
| PLL_LDO_CONFIG_REG | 0x08 | LDO Configuration Register |
| SOCPLL_CONFIG_REG1 | 0x40 | SoC-PLL Configuration Register1 |
| SOCPLL_CONFIG_REG2 | 0x44 | SoC-PLL Configuration Register2 |
| SOCPLL_CONFIG_REG3 | 0x48 | SoC-PLL Configuration Register3 |
| SOCPLL_STATUS_REG | 0x70 | SoC-PLL Status Register |
| INTFPLL_CONFIG_REG1 | 0x80 | Interface-PLL Configuration Register1 |
| INTFPLL_CONFIG_REG2 | 0x84 | Interface-PLL Configuration Register2 |
| INTFPLL_CONFIG_REG3 | 0x88 | Interface-PLL Configuration Register3 |
| I2SPLL_STATUS_REG | 0xB0 | Interface-PLL Status Register |
| I2SPLL_CONFIG_REG1 | 0xC0 | I2S-PLL Configuration Register1 |
| I2SPLL_CONFIG_REG2 | 0xC4 | I2S-PLL Configuration Register2 |

| Register Name | Offset | Description |
|--------------------|--------|---------------------------------|
| I2SPLL_CONFIG_REG3 | 0xC8 | I2S-PLL Configuration Register3 |
| I2SPLL_STATUS_REG | 0xF0 | I2S-PLL Status Register |

Base Address: 0x4008_0000

| Register Name | Offset | Description |
|-------------------|--------|----------------------------------|
| PLL_PROG_CTRL_REG | 0x00 | PLL Programming Control Register |

134 . Register Summary

4.10.10 Register Description

PLL_REF_CLK_CONFIG_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:14 | R/W | REF_CLK_SEL | 0 | Specified the input reference clock for the PLL's 0 - XTAL_CLK 1 - Reserved 2 - RC_32MHZ_CLK 3 - Reserved |
| 13:0 | - | Reserved | - | It is recommended to write these bits to 0. |

135 . Reference Clock Configuration Description

PLL_LDO_CONFIG_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|--|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:13 | R/W | SOCPLL_LDO_P ROG | 4 | Specified the configuration of SoC-PLL LDO output voltage 0-3 - Reserved 4 - 1.05V 5 - 1.1V 6,7 - Reserved |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------|-------------|--|
| 12:10 | R/W | INTFPLL_LDO_P ROG | 4 | Specified the configuration of Interface-PLL LDO output voltage 0-3 - Reserved 4 - 1.05V 5 - 1.1V 6,7 - Reserved |
| 9:7 | R/W | I2SPPLL_LDO_PR OG | 4 | Specified the configuration of I2S-PLL LDO output voltage 0-3 - Reserved 4 - 1.05V 5 - 1.1V 6,7 - Reserved |
| 6:0 | - | Reserved | - | It is recommended to write these bits to 0. |

136 . Common Control Register Description

SOCPLL_CONFIG_REG1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|--|
| 31:16 | - | Reserved | | It is recommended to write these bits to 0. |
| 15:6 | R/W | SOCPLL_M | 179 | Specifies the SoC-PLL Multiplication Factor. |
| 5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4 | R/W | SOCPLL_PD | 0 | Writing 1 to this disables power to the SoC-PLL. Writing 0 to this enables power to the SoC-PLL. |
| 3 | R/W | SOCPLL_CLK_EN | 1 | Writing 1 to this enables SoC-PLL Output clock. Writing 0 to this disables SoC-PLL Output clock. |
| 2 | - | Reserved | - | It is recommended to write these bits to 0. |
| 1:0 | R/W | SOCPLL_RANGE_SEL | 1 | Specifies the range for the Output frequency. 0 - Greater than 200MHz 1 - Less than 200MHz 2,3 - Reserved |

137 . SOCPLL_CONFIG_REG1 Description

SOCPLL_CONFIG_REG2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|--|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:9 | R/W | SOCPLL_P | 0 | Specifies the SoC-PLL Output Division Factor |
| 8:3 | R/W | SOCPLL_N | 39 | Specifies the SoC-PLL Input Division Factor |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|---|
| 2:0 | - | Reserved | - | It is recommended to write these bits to 0. |

138 . SOCPLL_CONFIG_REG2 Description

SOCPLL_CONFIG_REG3

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:2 | R/W | SOCPLL_FCW | 0 | Specifies the SoC-PLL Fractional Frequency Control Word |
| 1:0 | - | Reserved | - | It is recommended to write these bits to 0. |

139 . SOCPLL_CONFIG_REG3 Description

SOCPLL_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|--|
| 31:16 | - | Reserved | - | |
| 15 | R | SOCPLL_LOCK | 0 | Indicates the SoC-PLL Status 0 - Not Locked 1 - Locked |
| 14:0 | R/W | Reserved | - | |

140 . SOCPLL_STATUS_REG Description

INTFPLL_CONFIG_REG1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|---|
| 31:16 | - | Reserved | | It is recommended to write these bits to 0. |
| 15:6 | R/W | INTFPLL_M | 179 | Specifies the Interface-PLL Multiplication Factor. |
| 5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4 | R/W | INTFPLL_PD | 0 | Writing 1 to this disables power to the Interface-PLL. Writing 0 to this enables power to the Interface-PLL. |
| 3 | R/W | INTFPLL_CLK_EN | 1 | Writing 1 to this enables Interface-PLL Output clock. Writing 0 to this disables Interface-PLL Output clock. |
| 2 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-------------------|-------------|--|
| 1:0 | R/W | INTFPLL_RANGE_SEL | 1 | Specifies the range for the Output frequency. 0 - Greater than 200MHz 1 - Less than 200MHz 2,3 - Reserved |

141 .INTFPLL_CONFIG_REG1 Description

INTFPLL_CONFIG_REG2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------|-------------|--|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:9 | R/W | INTFPLL_P | 0 | Specifies the Interface-PLL Output Division Factor |
| 8:3 | R/W | INTFPLL_N | 39 | Specifies the Interface-PLL Input Division Factor |
| 2:0 | - | Reserved | - | It is recommended to write these bits to 0. |

142 .INTFPLL_CONFIG_REG2 Description

INTFPLL_CONFIG_REG3

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:2 | R/W | INTFPLL_FCW | 0 | Specifies the Interface-PLL Fractional Frequency Control Word |
| 1:0 | - | Reserved | - | It is recommended to write these bits to 0. |

143 .INTFPLL_CONFIG_REG3 Description

INTFPLL_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------|-------------|--|
| 31:16 | - | Reserved | - | |
| 15 | R | INTFPLL_LOCK | 0 | Indicates the Interface-PLL Status 0 - Not Locked 1 - Locked |
| 14:0 | R/W | Reserved | - | |

144 .INTFPLL_STATUS_REG Description

I2SPPLL_CONFIG_REG1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:6 | R/W | I2SPPLL_M | 73 | Specifies the I2S-PLL Multiplication Factor |
| 5 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------|-------------|---|
| 4 | R/W | I2SPLL_PD | 0 | Writing 1 to this disables power to the I2S-PLL. Writing 0 to this enables power to the I2S-PLL. |
| 3 | - | Reserved | - | It is recommended to write these bits to 0. |
| 2 | R/W | I2SPLL_CLK_EN | 1 | Writing 1 to this enables I2S-PLL Output clock. Writing 0 to this disables I2S-PLL Output clock. |
| 1:0 | - | Reserved | - | It is recommended to write these bits to 0. |

145 .I2SPLL_CONFIG_REG1 Description

I2SPLL_CONFIG_REG2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:11 | R/W | I2SPLL_P1 | 11 | Specifies the I2S-PLL Post Division factor1 |
| 10:8 | R/W | I2SPLL_P2 | 0 | Specifies the I2S-PLL Post Division factor2 |
| 7:1 | R/W | I2SPLL_N | 40 | Specifies the I2S-PLL Input Division factor |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

146 .I2SPLL_CONFIG_REG2 Description

I2SPLL_CONFIG_REG3

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:2 | R/W | I2SPLL_FCW | 14'd11928 | Specifies the I2S-PLL Fractional frequency control word |
| 1:0 | - | Reserved | 0 | It is recommended to write these bits to 0. |

147 .I2SPLL_CONFIG_REG3 Description

I2SPLL_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|--|
| 31:16 | - | Reserved | - | |
| 15 | R | I2SPLL_LOCK | 0 | Indicates the I2S-PLL Status 0 - Not Locked 1 - Locked |
| 14:0 | - | Reserved | - | |

148 .I2SPLL_STATUS_REG Description

PLL_PROG_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------|-------------|---|
| 31:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3:0 | R/W | BAUD_RATE | 4 | Specifies the Programming Baud Rate w.r.t source clock Programming_Baud_Rate = Processor_Clock/ ((BAUD_RATE+1)*2) |

149 .PLL_PROG_CTRL_REG Description

4.11 MCU HP Clock Architecture

4.11.1 General Description

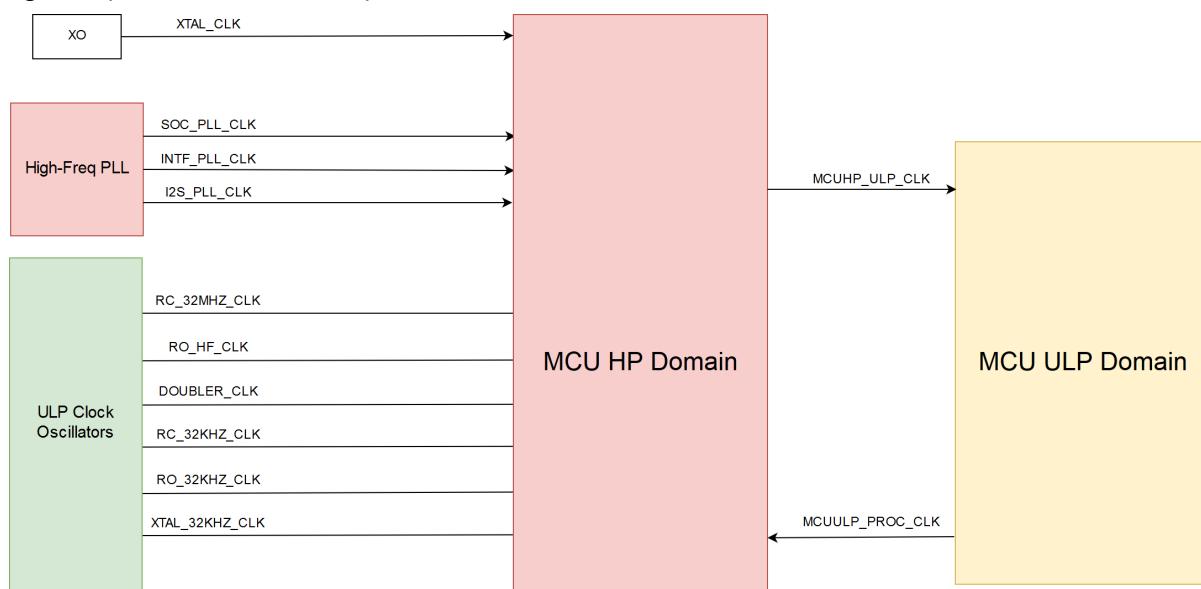
The MCU HP (High Performance) domain contains the Cortex-M4F Processor, FPU, Debugger, MCU High Speed Interfaces, MCU HP Peripherals, MCU HP DMA and MCU/SZP shareable Interfaces. This section describes the different clock sources possible for each interface/peripheral and the Processor.

4.11.2 Features

- The clock sources used for MCU HP domain includes RC, RO, XTAL clocks in addition to the PLL generated high frequency clocks.
- A dedicated PLL is present for High Speed Interfaces like Ethernet, SD-MEM (eMMC), CCI, UART, etc.
- A dedicated PLL is present for I²S interface with pre-defined frequencies.
- The frequency and clock source for High SPEED Interfaces and few Master Peripherals can be configured independently of the Processor clock.
- A clock synchronous to the processor clock is generated which can be used for MCU ULP AHB and Peripherals.
- The Processor, FPU, SRAM and MCU HP AHB Bridge operates on the same clock.

4.11.3 Functional Description

The sections below describes the clock architecture and the corresponding programming details. The following figure depicts the clock sources present in MCU HP domain.



MCU HP Clocking Scheme Overview

The clocks to following blocks can be configured independently.

1. Processor
2. SPI Flash Controller
3. Low-Power Clock
4. SDMEM(eMMC)
5. Chip-to-Chip AHB Interface (CCI)
6. Ethernet RMII
7. UART1
8. UART2
9. SPI/SSI Master
10. I²S in Master Mode
11. CAN Controller
12. Configurable Timers
13. Generic-SPI Master
14. Clock for MCU ULP Domain
15. External Clock

The following blocks use the processor clock.

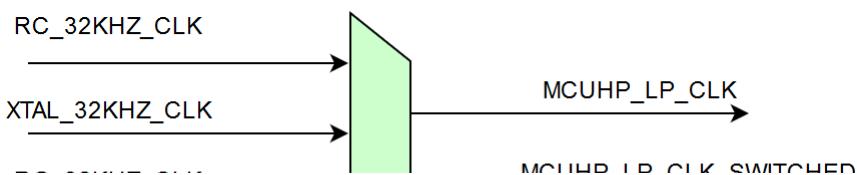
1. ICACHE
2. MCU HP DMA
3. Micro-DMA
4. Motor-Control PWM
5. Quadrature Encoder
6. I²C
7. SSI-Slave
8. Random-Number-Generator
9. CRC Accelerator
10. SIO
11. Enhanced-GPIO
12. eFUSE

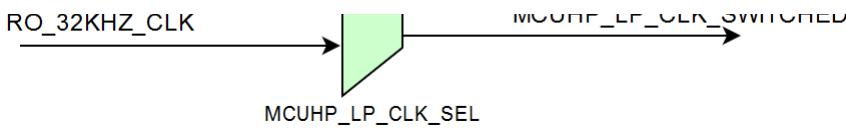
The following sections describe the clock architecture for each of the functionality mentioned above. The reference clock generated for MCU-HP domain (MCUHP_REF_CLK) will be used in generation of the clocks for different peripherals/modules.

4.11.4 Low-Power Clock

The source for low power clock is configured through MCUHP_LP_CLK_SEL in MCUHP_CLK_CONFIG_REG4 Register. The Clock switching status can be read from MCUHP_LP_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

This is also used as a low freq clock source for SDMEM(eMMC).



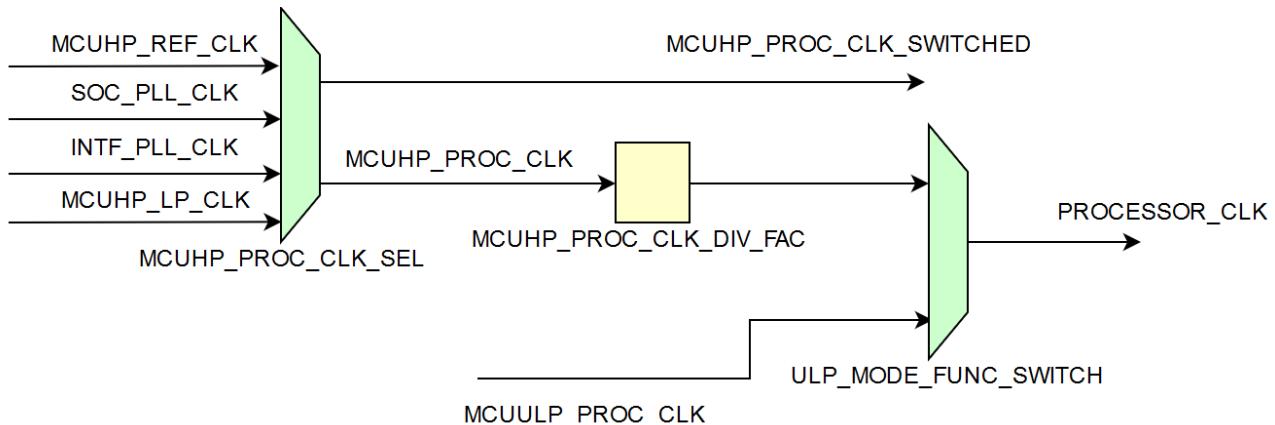


MCU-HP Low Power Clock Generation

4.11.5 Processor

The clock source and frequency for the Processor clock can be configured through MCUHP_PROC_CLK_SEL and MCUHP_PROC_CLK_DIV_DAC in MCUHP_CLK_CONFIG_REG5 Register. The Clock switching status can be read through MCUHP_PROC_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

The 2nd stage of Clock mux is used for switching from ultra-low power state(PS2) and is configured through ULP_MODE_FUNC_SWITCH which is described in [Power Architecture](#) Section. The MCUHP_PROC_CLK_SWITCHED status is valid only in PS4 state.

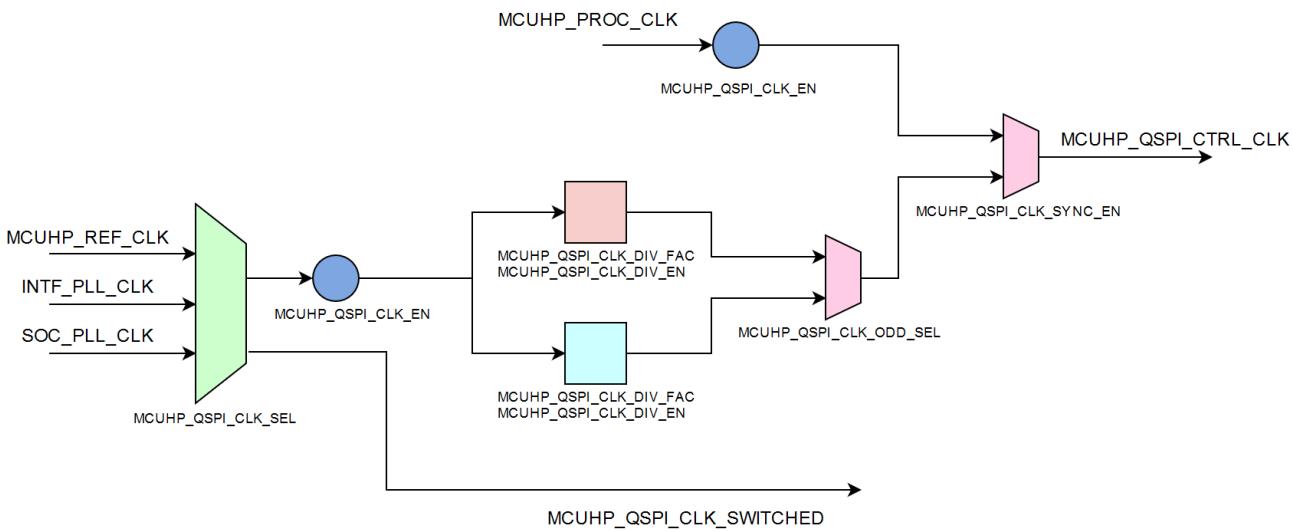


MCU-HP Processor Clock Generation

4.11.6 SPI Flash Controller

There are multiple modes of generating the clock for SPI Flash controller. It can be synchronous or independent of the Processor. The Clock switching status can be read through MCUHP_QSPI_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register. The status is valid only when Independent clock source is selected.

- Synchronous with Processor Clock. An undivided version of the Processor clock as shown in the Processor clock generation will be used in this mode.
 - Configure MCUHP_QSPI_CLK_SYNC_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register.
- Independent of Processor clock.
 - Clock source can be configured through MCUHP_QSPI_CLK_SEL in MCUHP_CLK_CONFIG_REG1 Register.
 - Division factor can be configured through MCUHP_QSPI_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG1 Register.
 - Output Clock with ODD/EVEN division factor can be selected through MCUHP_QSPI_CLK_ODD_SEL in MCUHP_CLK_CONFIG_REG2 Register.
- The clock to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and AHB clock can be controlled independently.
 - Configure MCUHP_QSPI_CLK_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register to enable/disable the controller clock.
 - Configure MCUHP_QSPI_AHB_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register to enable/disable the AHB Interface clock.
 - Configure MCUHP_QSPI_CLK_DIV_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register to enable/disable the QSPI Divider clocks.

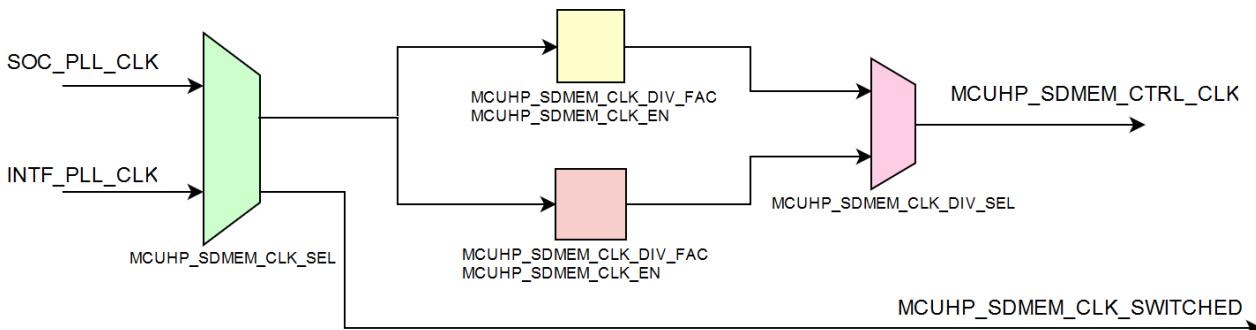


MCU-HP Flash Controller Clock Generation

4.11.7 SD-MEM (eMMC)

There are multiple modes of generating the clock for SD-MEM (eMMC) Controller. The Clock switching status can be read through MCUHP_SDMEM_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUHP_SDMEM_CLK_SEL in MCUHP_SDMEM_CLK_CONFIG Register.
 - Division factor can be configured through MCUHP_SDMEM_CLK_DIV_FAC in MCUHP_SDMEM_CLK_CONFIG Register.
 - Divided clock from a Clock swallow or Divider can be selected through MCUHP_SDMEM_CLK_DIV_SEL in MCUHP_SDMEM_CLK_CONFIG Register.
 - The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency.
 - Configure MCUHP_SDMEM_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register.

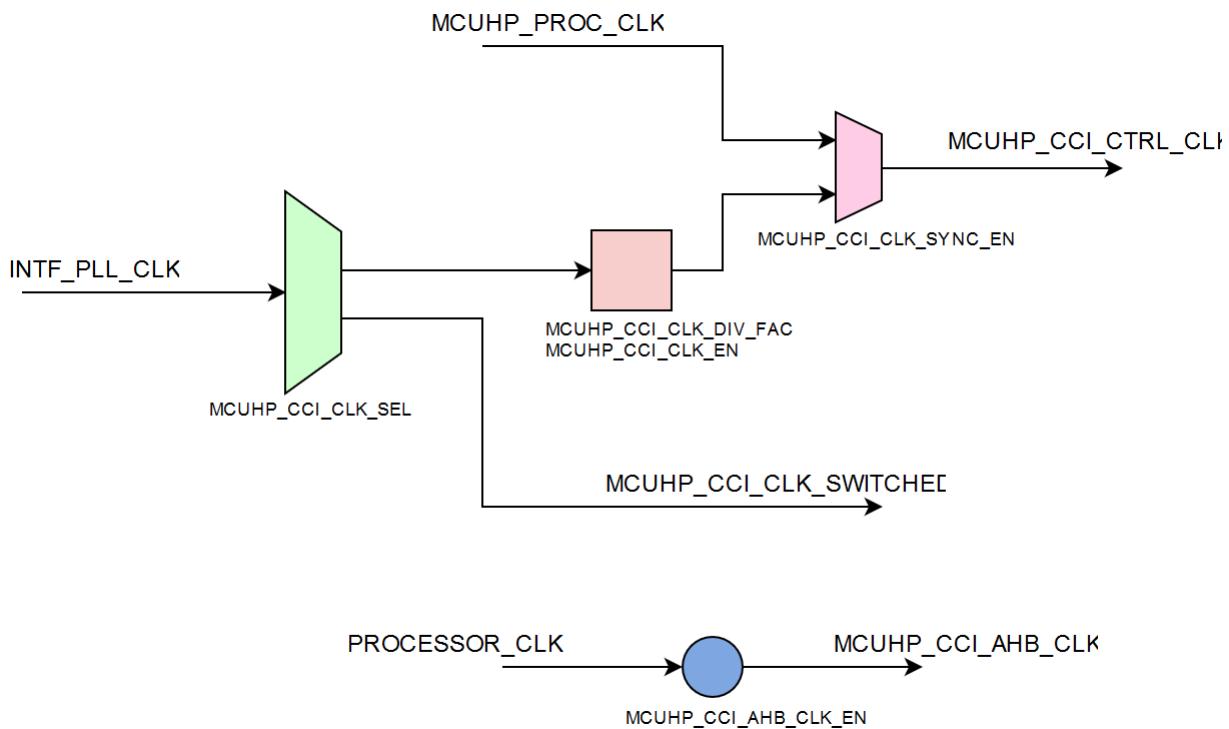


MCU-HP SDMEM Clock Generation

4.11.8 CCI

There are multiple modes of generating the clock for SD-MEM(eMMC) Controller. The Clock switching status can be read through MCUHP_CCI_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - A undivided version of the Processor clock used in PS4 state as shown in the Processor clock generation will be used as one of the clock source.
 - Configure MCUHP_CCI_CLK_SYNC_EN in MCUHP_MISC_CONFIG_1 Register
 - Clock source can be configured through MCUHP_CCI_CLK_SEL in MCUHP_CLK_CONFIG_REG4 Register.
 - Division factor can be configured through MCUHP_CCI_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG2 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and AHB clock can be controlled independently.
 - Configure MCUHP_CCI_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register for enabling/disabling the Controller clock.
 - Configure MCUHP_CCI_AHB_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register for enabling/disabling the AHB clock.



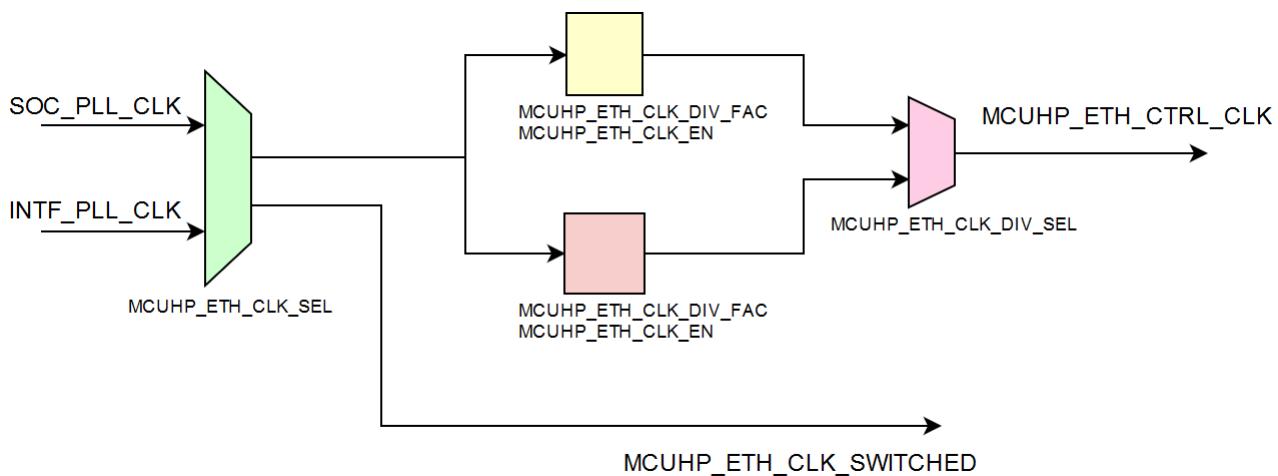
MCU-HP CCI Clock Generation

4.11.9 Ethernet RMII

There are multiple modes of generating the clock for Ethernet RMII Controller. The Clock switching status can be read through MCUHP_ETH_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation

- Clock source can be configured through MCUHP_ETH_CLK_SEL in MCUHP_CLK_CONFIG_REG1 Register.
- Division factor can be configured through MCUHP_ETH_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG1 Register.
- The divided clock can be selected through MCUHP_ETH_CLK_DIV_SEL in MCUHP_CLK_CONFIG_REG1 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and AHB clock can be controlled independently.
 - Configure MCUHP_ETH_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the Controller clock.

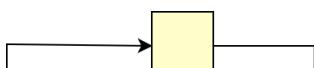


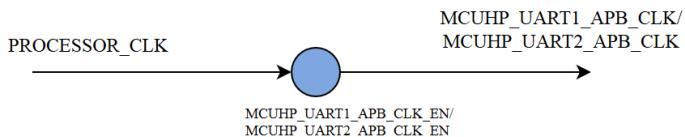
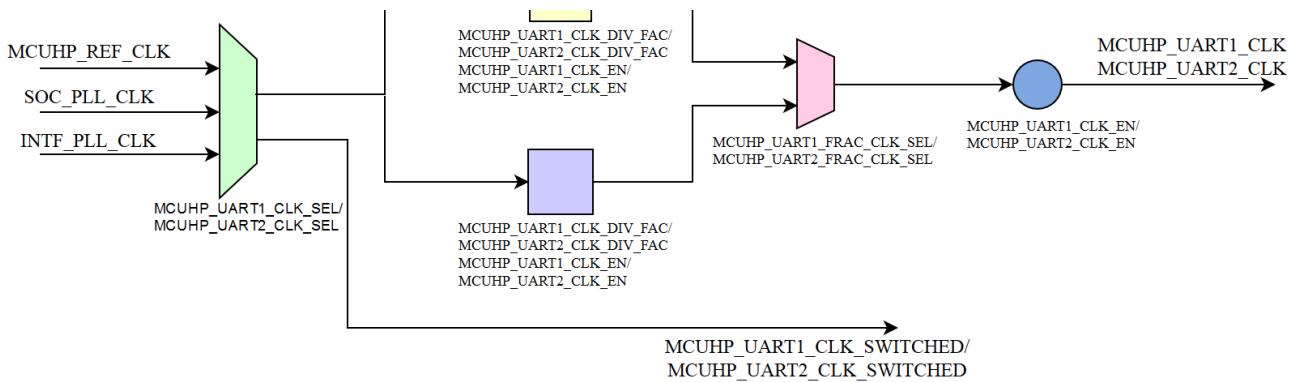
MCU-HP Ethernet Clock Generation

4.11.10 UART1/UART2

The clocking scheme is similar for UART1 & UART2 controller except for the configuration registers. There are multiple modes of generating the clock for UART1/UART2 Controller. The Clock switching status can be read through MCUHP_UART1_CLK_SWITCHED, MCUHP_UART2_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUHP_UART1_CLK_SEL, MCUHP_UART2_CLK_SEL in MCUHP_CLK_CONFIG_REG2 Register.
 - Division factor can be configured through MCUHP_UART1_CLK_DIV_FAC, MCUHP_UART2_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG2 Register.
 - Divided clock from a Clock Swallow or Fractional Divider can be selected through MCUHP_UART1_FRAC_CLK_SEL, MCUHP_UART2_FRAC_CLK_SEL in MCUHP_CLK_CONFIG_REG2 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and APB clock can be controlled independently.
 - Configure MCUHP_UART1_CLK_EN, MCUHP_UART2_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_CLEAR_REG1 Register for enabling/disabling the Controller clock.
 - Configure MCUHP_UART1_APB_CLK_EN, MCUHP_UART2_APB_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_CLEAR_REG1 Register for enabling/disabling the APB clock.



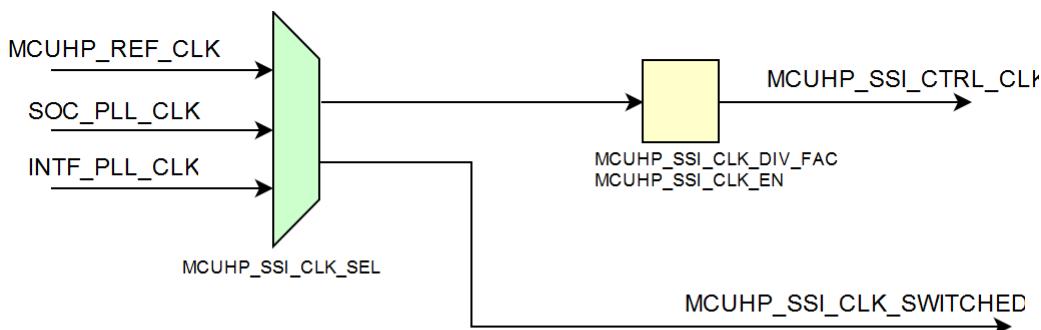


MCU-HP UART1/UART2 Clock Generation

4.11.11 SPI / SSI Master

There are multiple clock sources for SPI/SSI Master Controller. The Clock switching status can be read through MCUHP_SSI_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUHP_SSI_CLK_SEL in MCUHP_CLK_CONFIG_REG1 Register.
 - Division factor can be configured through MCUHP_SSI_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG1 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and APB clock can be controlled independently.
 - Configure MCUHP_SSI_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the Controller clock.
 - Configure MCUHP_SSI_APB_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the APB clock.



PROCESSOR_CLK

MCUHP_SSI_APB_CLK

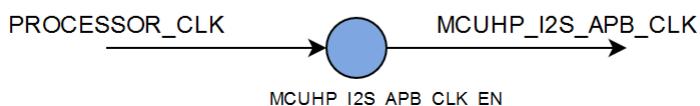
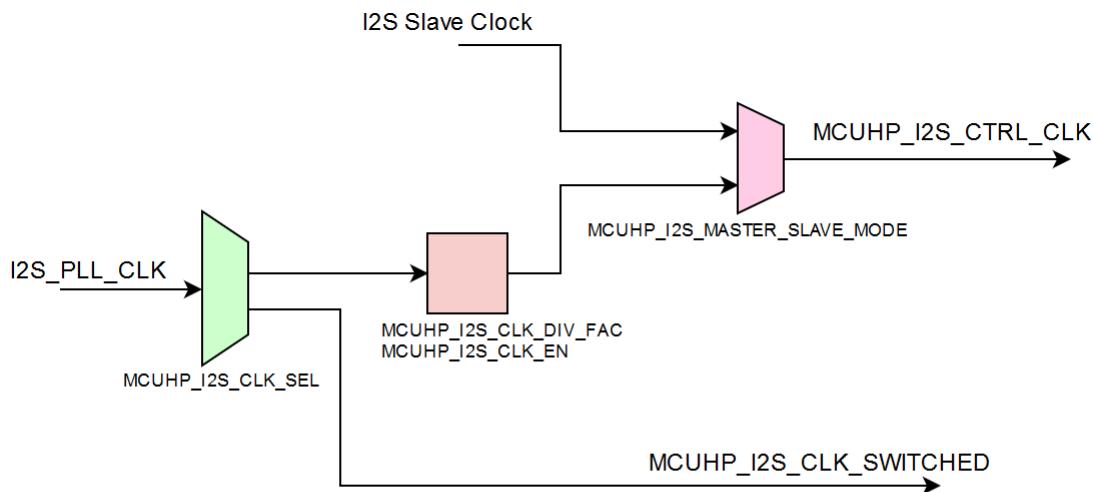
MCUHP_SSI_APB_CLK_EN

MCU-HP SPI/SSI Master Clock Generation

4.11.12 I²S Controller

There are multiple clock sources for I²S Controller which is used in Master Mode. The Clock switching status can be read through MCUHP_I2S_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - I²S Slave clock is derived from the external Master Device through GPIO PAD's.
 - Clock source can be configured through MCUHP_I2S_CLK_SEL in MCUHP_CLK_CONFIG_REG5 Register.
 - Division factor can be configured through MCUHP_I2S_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG5 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and APB clock can be controlled independently.
 - Configure MCUHP_I2S_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the Controller clock.
 - Configure MCUHP_I2S_APB_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the APB Interface clock.
- In addition to the above, the I²S Interface clock can be disabled.
 - Configure MCUHP_I2S_INTF_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the Interface clock.
- The Master/Slave mode is configured through MCUHP_I2S_MASTER_SLAVE_MODE in MCUHP_MISC_CONFIG_3 Register.

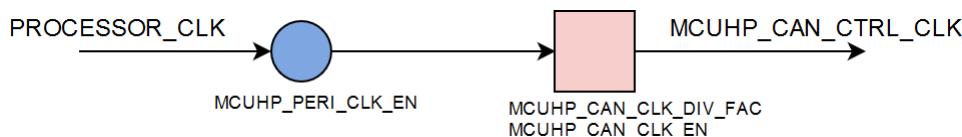


MCU-HP I2S Clock Generation

4.11.13 CAN Controller

There is only one clock source for Can Controller.

- Clock Generation
 - The Processor clock is used as the clock source.
 - This clock can be enabled/disabled through MCUHP_PERI_CLK_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register.
 - Division factor can be configured through MCUHP_CAN_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG3 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency.
 - Configure MCUHP_CAN_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the Controller clock.

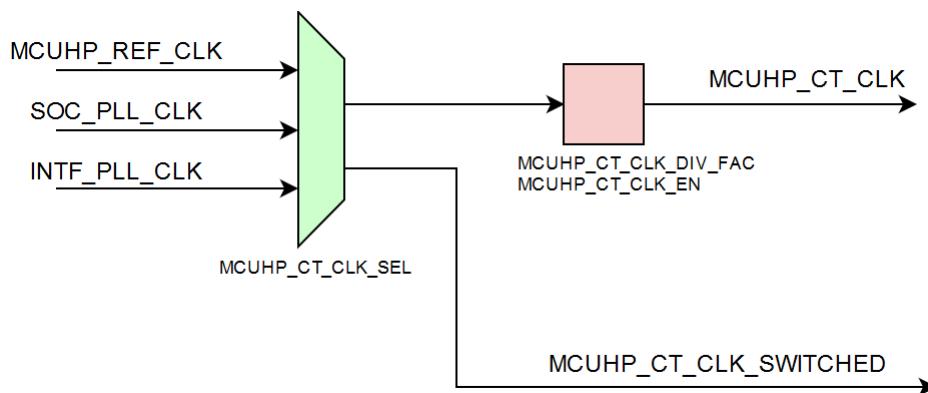


MCU-HP Can Clock Generation

4.11.14 Configurable Timers

There are multiple clock sources for Configurable Timers. The Clock switching status can be read through MCUHP_CT_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUHP_CT_CLK_SEL in MCUHP_CLK_CONFIG_REG5 Register.
 - Division factor can be configured through MCUHP_CT_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG5 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency.
 - Configure MCUHP_CT_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register for enabling/disabling the Controller clock.

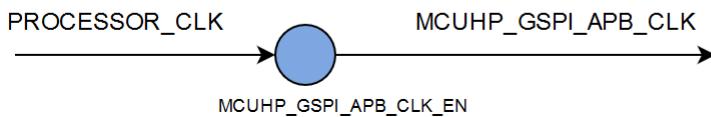
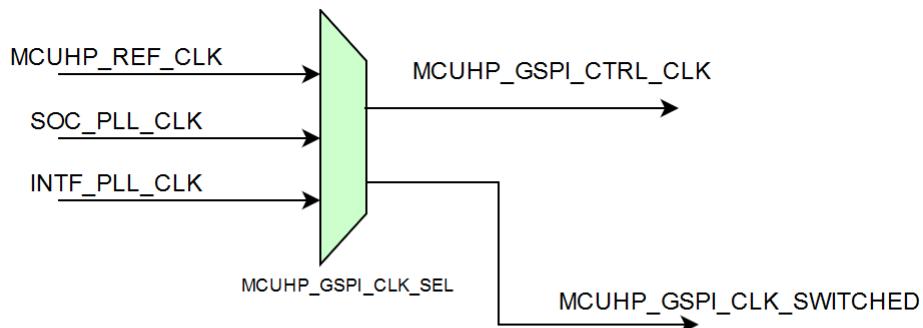


MCU-HP Configurable Timer Clock Generation

4.11.15 Generic SPI Master

There are multiple clock sources for Generic SPI Master Controller. The Clock switching status can be read through MCUHP_GSPI_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUHP_GSPI_CLK_SEL in MCUHP_CLK_CONFIG_REG1 Register.
- The clocks to the APB Interface can be disabled when not in use for efficient power consumption.
 - Configure MCUHP_GSPI_APB_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the Controller clock.

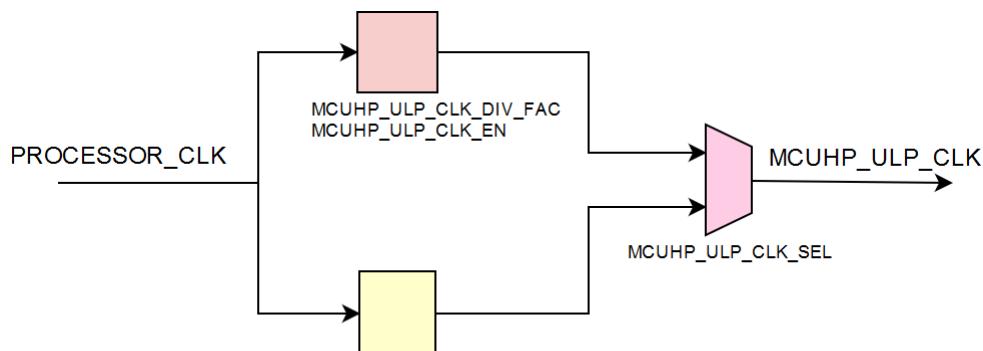


MCU-HP GSPI Clock Generation

4.11.16 MCU-ULP SoC Clock

There is only one clock source for MCU ULP Clock from MCU HP domain. This can be used only in PS4 power state.

- Clock Generation
 - The Processor clock used in PS4 state as shown in the Processor clock generation will be used as the clock source.
 - Division factor can be configured through MCUHP_ULP_DIV_FAC in MCUHP_CLK_CONFIG_REG4 Register.
 - The divided clock can be selected through MCUHP_ULP_CLK_SEL in MCUHP_CLK_CONFIG_REG5 Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency.
 - Configure MCUHP_CAN_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register.



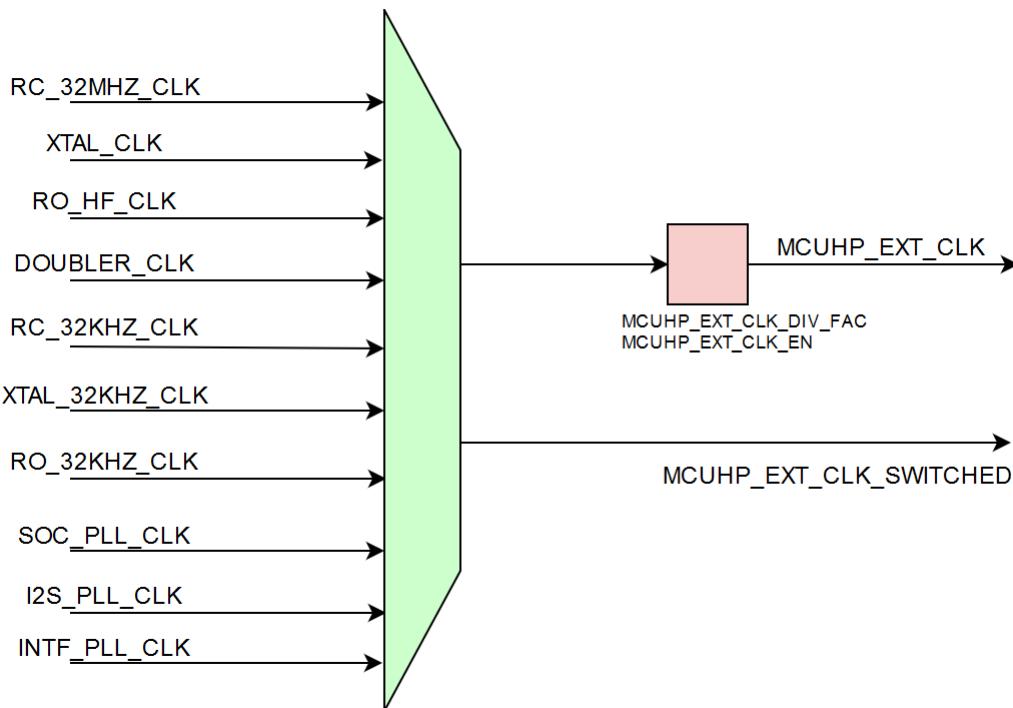
MCUHP_ULP_CLK_DIV_FAC
MCUHP_ULP_CLK_EN

MCU-HP ULP Clock Generation

4.11.17 External Clock

There are multiple source for generating the clock for External components through GPIO PAD's. The Clock switching status can be read through MCUHP_EXT_CLK_SWITCHED in MCUHP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUHP_EXT_CLK_SEL in MCUHP_CLK_CONFIG_REG3 Register.
 - Division factor can be configured through MCUHP_EXT_CLK_DIV_FAC in MCUHP_CLK_CONFIG_REG3 Register.
- The clocks to the external components can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency.
 - Configure MCUHP_EXT_CLK_EN in MCUHP_CLK_CONFIG_REG3 Register for enabling/disabling the clock.



MCU-HP External Clock Generation

4.11.18 Static Clock Gated Domains

The clock to the domains which operate on the processor clock can be disabled when not in use for efficient power management. Below mentioned are the programming details for the same.

- **ICACHE:** Configure MCUHP_ICACHE_CLK_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register for enabling/disabling the clock to ICACHE module.
- **MCU HP DMA:** Configure MCUHP_DMA_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register for enabling/disabling the clock to DMA module.
- **Random-Number-Generator:** Configure MCUHP_RNG_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register for enabling/disabling the clock to Random-Number-Generator module.

- **CRC Accelerator:** Configure MCUHP_CRC_CLK_EN in MCUHP_CLK_EN_SET_REG1/MCUHP_CLK_EN_CLEAR_REG1 Register for enabling/disabling the clock to CRC Accelerator module.
- **Micro-DMA:** Configure MCUHP_UDMA_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the clock to Micro-DMA module.
- **Motor-Control PWM:** Configure MCUHP_MCPWM_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the clock to Motor-Control PWM module.
- **Quadrature Encoder:** Configure MCUHP_QE_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the clock to Quadrature Encoder module.
- **I²C - 2x:**
 - Configure MCUHP_I2C2_APB_CLK_EN, MCUHP_I2C_APB_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the clock to I²C APB Interface.
 - Configure MCUHP_I2C2_CLK_EN, MCUHP_I2C_CLK_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register for enabling/disabling the clock to I²C Controller.
- **SSI-Slave:**
 - Configure MCUHP_SSI_SLV_APB_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the clock to I²C APB Interface.
 - Configure MCUHP_SSI_SLV_CLK_EN in MCUHP_CLK_EN_SET_REG2/MCUHP_CLK_EN_CLEAR_REG2 Register for enabling/disabling the clock to I²C Controller.
- **SIO:** Configure MCUHP_SIO_CLK_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register for enabling/disabling the clock to SIO module.
- **Enhanced-GPIO:** Configure MCUHP_EGPIO_CLK_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register for enabling/disabling the clock to Enhanced-GPIO module.
- **eFUSE:** Configure MCUHP_EFUSE_CLK_EN in MCUHP_CLK_EN_SET_REG3/MCUHP_CLK_EN_CLEAR_REG3 Register for enabling/disabling the clock to eFUSE module.

4.11.19 Register Summary

Base Address: 0x4600_0000

| Register Name | Offset | Description |
|------------------------|--------|---|
| MCUHP_CLKEN_SET_REG1 | 0x00 | Clock Enable Set Register 1 |
| MCUHP_CLKEN_CLEAR_REG1 | 0x04 | Clock Enable Clear Register 1 |
| MCUHP_CLKEN_SET_REG2 | 0x08 | Clock Enable Set Register 2 |
| MCUHP_CLKEN_CLEAR_REG2 | 0x0C | Clock Enable Clear Register 2 |
| MCUHP_CLKEN_SET_REG3 | 0x10 | Clock Enable Set Register 3 |
| MCUHP_CLKEN_CLEAR_REG3 | 0x14 | Clock Enable Clear Register 3 |
| MCUHP_CLK_CONFIG_REG1 | 0x18 | Clock Configuration Register 1 |
| MCUHP_CLK_CONFIG_REG2 | 0x1C | Clock Configuration Register 2 |
| MCUHP_CLK_CONFIG_REG3 | 0x20 | Clock Configuration Register 3 |
| MCUHP_CLK_CONFIG_REG4 | 0x24 | Clock Configuration Register 4 |
| MCUHP_CLK_CONFIG_REG5 | 0x28 | Clock Configuration Register 5 |
| MCUHP_SDMEM_CLK_CONFIG | 0x40 | SDMEM Clock Configuration Register |
| MCUHP_CLK_STATUS_REG | 0x58 | Clock Status Register for Dynamic Clock Muxes |

Base Address: 0x 4600_8000

| Register Name | Offset | Description |
|---------------|--------|---|
| | 0x14 | Miscellaneous Clock Configuration Register1 |

| Register Name | Offset | Description |
|---------------------|--------|---|
| MCUHP_MISC_CONFIG_3 | 0x44 | Miscellaneous Clock Configuration Register3 |

150 . List of Registers

4.11.20 Register Description

MCUHP_CLKEN_SET_REG1

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------------|---------------|---|
| 31 | R/W | MCUHP_ULP_CLK_EN | 0 | Writing 1 to this enables clock to MCU-ULPDomain. Writing 0 to this has no effect. |
| 30 | R/W | Reserved | 0 | It is recommended to write these bits to 0. |
| 29 | R/W | | 0 | Writing 1 to this enables clock to SD-MEM Controller. Writing 0 to this has no effect. |
| 28:27 | R/W | Reserved | - | It is recommended to write these bits to 0. |
| 26 | R/W | MCUHP_CCI_CLK_EN | 0 | Writing 1 to this enables clock to CCI Controller. Writing 0 to this has no effect. |
| 25 | R/W | MCUHP_CCI_AHB_CLK_EN | 0 | Writing 1 to this enables clock to CCI AHB Interface. Writing 0 to this has no effect. |
| 24:23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | R/W | MCUHP_RNG_CLK_EN | 0 | Writing 1 to this enables clock to Random-Number-Generator. Writing 0 to this has no effect. |
| 21 | R/W | MCUHP_ETH_AHB_CLK_EN | 0 | Writing 1 to this enables clock to Ethernet Controller AHB Interface. Writing 0 to this has no effect. |
| 20:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | MCUHP_CRC_CLK_EN | 0 | Writing 1 to this enables clock to CRC Accelerator. Writing 0 to this has no effect. |
| 17:14 | - | Reserved | - | It is recommended to write these bits to 0. |
| 13 | R/W | MCUHP_DMA_CLK_EN | 1 | Writing 1 to this enables clock to DMA. Writing 0 to this has no effect. |
| 12:10 | - | Reserved | - | It is recommended to write these bits to 0. |
| 9 | R/W | MCUHP_CT_CLK_EN | 0 | Writing 1 to this enables clock to Configurable Timers. Writing 0 to this has no effect. |
| 8:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | R/W | MCUHP_UART2_CLK_EN | 0 | Writing 1 to this enables clock to UART2 Controller. Writing 0 to this has no effect. |

| Bit | Access | Function | Default Value | Description |
|-----|--------|------------------------|---------------|---|
| 2 | R/W | MCUHP_UART2_APB_CLK_EN | 0 | Writing 1 to this enables clock to UART2 APB Interface. Writing 0 to this has no effect. |
| 1 | R/W | MCUHP_UART1_CLK_EN | 0 | Writing 1 to this enables clock to UART1 Controller. Writing 0 to this has no effect. |
| 0 | R/W | MCUHP_UART1_APB_CLK_EN | 0 | Writing 1 to this enables clock to UART1 APB Interface. Writing 0 to this has no effect. |

151 . MCUHP_CLKEN_SET_REG1 Description

MCUHP_CLKEN_CLEAR_REG1

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------------|---------------|--|
| 31 | R/W | MCUHP_ULP_CLK_EN | 0 | Writing 1 to this disables clock to MCU-ULPDomain. Writing 0 to this has no effect. |
| 30 | R/W | Reserved | 0 | It is recommended to write these bits to 0. |
| 29 | R/W | | 0 | Writing 1 to this disables clock to SD-MEM Controller. Writing 0 to this has no effect. |
| 28:27 | R/W | Reserved | - | It is recommended to write these bits to 0. |
| 26 | R/W | MCUHP_CCI_CLK_EN | 0 | Writing 1 to this disables clock to CCI Controller. Writing 0 to this has no effect. |
| 25 | R/W | MCUHP_CCI_AHB_CLK_EN | 0 | Writing 1 to this disables clock to CCI AHB Interface. Writing 0 to this has no effect. |
| 24:23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | R/W | MCUHP_RNG_CLK_EN | 0 | Writing 1 to this disables clock to Random-Number-Generator. Writing 0 to this has no effect. |
| 21 | R/W | MCUHP_ETH_AHB_CLK_EN | 0 | Writing 1 to this disables clock to Ethernet Controller AHB Interface. Writing 0 to this has no effect. |
| 20:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | MCUHP_CRC_CLK_EN | 0 | Writing 1 to this disables clock to CRC Accelerator. Writing 0 to this has no effect. |
| 17:14 | - | Reserved | - | It is recommended to write these bits to 0. |
| 13 | R/W | MCUHP_DMA_CLK_EN | 1 | Writing 1 to this disables clock to DMA. Writing 0 to this has no effect. |
| 12:10 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Default Value | Description |
|------------|---------------|--------------------------|----------------------|--|
| 9 | R/W | MCUHP_CT_CLK_EN | 0 | Writing 1 to this disables clock to Configurable Timers. Writing 0 to this has no effect. |
| 8:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | R/W | MCUHP_UART2_CLK_EN | 0 | Writing 1 to this disables clock to UART2 Controller. Writing 0 to this has no effect. |
| 2 | R/W | MCUHP_UART2_APB_CLK_0 EN | | Writing 1 to this disables clock to UART2 APB Interface. Writing 0 to this has no effect. |
| 1 | R/W | MCUHP_UART1_CLK_EN | 0 | Writing 1 to this disables clock to UART1 Controller. Writing 0 to this has no effect. |
| 0 | R/W | MCUHP_UART1_APB_CLK_0 EN | | Writing 1 to this disables clock to UART1 APB Interface. Writing 0 to this has no effect. |

152 . MCUHP_CLKEN_CLEAR_REG1 Description

MCUHP_CLKEN_SET_REG2

| Bit | Access | Function | Default Value | Description |
|------------|---------------|------------------------|----------------------|--|
| 31:29 | R/W | Reserved | - | It is recommended to write these bits to 0. |
| 28 | R/W | | 0 | Writing 1 to this enables clock to Ethernet Controller. Writing 0 to this has no effect. |
| 27:25 | - | Reserved | - | It is recommended to write these bits to 0. |
| 24 | R/W | MCUHP_SSI_CLK_EN | 0 | Writing 1 to this enables clock to SPI/SSI Master Controller. Writing 0 to this has no effect. |
| 23 | R/W | MCUHP_SSI_APB_CLK_E0 N | | Writing 1 to this enables clock to SPI/SSI Master APN Interface. Writing 0 to this has no effect. |
| 22:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | MCUHP_MCPWM_CLK_E0 N | | Writing 1 to this enables clock to Motor-Control PWM. Writing 0 to this has no effect. |
| 17 | R/W | MCUHP_QE_CLK_EN | 0 | Writing 1 to this enables clock to Quadrature Encoder. Writing 0 to this has no effect. |
| 16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15 | R/W | MCUHP_I2S_APB_CLK_E0 N | | Writing 1 to this enables clock to I ² S APB Interface. Writing 0 to this has no effect. |

| Bit | Access | Function | Default Value | Description |
|-----|--------|--------------------------|---------------|--|
| 14 | R/W | MCUHP_I2S_INTF_CLK_EN | 0 | Writing 1 to this enables clock to I ² S Interface. Writing 0 to this has no effect. |
| 13 | R/W | MCUHP_I2S_CLK_EN | 0 | Writing 1 to this enables clock to I ² S Controller in Master Mode. Writing 0 to this has no effect. |
| 12 | R/W | MCUHP_QSPI_AHB_CLK_EN | 1 | Writing 1 to this enables clock to AHB Interface for SPI Flash Controller. Writing 0 to this has no effect. |
| 11 | R/W | MCUHP_QSPI_CLK_DIV_EN | 1 | Writing 1 to this enables clock to Clock dividers for SPI Flash Controller. Writing 0 to this has no effect. |
| 10 | R/W | MCUHP_SSI_SLV_CLK_EN | 0 | Writing 1 to this enables clock to SSI Slave Controller. Writing 0 to this has no effect. |
| 9 | R/W | MCUHP_SSI_SLV_APB_CLK_EN | 0 | Writing 1 to this enables clock to SSI Slave APB Interface. Writing 0 to this has no effect. |
| 8 | R/W | MCUHP_I2C2_APB_CLK_EN | 0 | Writing 1 to this enables clock to I ² C-2 APB Interface. Writing 0 to this has no effect. |
| 7 | R/W | MCUHP_I2C_APB_CLK_EN | 0 | Writing 1 to this enables clock to I ² C-1 APB Interface. Writing 0 to this has no effect. |
| 6 | R/W | MCUHP_UDMA_CLK_EN | 0 | Writing 1 to this enables clock to Micro-DMA. Writing 0 to this has no effect. |
| 5:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | R/W | MCUHP_CAN_CLK_EN | 0 | Writing 1 to this enables clock to CAN Controller. Writing 0 to this has no effect. |
| 2:1 | - | Reserved | - | It is recommended to write these bits to 0. |
| 0 | R/W | MCUHP_GSPI_APB_CLK_EN | 0 | Writing 1 to this enables clock to Generic-SPI Master APB Interface. Writing 0 to this has no effect. |

153 .MCUHP_CLKEN_SET_REG2 Description

MCUHP_CLKEN_CLEAR_REG2

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------|---------------|---|
| 31:29 | R/W | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Default Value | Description |
|-------|--------|--------------------------------|---------------|---|
| 28 | R/W | | 0 | Writing 1 to this disables clock to Ethernet Controller. Writing 0 to this has no effect. |
| 27:25 | - | Reserved | - | It is recommended to write these bits to 0. |
| 24 | R/W | MCUHP_SSI_CLK_EN | 0 | Writing 1 to this disables clock to SPI/SSI Master Controller. Writing 0 to this has no effect. |
| 23 | R/W | MCUHP_SSI_APB_CLK_E 0 N | | Writing 1 to this disables clock to SPI/SSI Master APB Interface. Writing 0 to this has no effect. |
| 22:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | MCUHP_MCPWM_CLK_E 0 N | | Writing 1 to this disables clock to Motor-Control PWM. Writing 0 to this has no effect. |
| 17 | R/W | MCUHP_QE_CLK_EN | 0 | Writing 1 to this disables clock to Quadrature Encoder. Writing 0 to this has no effect. |
| 16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15 | R/W | MCUHP_I2S_APB_CLK_E 0 N | | Writing 1 to this disables clock to I ² S APB Interface. Writing 0 to this has no effect. |
| 14 | R/W | MCUHP_I2S_INTF_CLK_ 0 EN | | Writing 1 to this disables clock to I ² S Interface. Writing 0 to this has no effect. |
| 13 | R/W | MCUHP_I2S_CLK_EN | 0 | Writing 1 to this disables clock to I ² S Controller in Master Mode. Writing 0 to this has no effect. |
| 12 | R/W | MCUHP_QSPI_AHB_CLK_ 1 _EN | | Writing 1 to this disables clock to AHB Interface for SPI Flash Controller. Writing 0 to this has no effect. |
| 11 | R/W | MCUHP_QSPI_CLK_DIV_ 1 EN | | Writing 1 to this disables clock to Clock dividers for SPI Flash Controller. Writing 0 to this has no effect. |
| 10 | R/W | MCUHP_SSI_SLV_CLK_E 0 N | | Writing 1 to this disables clock to SSI Slave Controller. Writing 0 to this has no effect. |
| 9 | R/W | MCUHP_SSI_SLV_APB_C 0 LK_EN | | Writing 1 to this disables clock to SSI Slave APB Interface. Writing 0 to this has no effect. |

| Bit | Access | Function | Default Value | Description |
|-----|--------|----------------------------|---------------|---|
| 8 | R/W | MCUHP_I2C2_APB_CLK_0 EN | | Writing 1 to this disables clock to I ² C-2 APB Interface. Writing 0 to this has no effect. |
| 7 | R/W | MCUHP_I2C_APB_CLK_E_0 N | | Writing 1 to this disables clock to I ² C-1 APB Interface. Writing 0 to this has no effect. |
| 6 | R/W | MCUHP_UDMA_CLK_EN | 0 | Writing 1 to this disables clock to Micro-DMA. Writing 0 to this has no effect. |
| 5:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | R/W | MCUHP_CAN_CLK_EN | 0 | Writing 1 to this disables clock to CAN Controller. Writing 0 to this has no effect. |
| 2:1 | - | Reserved | - | It is recommended to write these bits to 0. |
| 0 | R/W | MCUHP_GSPI_APB_CLK _EN | 0 | Writing 1 to this disables clock to Generic-SPI Master APB Interface. Writing 0 to this has no effect. |

154 . MCUHP_CLKEN_CLEAR_REG2 Description

MCUHP_CLKEN_SET_REG3

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------------------|---------------|---|
| 31:28 | - | Reserved | - | It is recommended to write these bits to 0. |
| 27 | R/W | MCUHP_ICACHE_CLK_ EN | 1 | Writing 1 to this enables clock to Generic-SPI Master. Writing 0 to this has no effect. |
| 26 | R/W | | 0 | Writing 1 to this enables clock source to Peripherals. Writing 0 to this has no effect. |
| 25:21 | R/W | Reserved | - | It is recommended to write these bits to 0. |
| 20 | R/W | MCUHP_SIO_CLK_EN | 0 | Writing 1 to this enables clock to SIO Controller. Writing 0 to this has no effect. |
| 19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | MCUHP_I2C2_CLK_EN | 0 | Writing 1 to this enables clock to I ² C-2 Controller. Writing 0 to this has no effect. |
| 17 | R/W | MCUHP_I2C_CLK_EN | 0 | Writing 1 to this enables clock to I ² C-1 Controller. Writing 0 to this has no effect. |
| 16 | R/W | MCUHP_EGPIO_CLK_E N | 0 | Writing 1 to this enables clock to Enhanced-GPIO Controller. Writing 0 to this has no effect. |
| 15 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Default Value | Description |
|------|--------|-------------------------|---------------|---|
| 14 | R/W | MCUHP_QSPI_CLK_SY NC_EN | 0 | Writing 1 to this enables SPI Flash Controller clock in synchronous with Processor Clock. Writing 0 to this has no effect. |
| 13 | R/W | MCUHP_QSPI_CLK_EN | 1 | Writing 1 to this enables clock to SPI Flash Controller. Writing 0 to this has no effect. |
| 12:6 | R/W | Reserved | - | It is recommended to write these bits to 0. |
| 5 | R/W | MCUHP_EFUSE_CLK_E N | 1 | Writing 1 to this enables clock to eFUSE Controller. Writing 0 to this has no effect. |
| 4:0 | - | Reserved | - | It is recommended to write these bits to 0. |

155 . MCUHP_CLKEN_SET_REG3 Description

MCUHP_CLKEN_CLEAR_REG3

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------------------|---------------|--|
| 31:28 | - | Reserved | - | It is recommended to write these bits to 0. |
| 27 | R/W | MCUHP_ICACHE_CLK_EN | 1 | Writing 1 to this disables clock to Generic-SPI Master. Writing 0 to this has no effect. |
| 26 | R/W | | 0 | Writing 1 to this disables clock source to Peripherals. Writing 0 to this has no effect. |
| 25:21 | R/W | Reserved | - | It is recommended to write these bits to 0. |
| 20 | R/W | MCUHP_SIO_CLK_EN | 0 | Writing 1 to this disables clock to SIO Controller. Writing 0 to this has no effect. |
| 19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | MCUHP_I2C2_CLK_EN | 0 | Writing 1 to this disables clock to I ² C-2 Controller. Writing 0 to this has no effect. |
| 17 | R/W | MCUHP_I2C_CLK_EN | 0 | Writing 1 to this disables clock to I ² C-1 Controller. Writing 0 to this has no effect. |
| 16 | R/W | MCUHP_EGPIO_CLK_E N | 0 | Writing 1 to this disables clock to Enhanced-GPIO Controller. Writing 0 to this has no effect. |
| 15 | - | Reserved | - | It is recommended to write these bits to 0. |
| 14 | R/W | MCUHP_QSPI_CLK_SY NC_EN | 0 | Writing 1 to this disables SPI Flash Controller clock in synchronous with Processor Clock. Writing 0 to this has no effect. |
| 13 | R/W | MCUHP_QSPI_CLK_EN | 1 | Writing 1 to this disables clock to SPI Flash Controller. Writing 0 to this has no effect. |
| 12:6 | R/W | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Default Value | Description |
|-----|--------|--------------------|---------------|---|
| 5 | R/W | MCUHP_EFUSE_CLK_EN | 1 | Writing 1 to this disables clock to eFUSE Controller. Writing 0 to this has no effect. |
| 4:0 | - | Reserved | - | It is recommended to write these bits to 0. |

156 .MCUHP_CLKEN_CLEAR_REG3 Description

MCUHP_CLK_CONFIG_REG1

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------------------|---------------|---|
| 31:27 | - | Reserved | - | It is recommended to write these bits to 0. |
| 26:24 | R/W | | 7 | Specifies the clock source for Ethernet RMII Controller. 0 - Reserved 1 - MCU-HP Reference Clock 2 - SoC-PLL Clock 3 - Reserved 4 - Interface-PLL Clock 5,6,7 - Output clock is gated |
| 23 | R/W | MCUHP_ETH_C_LK_DIV_SEL | 0 | Selects the Divider type for Ethernet Controller. 0 - EVEN Clock Divider output is selected 1 - Clock Divider output is selected |
| 22:19 | R/W | MCUHP_ETH_C_LK_DIV_FAC | 2 | Specifies the clock division factor for Ethernet RMII Controller. |
| 18 | R/W | MCUHP_ETH_C_LK_SEL | 0 | Specifies the clock source for Ethernet RMII Controller. 0 - Interface-PLL Clock 1 - SoC-PLL Clock |
| 17:15 | R/W | MCUHP_SSI_CLK_SEL | 7 | Specifies the clock source for SPI/SSI Master. 0 - MCU-HP Reference Clock 1 - SoC-PLL Clock 2 - Reserved 3 - Interface-PLL Clock 4,5 - Reserved 6,7 - Output Clock is gated |
| 14:11 | R/W | MCUHP_SSI_CLK_DIV_FAC | 1 | Specifies the clock division factor for SPI/SSI Master. |

| Bit | Access | Function | Default Value | Description |
|------------|---------------|--------------------------|----------------------|---|
| 10:9 | - | Reserved | - | It is recommended to write these bits to 0. |
| 8:3 | R/W | MCUHP_QSPI_C1_LK_DIV_FAC | | Specifies the clock division factor for SPI Flash Controller. |
| 2:0 | R/W | MCUHP_QSPI_C0_LK_SEL | | Specifies the clock source for SPI Flash controller when independent clock source w.r.t Processor is selected. 0 - MCU-HP Reference Clock 1 - Interface-PLL Clock 2 - Reserved 3 - SoC-PLL Clock 4,5,6,7 - Output Clock is gated |

157 . MCUHP_CLK_CONFIG_REG1 Description

MCUHP_CLK_CONFIG_REG2

| Bit | Access | Function | Default Value | Description |
|------------|---------------|----------------------------|----------------------|---|
| 31 | - | Reserved | - | It is recommended to write these bits to 0. |
| 30 | R/W | | 0 | Selects the Divider type for UART2 Controller. 0 - Clock Swallow output is selected 1 - Fractional Clock Divider output is selected |
| 29 | R/W | MCUHP_UART1_FRAC_0_CLK_SEL | 0 | Selects the Divider type for UART1 Controller. 0 - Clock Swallow output is selected 1 - Fractional Clock Divider output is selected |
| 28 | R/W | MCUHP_QSPI_CLK_O_0_DD_SEL | 0 | Selects the Divider type for SPI Flash Controller when independent clock source w.r.t Processor is selected. 0 - EVEN Clock Divider output is selected 1 - ODD Clock Divider output is selected |
| 27:24 | R/W | MCUHP_CCI_CLK_DIV_1_FAC | 1 | Specifies the clock division factor for CCI Controller. |
| 23:14 | - | Reserved | - | It is recommended to write these bits to 0. |
| 13:10 | R/W | MCUHP_UART2_CLK_1_DIV_FAC | 1 | Specifies the clock division factor for UART2 Controller. |

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-------------------------|----------------------|---|
| 9:7 | R/W | MCUHP_UART2_CLK_SEL | 7 | <p>Specifies the clock source to be used for UART2 Controller.</p> <p>0 - MCU-HP Reference Clock</p> <p>1 - SoC-PLL Clock</p> <p>2 - Reserved</p> <p>3 - Interface-PLL Clock</p> <p>4 - Reserved</p> <p>5,6,7 - Clock is gated</p> |
| 6:3 | R/W | MCUHP_UART1_CLK_DIV_FAC | 1 | Specifies the clock division factor for UART1 Controller. |
| 2:0 | R/W | MCUHP_UART1_CLK_SEL | 7 | <p>Specifies the clock source to be used for UART1 Controller.</p> <p>0 - MCU-HP Reference Clock</p> <p>1 - SoC-PLL Clock</p> <p>2 - Reserved</p> <p>3 - Interface-PLL Clock</p> <p>4 - Reserved</p> <p>5,6,7 - Output Clock is gated</p> |

158 . MCUHP_CLK_CONFIG_REG2 Description

MCUHP_CLK_CONFIG_REG3

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------------|----------------------|--|
| 31:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | | 0 | <p>Writing 1 to this enables the External clock on GPIO PAD.</p> <p>Writing 0 to this disables the External clock on GPIO PAD.</p> |
| 17:12 | R/W | MCUHP_EXT_CLK_DIV_FAC | 0 | Specifies the clock division factor for External Clock. |

| Bit | Access | Function | Default Value | Description |
|------|--------|-------------------------|---------------|---|
| 11:8 | R/W | MCUHP_EXT_CLK_SEL | 0 | <p>Specifies the clock source to be used for External Clock.</p> <p>0 - Output Clock is Gated</p> <p>1 - High Freq RC Clock source</p> <p>2 - XTAL Clock source.</p> <p>3 - Reserved</p> <p>4 - High Freq RO Clock source</p> <p>5 - Doubler Clock</p> <p>6 - Reserved</p> <p>7 - Low Freq RC Clock source</p> <p>8 - Low Freq XTAL Clock source</p> <p>9 - Low Freq RO Clock source</p> <p>10 - Interface-PLL Clock</p> <p>11,12 - Reserved</p> <p>13 - SoC-PLL Clock</p> <p>14 - I2S-PLL Clock</p> <p>15 - Reserved</p> |
| 7:0 | R/W | MCUHP_CAN_CLK_DIV_F1_AC | | Specifies the clock division factor for CAN Controller. |

159 . MCUHP_CLK_CONFIG_REG3 Description

MCUHP_CLK_CONFIG_REG4

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------------|---------------|--|
| 31 | R/W | | 0 | <p>Specifies the clock source to be used for Low-Power Clock.</p> <p>0 - Reserved</p> <p>1 - Interface-PLL Clock</p> |
| 30:25 | R/W | MCUHP_ULP_DIV_FAC | 4 | Specifies the clock division factor for MCU ULP Domain source. |
| 24:23 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |

| Bit | Access | Function | Default Value | Description |
|-------|--------|--------------------|---------------|--|
| 22:21 | R/W | MCUHP_LP_CLK_2_SEL | 2 | <p>Specifies the clock source to be used for Low-Power Clock.</p> <p>0 - Low Freq RC Clock source</p> <p>1 - Low Freq XTAL Clock source</p> <p>2 - Output Clock is gated</p> <p>3 - Low Freq RO Clock source</p> |
| 20:0 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |

160 . MCUHP_CLK_CONFIG_REG4 Description

MCUHP_CLK_CONFIG_REG5

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------------------|---------------|--|
| 30:29 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |
| 28 | R/W | | 0 | <p>Specifies the divider type to be selected to MCU ULP Domain source.</p> <p>0 - Clock Divider output is selected</p> <p>1 - Odd Clock Divider output is selected</p> |
| 27:26 | - | Reserved | - | It is recommended to write these bits to 0. |
| 25:20 | R/W | MCUHP_CT_CLK1_DIV_FAC | | Specifies the clock division factor for Configurable Timers. |
| 19:17 | R/W | MCUHP_CT_CLK7_SEL | | <p>Specifies the clock source to be used for Configurable Timers.</p> <p>0 - MCU-HP Reference Clock</p> <p>1 - Interface-PLL Clock</p> <p>2 - SoC-PLL Clock</p> <p>3 - Reserved</p> <p>4,5,6,7 - Output Clock is gated</p> |
| 16:11 | R/W | MCUHP_I2S_CLK1_DIV_FAC | | Specifies the clock division factor for I ² S Controller in Master Mode. |
| 10 | R/W | MCUHP_I2S_CLK0_SEL | | Specifies the clock source to be used for I ² S Controller in Master Mode. |
| 9:4 | R/W | MCUHP_PROC_CLK_DIV_DAC | 1 | Specifies the clock division factor for Processor Clock. |

| Bit | Access | Function | Default Value | Description |
|-----|--------|--------------------|---------------|--|
| 3:0 | R/W | MCUHP_PROC_CLK_SEL | 0 | <p>Specifies the clock source to be used for Processor.</p> <p>0 - MCU-HP Reference Clock</p> <p>1 - Reserved</p> <p>2 - SoC-PLL Clock</p> <p>3 - Reserved</p> <p>4 - Interface-PLL Clock</p> <p>5 - Low-Power Clock</p> <p>6,7 - Reserved</p> |

161 . MCU_CLK_CONFIG_REG5 Description

MCUHP_SDMEM_CLK_CONFIG

| Bit | Access | Function | Default Value | Description |
|-------|--------|---------------------------|---------------|---|
| 31:10 | - | Reserved | - | It is recommended to write these bits to 0. |
| 9 | R/W | | 0 | <p>Specifies the divider type to be selected to SDMEM Controller</p> <p>0 - EVEN Clock Divider output is selected</p> <p>1 - Clock Divider output is selected</p> |
| 8:6 | R/W | MCUHP_SDMEM_CLK_0_SEL | | <p>Specifies the clock source to be used for SD-MEM(eMMC).</p> <p>0 - SoC-PLL Clock</p> <p>2 - Interface-PLL Clock</p> <p>3 - Reserved</p> <p>4,5,6,7 : Output clock is disabled</p> <p>After programming, wait for M4SS_SDMEMCLK_SWITCHED to be 1'b1</p> |
| 5:0 | R/W | MCUHP_SDMEM_CLK_8_DIV_FAC | | Specifies the clock division factor for SDMEM Controller. |

162 . MCUHP_SDMEM_CLK_CONFIG Description

MCUHP_CLK_STATUS_REG

| Bit | Access | Function | Default Value | Description |
|-------|--------|--------------------------|---------------|---|
| 31 | R | | 1 | Status of Dynamic Clock Mux in Reference Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 30:24 | - | Reserved | - | |
| 23 | R | MCUHP_CCI_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in CCI Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 22 | R | MCUHP_EXT_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in External Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 21 | | MCUHP_LP_CLK_SWITCHED | | Status of Dynamic Clock Mux in Low-Power Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 20:19 | - | Reserved | - | |
| 18 | R | MCUHP_ETH_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in Ethernet RMII Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 17 | R | MCUHP_I2S_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in I ² S Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 16 | - | Reserved | - | |
| 15 | R | MCUHP_CT_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in Configurable Timer Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 14 | R | MCUHP_SDMEM_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in SD-MEM(eMMC) Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 13 | R | MCUHP_SSI_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in SPI/SSI Master Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |

| Bit | Access | Function | Default Value | Description |
|-----|--------|--------------------------|---------------|---|
| 12 | R | MCUHP_GSPI_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in Generic SPI Master Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 11 | R | MCUHP_UART2_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in UART2 Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 10 | R | MCUHP_UART1_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in UART1 Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 9 | R | MCUHP_QSPI_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in SPI Flash Controller Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 8 | R | MCUHP_PROC_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in Processor Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 7:0 | - | Reserved | - | |

163 . MCUHP_CLK_STATUS_REG Description

MCUHP_MISC_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------------------|---------------|---|
| 31:17 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |
| 16 | R/W | MCUHP_CCI_CLK_SYNC_0EN | | Specifies the clock dependency w.r.t processor clock used for MCUHP CCI 0 - Independent clock source 1 - Undivided version of Processor Clock |
| 15:0 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |

164 . MCUHP_MISC_CONFIG_1 Description

MCUHP_MISC_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|---------|--------|-----------------------------|---------------|---|
| 31:24 - | | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |
| 23 | R/W | MCUHP_I2S_MASTER_SLAVE_MODE | | Writing 1 to this configures I ² S controller to Master Mode. Writing 0 to this configures I ² S controller to Slave Mode. |
| 22:0 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |

165 .MCUHP_MISC_CONFIG_3 Description

4.12 MCU ULP Clock Architecture

4.12.1 General Description

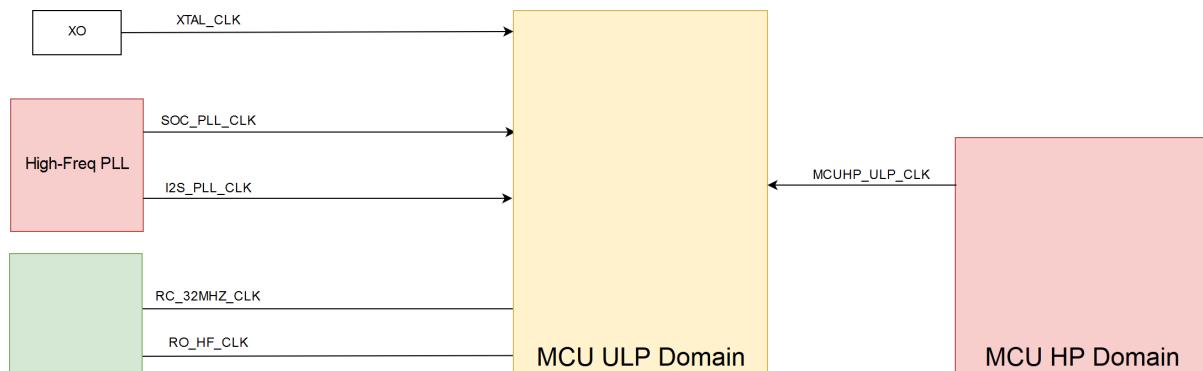
The MCU ULP (Ultra Low Power) domain contains the MCU ULP AHB Inter-Connect-Matrix, MCU ULP Peripherals and the direct AHB Interface with the Processor. This section describes the different clock sources possible for each peripheral and the AHB/APB Interfaces.

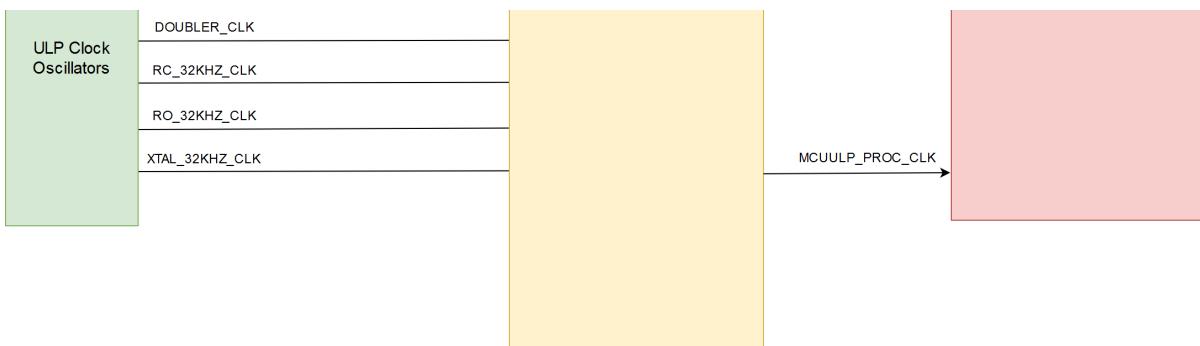
4.12.2 Features

- The clock sources used for MCU ULP domain includes RC, RO, XTAL clocks in addition to the PLL generated high frequency clocks.
- The SoC-PLL can be used as a clock source for MCU ULP Processing Interface.
- The I2S-PLL can be used for ULP-I²S peripheral.
- The clocks from High-Frequency PLL's are valid only in PS4/PS3 state which are described in Power Architecture section.
- The XTAL Clock is valid only in PS4/PS3 state which are described in Power Architecture section.
- The frequency and clock source for Peripherals can be configured independently of the MCU ULP AHB clock.
- A clock source from MCU-HP Domain which is a divided version of Processor clock can be used for MCU ULP AHB Clock and Peripherals.
- The UULP Vbat Peripherals are configured by the processor through the MCU ULP APB Interface.

4.12.3 Functional Description

The following figure depicts the clock sources used for the functionality present in MCU ULP domain.





MCU-ULP Clocking Scheme Overview

The clock to the following blocks can be configured independently.

1. MCU-ULP AHB Clock
2. I²S in Master Mode
3. SPI/SSI Master
4. UART
5. Timer
6. Touch Sensor
7. Aux-ADC/DAC Controller
8. Voice-Activity Detection
9. UULP APB Clock

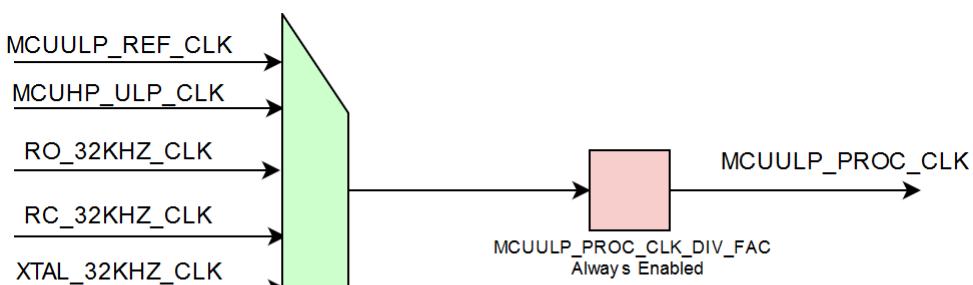
The following blocks operate using the MCU ULP AHB clock.

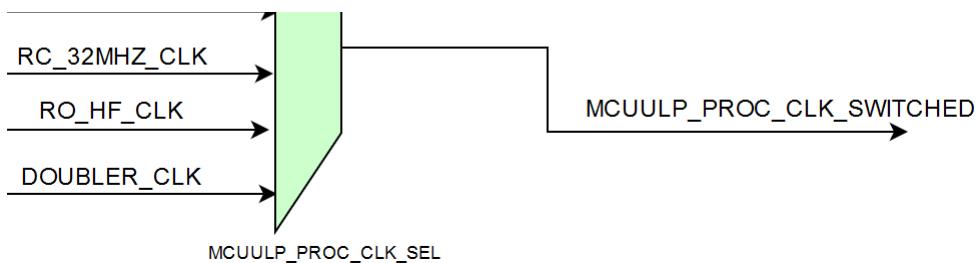
1. Micro-DMA
2. FIM
3. I²C
4. IR Receiver

The following sections describe the clock architecture for each of the functionality mentioned above. The reference clock generated for MCU-ULP domain (mcuulp_ref_clock) is used in generation of the clocks for different peripherals/modules.

4.12.4 AHB Interface Clock

The MCU ULP AHB ICM, MCU ULP AHB Bridge and the MCU ULP APB Interfaces operate using this clock. The clock from MCU-HP Domain (mcuhp_ulp_clk) is used in generation of this clock. The clock source and frequency for this clock can be configured through MCUULP_PROC_CLK_SEL and MCUULP_PROC_CLK_DIV_DAC in MCUULP_PROC_CLK_CONFIG Register. The Clock switching status can be read through MCUULP_PROC_CLK_SWITCHED in MCUULP_CLK_STATUS_REG Register.



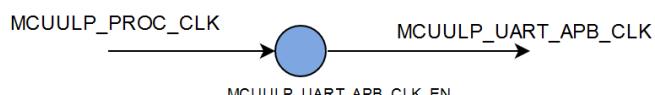
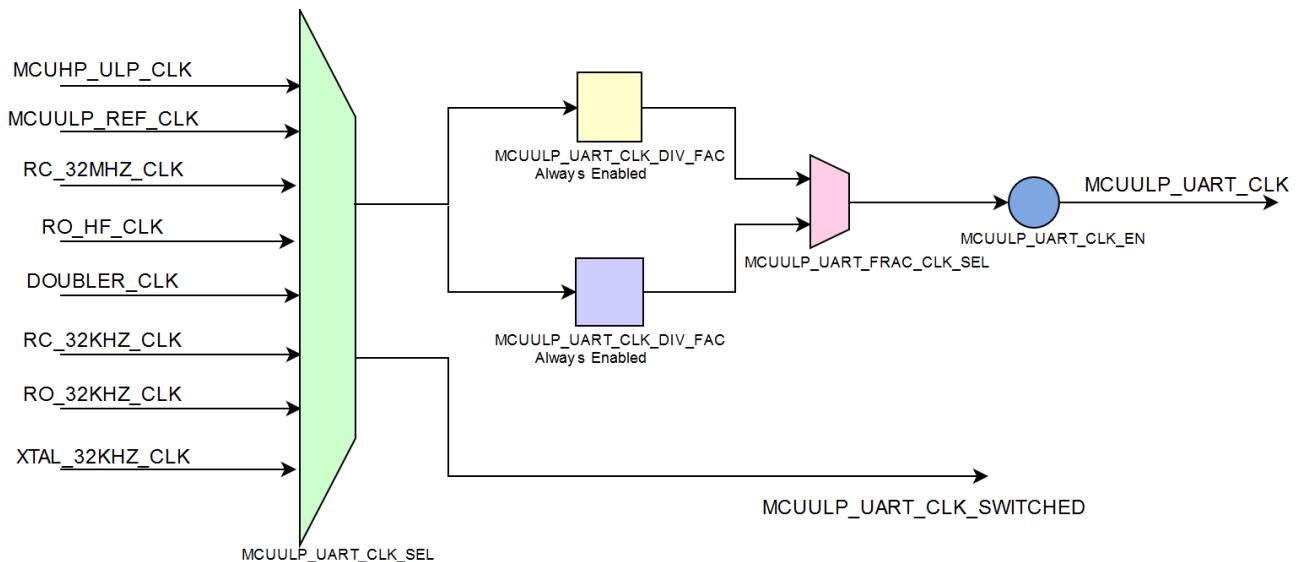


MCU-ULP AHB Interface Clock Generation

4.12.5 UART

There are multiple modes of generating the clock for UART Controller. The Clock switching status can be read through MCUHP_UART_CLK_SWITCHED in MCUULP_CLK_STATUS_REG Register.

- **Clock Generation**
 - Clock source can be configured through MCUULP_UART_CLK_SEL in MCUULP_UART_CLK_CONFIG Register.
 - Clock Division factor can be configured through MCUULP_UART_CLK_DIV_FAC in MCUULP_UART_CLK_CONFIG Register.
 - Divided clock from a Clock swallow or Fractional Divider can be selected through MCUULP_UART_FRAC_CLK_SEL in MCUULP_UART_CLK_CONFIG Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and APB clock can be controlled independently.
 - Configure MCUULP_UART_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the Controller clock.
 - Configure MCUULP_UART_APB_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the APB clock.



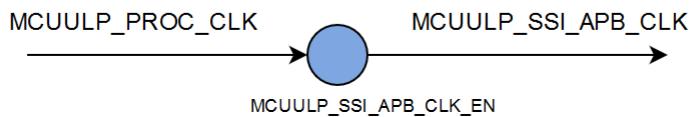
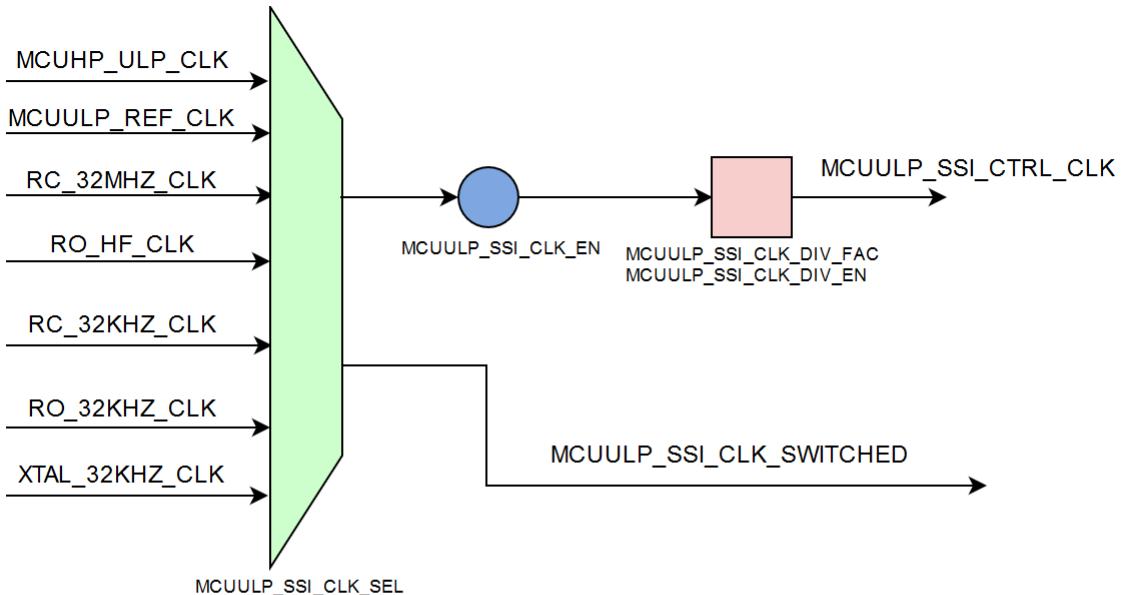
MCUULP_UART1_APB_CLK_EN

MCU-ULP UART Clock Generation

4.12.6 SPI / SSI Master

There are multiple clock sources for SPI/SSI Master Controller. The Clock switching status can be read through MCUULP_SSI_CLK_SWITCHED in MCUULP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUULP_SSI_CLK_SEL in MCUULP_I2C_SSI_CLK_CONFIG Register.
 - Clock Division factor can be configured through MCUULP_SSI_CLK_DIV_FAC in MCUULP_I2C_SSI_CLK_CONFIG Register.
 - Clock Divider can be disabled by configuring MCUULP_SSI_CLK_DIV_EN in MCUULP_I2C_SSI_CLK_CONFIG Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and APB clock can be controlled independently.
 - Configure MCUULP_SSI_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the Controller clock.
 - Configure MCUULP_SSI_APB_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the APB clock.

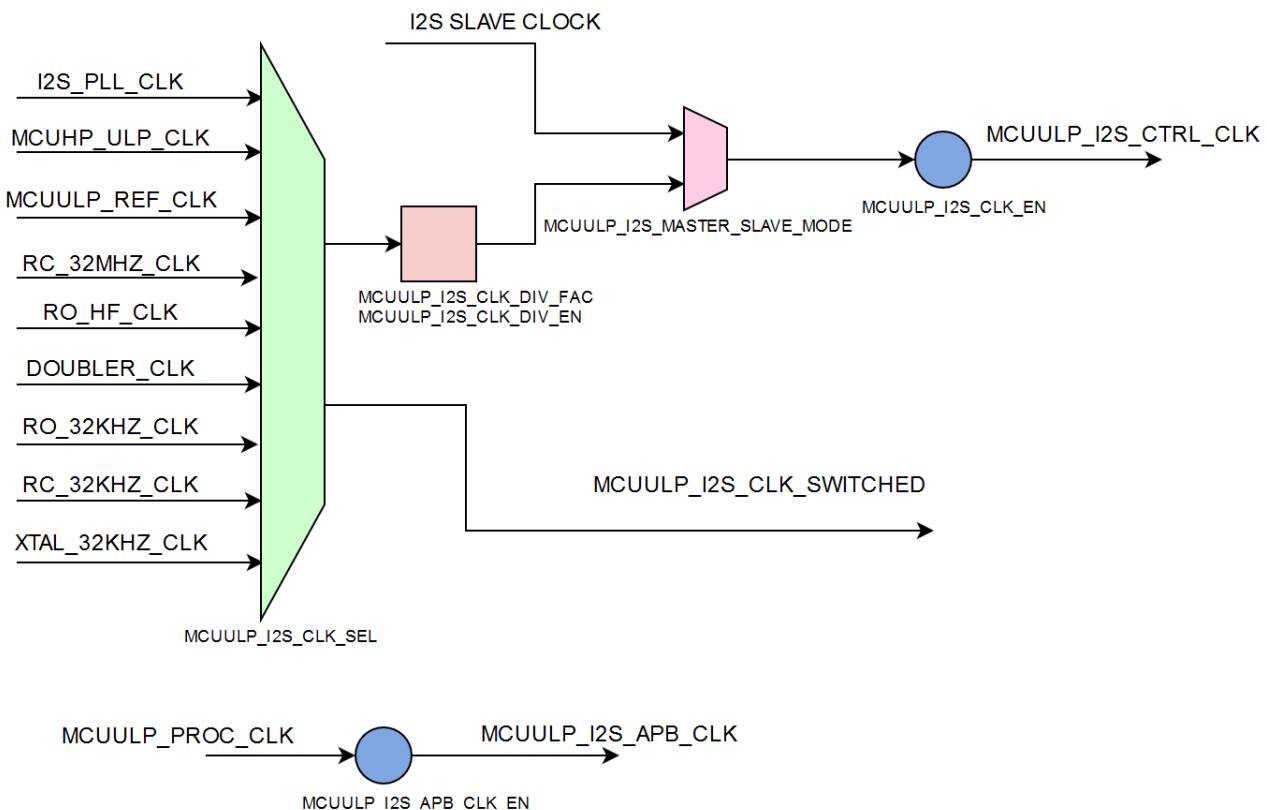


MCU-ULP SSI Clock Generation

4.12.7 I²S Controller

There are multiple clock sources for I²S Controller which is used in Master Mode. The Clock switching status can be read through MCUULP_I2S_CLK_SWITCHED in MCUULP_CLK_STATUS_REG Register.

- Clock Generation
 - I²S Slave clock is derived from the external Master Device through GPIO PAD's.
 - Clock source can be configured through MCUULP_I2S_CLK_SEL in MCUULP_I2S_CLK_CONFIG Register.
 - Clock Division factor can be configured through MCUULP_I2S_CLK_DIV_FAC in MCUULP_I2S_CLK_CONFIG Register.
 - Clock Divider can be disabled by configuring MCUULP_I2S_CLK_DIV_EN in MCUULP_I2S_CLK_CONFIG Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption or before configuring the clock source and frequency. The controller clock and AHB clock can be controlled independently.
 - Configure MCUULP_I2S_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the Controller clock.
 - Configure MCUULP_I2S_APB_CLK_EN in MCUULP_I2S_CLK_CONFIG Register for enabling/disabling the APB Interface clock.
- The Master/Slave mode is configured through MCUULP_I2S_MASTER_SLAVE_MODE in MCUULP_I2S_CLK_CONFIG Register.

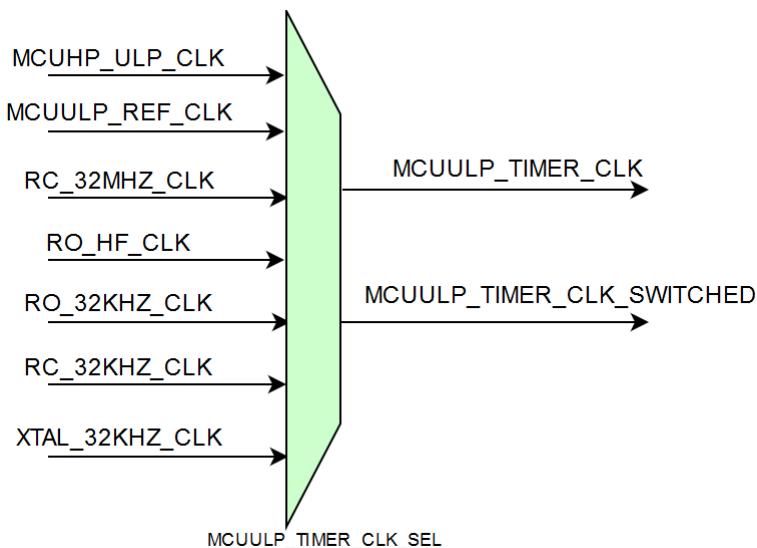


I2S Controller Clock Generation

4.12.8 Timer

There are multiple clock sources for Timers. The Clock switching status can be read through MCUULP_TIMER_CLK_SWITCHED in MCUULP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUULP_TIMER_CLK_SEL in MCUULP_TIMER_CLK_CONFIG Register.

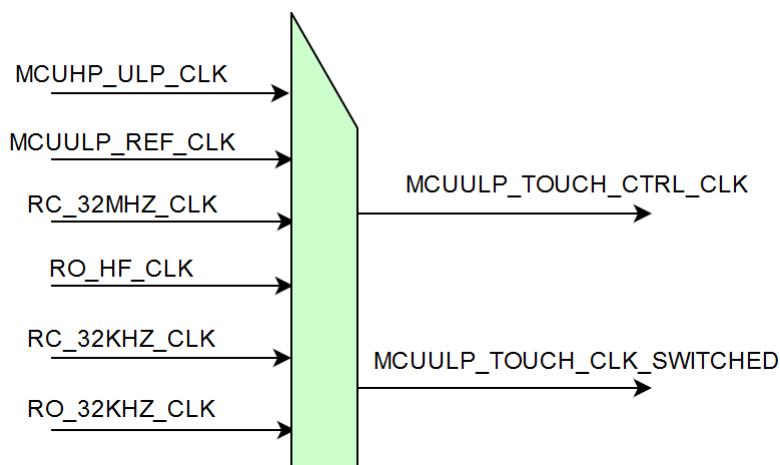


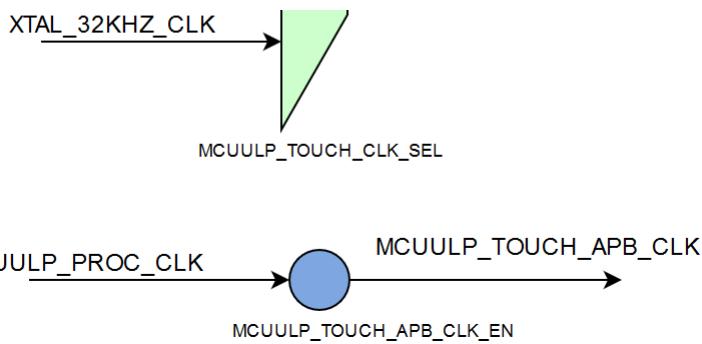
MCU-ULP Timer Clock Generation

4.12.9 Touch Sensor

There are multiple clock sources for Touch Sensor Controller. The Clock switching status can be read through MCUULP_TOUCH_CLK_SWITCHED in MCUULP_CLK_STATUS_REG Register.

- Clock Generation
 - Clock source can be configured through MCUULP_TOUCH_CLK_SEL in MCUULP_TOUCH_CLK_CONFIG Register.
- The clocks to the controller APB Interface can be disabled when not in use for efficient power consumption.
 - Configure MCUULP_TOUCH_APB_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the APB Interface clock.



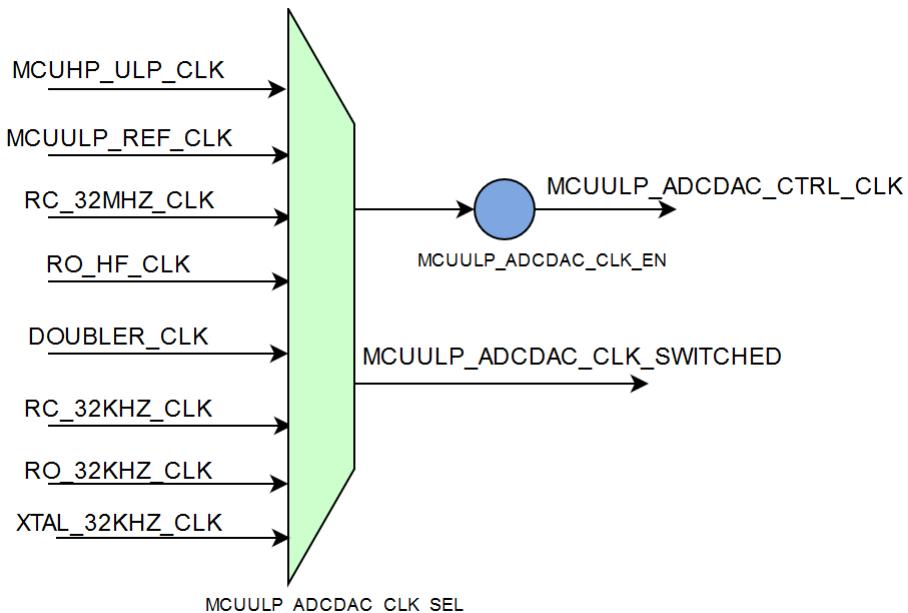


MCU-ULP Touch Sensor Clock Generation

4.12.10 Aux-ADC/DAC Controller

There are multiple clock sources for Aux-ADC/DAC Controller. The Clock switching status can be read through `MCUULP_ADCDAC_CLK_SWITCHED` in `MCUULP_CLK_STATUS_REG` Register.

- Clock Generation
 - Clock source can be configured through `MCUULP_ADCDAC_CLK_SEL` in `MCUULP_ADCDAC_CLK_CONFIG` Register.
- The clocks to the controller can be disabled when not in use for efficient power consumption.
 - Configure `MCUULP_ADCDAC_CLK_EN` in `MCUULP_CLK_EN_REG2` Register for enabling/disabling the Controller clock.



MCU-ULP Aux-ADC/DAC Clock Generation

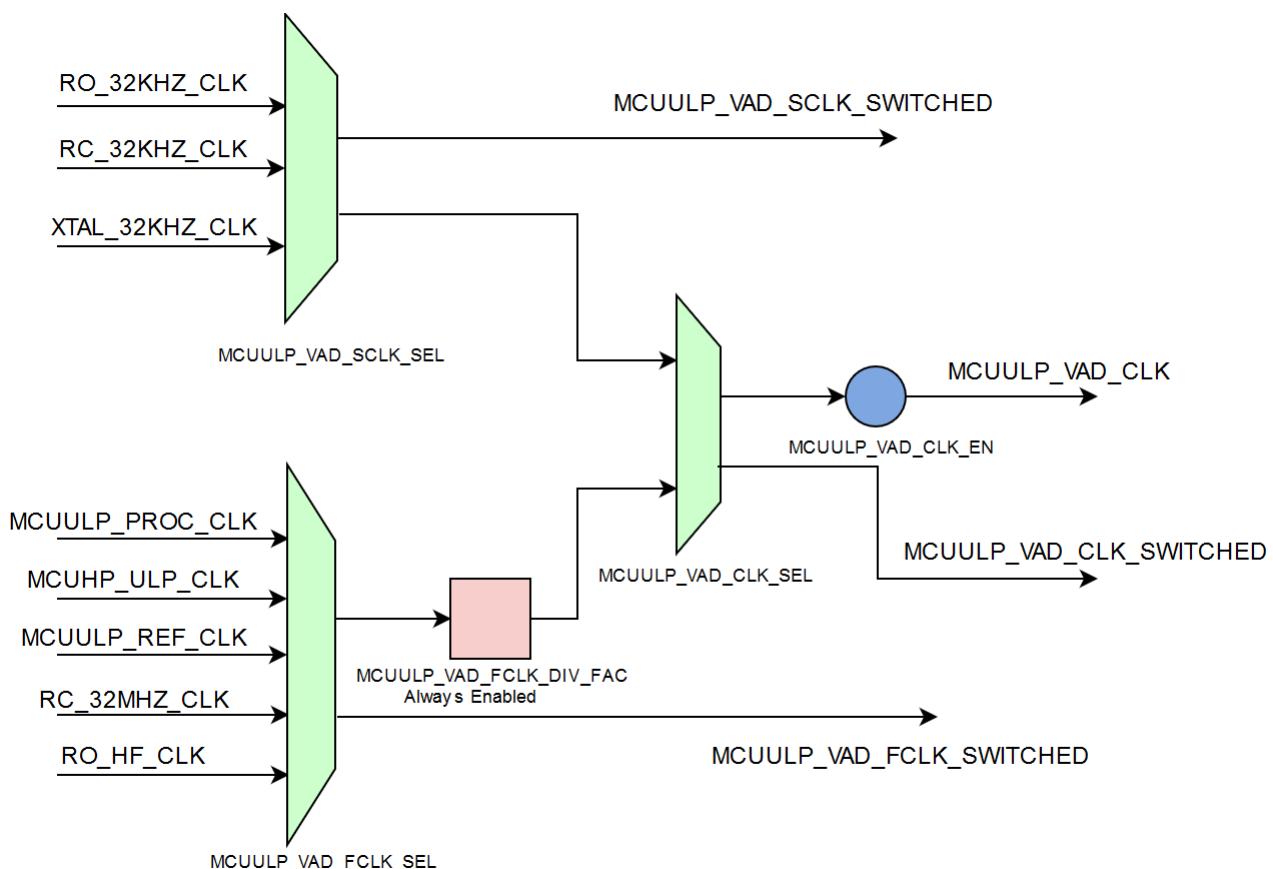
4.12.11 Voice-Activity Detection (VAD)

There are two clocks generated for VAD. One is a slow clock and other is a fast clock. The VAD Controller can be switched to slow clock when not in use for efficient power management.

There are multiple clock sources for both the clocks. The Clock switching status for slow, fast and the VAD Controller clock can be read through `MCUULP_VAD_SCLK_SWITCHED`, `MCUULP_VAD_FCLK_SWITCHED` and `MCUULP_VAD_CLK_SWITCHED` in `MCUULP_CLK_STATUS_REG` Register respectively.

- Slow Clock Generation

- Clock source can be configured through MCUULP_VAD_SCLK_SEL in MCUULP_VAD_CLK_CONFIG Register.
- Fast Clock Generation
 - Clock source can be configured through MCUULP_VAD_FCLK_SEL in MCUULP_VAD_CLK_CONFIG Register.
 - Clock Division factor can be configured through MCUULP_VAD_FCLK_DIV_FAC in MCUULP_VAD_CLK_CONFIG Register.
- Controller Clock generation
 - Fast/Slow clocks to the VAD controller can be configured through MCUULP_VAD_CLK_SEL in MCUULP_VAD_CLK_CONFIG Register
- The clocks to the controller can be disabled when not in use for efficient power consumption.
 - Configure MCUULP_VAD_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the Controller clock.

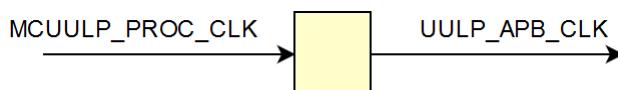


MCU-ULP VAD Clock Generation

4.12.12 UULP APB Clock

The APB Clock needs to be configured such that the max frequency is 20MHz. This clock is derived from the MCU ULP AHB clock.

- The Clock Division factor can be configured through MCUULP_APB_CLK_DIV_FAC in MCUULP_UULP_CLK_CONFIG Register.



MCUULP_APB_CLK_DIV_FAC
Always Enabled

MCU-ULP UULP APB Clock Generation

4.12.13 Static Clock Gated Domains

The clock to the domains which operate on the AHB Interface clock can be disabled when not in use for efficient power management. Below mentioned are the programming details for the same.

- **Micro-DMA:** Configure MCUULP_UDMA_CLK_EN in MCUULP_CLK_EN_REG2 Register for enabling/disabling the clock to Micro-DMA module.
- **FIM:** Configure MCUULP_FIM_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the clock to FIM module.
- **I²C:**
 - Configure MCUULP_I2C_APB_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the clock to I²C APB Interface.
 - Configure MCUULP_I2C_CLK_EN in MCUULP_I2C_SSI_CLK_CONFIG Register for enabling/disabling the clock to I²C Controller.
- **IR Receiver:** Configure MCUULP_IR_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the clock to IR-Receiver module.
- **Enhanced-GPIO:** Configure MCUULP_EGPIO_CLK_EN in MCUULP_CLK_EN_REG1 Register for enabling/disabling the clock to Enhanced-GPIO module.

4.12.14 Register Summary

Base_address = 0x2404_1400

| Register Name | Offset | Description |
|---------------------------|--------|---|
| MCUULP_CLK_EN_REG1 | 0x00 | MCU-ULP Clock Enable Register 1 |
| MCUULP_PROC_CLK_CONFIG | 0x14 | MCU-ULP AHB Clock Configuration Register |
| MCUULP_I2C_SSI_CLK_CONFIG | 0x18 | MCU-ULP SSI Master and I2C Clock Configuration Register |
| MCUULP_I2S_CLK_CONFIG | 0x1C | MCU-ULP I ² S Clock Configuration Register |
| MCUULP_UART_CLK_CONFIG | 0x20 | MCU-ULP UART Clock Configuration Register |
| MCUULP_CLK_STATUS_REG | 0x28 | MCU-ULP Clock Status Register |
| MCUULP_TOUCH_CLK_CONFIG | 0x2C | MCU-ULP Touch Sensor Clock Configuration Register |
| MCUULP_TIMER_CLK_CONFIG | 0x30 | MCU-ULP Timer Clock Configuration Register |
| MCUULP_ADCDAC_CLK_CONFIG | 0x34 | MCU-ULP Aux-ADC/DAC Configuration Register |
| MCUULP_VAD_CLK_CONFIG | 0x38 | MCU-ULP VAD Configuration Register |
| MCUULP_CLK_EN_REG2 | 0xA0 | MCU-ULP Clock Enable Register 2 |
| MCUULP_UULP_CLK_CONFIG | 0xA4 | UULP APB Clock Configuration Register |

166 . List of Registers

4.12.15 Register Description

MCUULP_CLK_EN_REG1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|---|
| 31 | R/W | MCUULP_TOUCH_APB_CLK_EN | 0 | Writing 1 to this enables clock to Touch Sensor APB Interface. Writing 0 to this disables clock to Touch Sensor APB Interface. |
| 30:22 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |
| 21 | R/W | MCUULP_TIMER_PCLK_EN | 0 | This bit is used to enable static clock to Timer APB Interface. |
| 20:18 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |
| 17 | R/W | MCUULP_VAD_CLK_EN | 0 | Writing 1 to this enables clock to VAD Controller. Writing 0 to this disables clock to VAD Controller. |
| 16 | R/W | MCUULP_FIM_CLK_EN | 0 | Writing 1 to this enables clock to FIM Engine. Writing 0 to this disables clock to FIM Engine. |
| 15 | - | Reserved | - | It is recommended to write these bits to 0. |
| 14 | R/W | MCUULP_EGPIO_CLK_EN | 0 | Writing 1 to this enables clock to Enhanced-GPIO. Writing 0 to this disables clock to Enhanced-GPIO. |
| 13 | R/W | MCUULP_TIMER_CLK_EN | 0 | Writing 1 to this enables clock to Timers. Writing 0 to this disables clock to Timers. |
| 12:11 | - | Reserved | - | It is recommended to write these bits to 0. |
| 10 | R/W | MCUULP_UART_CLK_EN | 0 | Writing 1 to this enables clock to UART Controller. Writing 0 to this disables clock to UART Controller. |
| 9 | R/W | MCUULP_UART_APB_CLK_EN | 0 | Writing 1 to this enables clock to UART APB Interface. Writing 0 to this disables clock to UART APB Interface. |
| 8 | R/W | MCUULP_SSI_CLK_EN | 0 | Writing 1 to this enables clock to SPI/SSI Master. Writing 0 to this disables clock to SPI/SSI Master. |
| 7 | R/W | MCUULP_SSI_APB_CLK_EN | 0 | Writing 1 to this enables clock to SPI/SSI Master APB Interface. Writing 0 to this disables clock to SPI/SSI Master APB Interface. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------------|-------------|---|
| 6 | R/W | MCUULP_I2S_CLK_EN | 0 | Writing 1 to this enables clock to I ² S Controller. |
| | | | | Writing 0 to this disables clock to I ² S Controller. |
| 5 | R/W | MCUULP_I2C_APB_CLK_EN | 0 | Writing 1 to this enables clock to I ² C APB Interface. |
| | | | | Writing 0 to this disables clock to I ² C APB Interface. |
| 4 | R/W | MCUULP_IR_CLK_EN | 0 | Writing 1 to this enables clock to IR Receiver. |
| | | | | Writing 0 to this enables clock to IR Receiver. |
| 3:0 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |

167 . MCUULP_CLK_EN_REG1 Description

MCUULP_PROC_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|--|
| 31:13 | - | Reserved | 0 | It is recommended to write these bits to 0. |
| 12:5 | R/W | MCUULP_PROC_CLK_DIV_DAC | 0 | Specifies the clock division factor for AHB Interface Clock. |
| 4:1 | R/W | MCUULP_PROC_CLK_SEL | 0 | Specifies the clock source to be used for AHB Interface 0 - MCU-ULP Reference Clock 1 - ro_32khz_clk 2 - rc_32khz_clk 3 - xtal_32khz_clk 4 - rc_32mhz_clk 5 - ro_hf_clk 6 - MCU-HP ULP Clock 7 - doubler_clk 8-15 - Output clock is gated |
| 0 | R/W | MCUULP_BRIDGE_CLK_EN | 1 | Controls the clock used for ULP-PERIPHERAL accesses in PS4/PS3(described in Power Architecture) power states 0 - Clock is disabled 1 - Clock is enabled |

168 . MCUULP_PROC_CLK_CONFIG Description

MCUULP_I2C_SSI_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|--|
| 31:28 | R/W | MCUULP_SSI_CLK_SE_L | 15 | <p>Specifies the clock source to be used for SPI/SSI Master</p> <p>0 - MCU-ULP Reference Clock</p> <p>1 - ro_32khz_clk</p> <p>2 - rc_32khz_clk</p> <p>3 - xtal_32khz_clk</p> <p>4 - rc_32mhz_clk</p> <p>5 - ro_hf_clk</p> <p>6 - MCU-HP ULP Clock</p> <p>7-14 - Reserved</p> <p>15 - Output clock is gated</p> |
| 27:24 | - | Reserved | - | It is recommended to write these bits to 0. |
| 23:17 | R/W | MCUULP_SSI_CLK_DIV0_FAC | | Specifies the clock division factor for SPI/SSI Master Clock. |
| 16 | R/W | MCUULP_SSI_CLK_DIV_EN | 0 | <p>Writing 1 to this enables clock to SPI/SSI Master Clock Dividers.</p> <p>Writing 0 to this disables clock to SPI/SSI Master Clock Dividers.</p> |
| 15:1 | - | Reserved | - | It is recommended to write these bits to 0. |
| 0 | R/W | MCUULP_I2C_CLK_EN | 0 | <p>Writing 1 to this enables clock to I²C Controller.</p> <p>Writing 0 to this disables clock to I²C Controller.</p> |

169 .MCUULP_I2C_SSI_CLK_CONFIG Description

MCUULP_I2S_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------|-------------|--|
| 31:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | R/W | MCUULP_I2S_APB_CLK_EN | 0 | <p>Writing 1 to this enables clock to I²S APB Interface.</p> <p>Writing 0 to this disables clock to I²S APB Interface.</p> |
| 17:14 | - | Reserved | - | It is recommended to retain the contents by using read/modify write to this register. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------------------|-------------|--|
| 13 | R/W | MCUULP_I2S_MASTER_S LAVE_MODE | 0 | Writing 1 to this configures I ² S to Master Mode. Writing 0 to this configures I ² S to Slave Mode. |
| 12:5 | R/W | MCUULP_I2S_CLK_DIV_F AC | 0 | Specifies the clock division factor for I ² S Master Clock. |
| 4:1 | R/W | MCUULP_I2S_CLK_SEL | 15 | Specifies the clock source to be used for I ² S Master 0 - MCU-ULP Reference Clock 1 - ro_32khz_clk 2 - rc_32khz_clk 3 - xtal_32khz_clk 4 - rc_32mhz_clk 5 - ro_hf_clk 6 - MCU-HP ULP Clock 7 - doubler_clk 8 - i2s_pll_clk 9-14 - Reserved 15 - Output clock is gated |
| 0 | R/W | MCUULP_I2S_CLK_DIV_E N | 0 | Writing 1 to this enables clock to I ² S Clock Dividers. Writing 0 to this disables clock to I ² S Clock Dividers. |

170 .MCUULP_I2S_CLK_CONFIG Description

MCUULP_UART_CLK_CONFIG

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------------------------|---------------|---|
| 31:13 | - | Reserved | - | It is recommended to write these bits to 0. |
| 12:5 | R/W | MCUULP_UART_CLK_DI0 V_FAC | 0 | Specifies the clock division factor for UART. |

| Bit | Access | Function | Default Value | Description |
|-----|--------|--------------------------|---------------|---|
| 4:1 | R/W | MCUULP_UART_CLK_SEL | 15 | <p>Specifies the clock source to be used for UART</p> <p>0 - MCU-ULP Reference Clock</p> <p>1 - ro_32khz_clk</p> <p>2 - rc_32khz_clk</p> <p>3 - xtal_32khz_clk</p> <p>4 - rc_32mhz_clk</p> <p>5 - ro_hf_clk</p> <p>6 - MCU-HP ULP Clock</p> <p>7 - doubler_clk</p> <p>8-14 - Reserved</p> <p>15 - Output clock is gated</p> |
| 0 | R/W | MCUULP_UART_FRAC_CLK_SEL | 0 | <p>Selects the Divider type for UART Controller.</p> <p>0 - Clock Swallow output is selected</p> <p>1 - Fractional Clock Divider output is selected</p> |

171 . MCUULP_UART_CLK_CONFIG Description

MCUULP_CLK_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------------|-------------|--|
| 31:12 | - | Reserved | - | |
| 11 | R | MCUULP_VAD_SCLK_SWITCHED | 0 | <p>Status of Dynamic Clock Mux in VAD Slow Clock Generation</p> <p>1 : Clock got switched and output clock can be used</p> <p>0 : Clock switching is in progress</p> |
| 10 | R | MCUULP_VAD_FCLK_SWITCHED | 0 | <p>Status of Dynamic Clock Mux in VAD Fast Clock Generation</p> <p>1 : Clock got switched and output clock can be used</p> <p>0 : Clock switching is in progress</p> |
| 9 | R | MCUULP_TOUCH_CLK_SWITCHED | 0 | <p>Status of Dynamic Clock Mux in Touch Sensor Clock Generation</p> <p>1 : Clock got switched and output clock can be used</p> <p>0 : Clock switching is in progress</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------------|-------------|--|
| 8 | R | MCUULP_TIMER_CLK_SWITCHED | 0 | Status of Dynamic Clock Mux in Timer Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 7 | R | MCUULP_ADCDAC_CLK_SWITCHED | 0 | Status of Dynamic Clock Mux in Aux-ADC/DAC Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 6 | R | MCUULP_VAD_CLK_SWITCHED | 0 | Status of Dynamic Clock Mux in VAD Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 5 | R | MCUULP_SSI_CLK_SWITCHED | 0 | Status of Dynamic Clock Mux in SPI/SSI Master Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 4 | - | Reserved | - | |
| 3 | R | MCUULP_PROC_CLK_SWITCHED | 1 | Status of Dynamic Clock Mux in AHB Interface Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 2 | - | Reserved | - | |
| 1 | R | MCUULP_I2S_CLK_SWITCHED | 0 | Status of Dynamic Clock Mux in I ² S Controller Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 0 | R | MCUHP_UART_CLK_SWITCHED | 0 | Status of Dynamic Clock Mux in UART Clock Generation 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |

172 . MCUULP_CLK_STATUS_REG Register

MCUULP_TOUCH_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------|-------------|---|
| 12:5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4:1 | R/W | MCUULP_TOUCH_CLK_SEL | 15 | Specifies the clock source to be used for Touch Sensor 0 - MCU-ULP Reference Clock 1 - ro_32khz_clk 2 - rc_32khz_clk 3 - xtal_32khz_clk 4 - rc_32mhz_clk 5 - ro_hf_clk 6 - MCU-HP ULP Clock 7-14 - Reserved 15 - Output clock is gated |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

173 .MCUULP_TOUCH_CLK_CONFIG Description

MCUULP_TIMER_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------|-------------|--|
| 13:5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4:1 | R/W | MCUULP_TIMER_CLK_SEL | 15 | Specifies the clock source to be used for Timer 0 - MCU-ULP Reference Clock 1 - ro_32khz_clk 2 - rc_32khz_clk 3 - xtal_32khz_clk 4 - rc_32mhz_clk 5 - ro_hf_clk 6 - MCU-HP ULP Clock 7-14 - Reserved 15 - Output clock is gated |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

174 .MCUULP_TIMER_CLK_CONFIG Description

MCUULP_ADCDAC_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------|-------------|---|
| 12:5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4:1 | R/W | MCUULP_ADCDAC_CLK_SEL | 15 | Specifies the clock source to be used for Aux-ADC/DAC Controller 0 - MCU-ULP Reference Clock 1 - ro_32khz_clk 2 - rc_32khz_clk 3 - xtal_32khz_clk 4 - rc_32mhz_clk 5 - ro_hf_clk 6 - MCU-HP ULP Clock 7 - doubler_clk 8 - i2s_pll_clk 8-14 - Reserved 15 - Output clock is gated |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

175 .MCUULP_ADCDAC_CLK_CONFIG Description

MCUULP_VAD_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|--|
| 31:17 | - | Reserved | - | It is recommended to write these bits to 0. |
| 16:9 | R/W | MCUULP_VAD_FCLK_DIV_FAC | 0 | Specifies the clock division factor for VAD Fast clock. |
| 8:5 | R/W | MCUULP_VAD_FCLK_SEL | 15 | Specifies the clock source to be used for VAD Fast Clock 0 - MCU ULP AHB Interface clock 1 - MCU-ULP Reference clock 2 - rc_32mhz_clk 3 - ro_hf_clk 4 - MCU-HP ULP clock 5-14 - Reserved 15 - Output clock is gated |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------|-------------|---|
| 4 | R/W | MCUULP_VAD_CLK_SEL | 0 | Specifies the clock source to be used for VAD Controller 0 - VAD Slow Clock 1 - VAD Fast Clock |
| 3:1 | R/W | MCUULP_VAD_SCLK_SEL | 7 | Specifies the clock source to be used for VAD Slow Clock 0 - ro_32khz_clk 1 - rc_32khz_clk 2 - xtal_32khz_clk 4-6 - Reserved 8 - Output clock is gated |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

176 . MCUULP_VAD_CLK_CONFIG Description

MCUULP_CLK_EN_REG2

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------------------|---------------|---|
| 31:18 | - | Reserved | - | It is recommended to write these bits to 0. |
| 17 | R/W | MCUULP_UDMA_CLK_0_EN | 0 | Writing 1 to this enables clock to Micro-DMA. Writing 0 to this disables clock to Micro-DMA. |
| 16:13 | - | Reserved | - | It is recommended to write these bits to 0. |
| 12 | R/W | MCUULP_ADCDAC_C_0_LK_EN | 0 | Writing 1 to this enables clock to Aux-ADC/DAC Controller. Writing 0 to this disables clock to Aux-ADC/DAC Controller. |
| 11:0 | - | Reserved | - | It is recommended to write these bits to 0. |

177 . MCUULP_CLK_EN_REG2 Description

MCUULP_UULP_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------------------|-------------|--|
| 31:9 | - | Reserved | - | It is recommended to write these bits to 0. |
| 8 | - | Reserved | - | It is recommended to write these bits to 1. |
| 7:0 | R/W | MCUULP_APB_CLK_DI_2_V_FAC | 2 | Specifies the clock division factor for UULP APB Interface |

178 . MCUULP_UULP_CLK_CONFIG Description

4.13 MCU ULP VBAT Clock Architecture

4.13.1 General Description

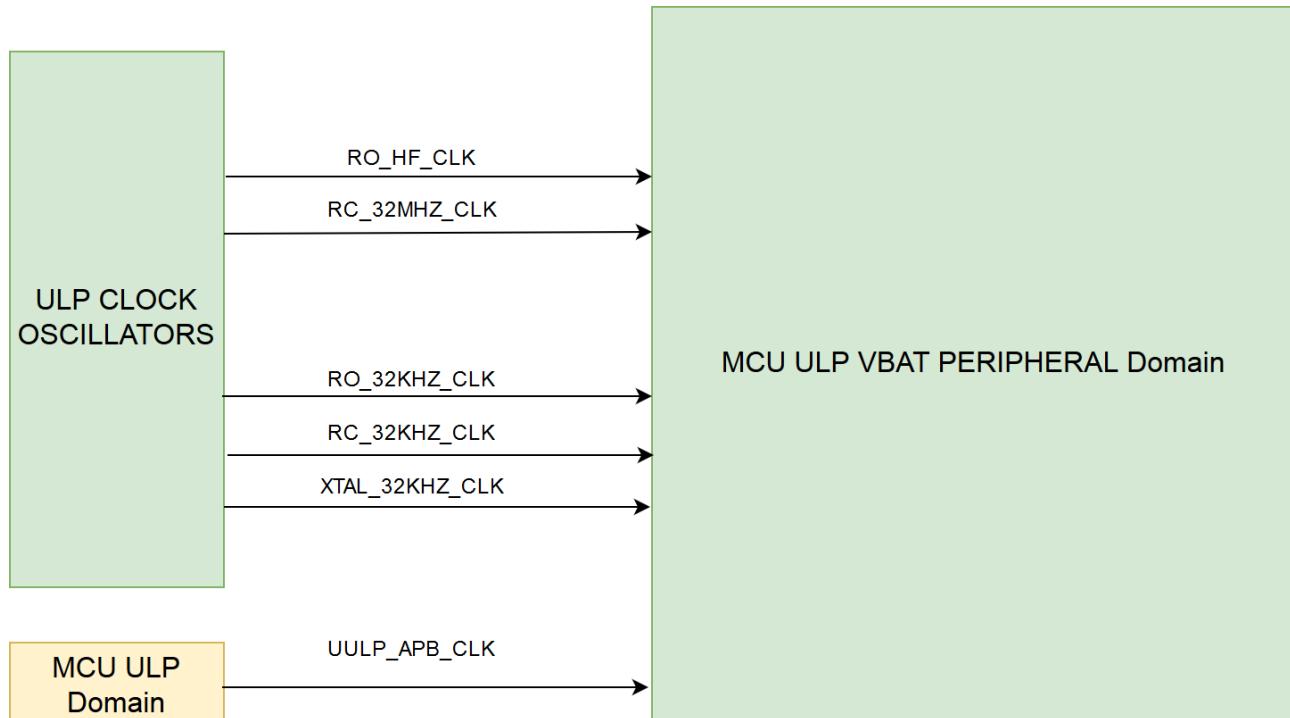
The MCU ULP VBAT domain contains the MCU ULP VBAT Peripherals. This section describes the different clock sources possible for each peripheral.

4.13.2 Features

- The clock sources used for MCU ULP VBAT domain includes RC, RO, XTAL clocks from the ULP Clock Oscillators.
- The Peripherals are configured by the processor through the APB Interface for which the clock is fed from MCU ULP Domain.

4.13.3 Functional Description

The following figure depicts the clock sources used for the functionality present in MCU ULP VBAT domain.



Clock Generation

The list below contains the different peripherals for which clock can be configured independently.

1. HIGH-FREQ CLOCK
 - a. This clock is used for Low-Power State machines.
 - b. This can be configured using `MCUULP_VBAT_HF_CLK_SEL` in **MCUULP_VBAT_HFCLK_REG** Register.
 - c. The clock switching status can be read through `MCUULP_VBAT_HF_CLK_SWITCHED` in **MCUULP_VBAT_HFCLK_REG** Register
2. LOW-FREQ CLOCK
 - a. This clock is used for the following for the following UULP-PERIPHERALS
 - i. WatchDog Timer
 - ii. Deep-Sleep Timer
 - iii. GPIO Timestamping
 - b. This can be configured using `MCUULP_VBAT_LF_CLK_SEL` in **MCUULP_VBAT_LFCLK_REG** Register.

- c. The clock switching status can be read through MCUULP_VBAT_LF_CLK_SWITCHED in **MCUULP_VBAT_LFCLK_REG** Register

4.13.4 Register Summary

Base_address = 0x2404_8000

| Register Name | Offset | Description |
|-----------------------|--------|--------------------------------------|
| MCUULP_VBAT_LFCLK_REG | 0x020 | Low Frequency Clock Select Register |
| MCUULP_VBAT_HFCLK_REG | 0x118 | High Frequency Clock Select Register |

179 . Register Summary

4.13.5 Register Description

MCUULP_VBAT_LFCLK_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------|-----------|---|
| 31:4 | R | Reserved | NA | Reserved |
| 3 | R | MCUULP_VBAT_LF_CLK_SWITCHED | 1'b1 | Status of NPSS Low Frequency Clock Dynamic Clock Mux 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 2:0 | R/W | MCUULP_VBAT_LF_CLK_SEL | 2'b00 | 001 : ro_32k_clk 010 : rc_32k_clk 100 : xtal_32k_clk After programming, wait for NPSS_LFCLK_SWITCHED to be 1'b1 |

180 . MCUULP_VBAT_LFCLK_REG Description

MCUULP_VBAT_HFCLK_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------------|-------------|--|
| 30:16 | R | Reserved | NA | Reserved |
| 15 | R | MCUULP_VBAT_HF_CLK_SWITCHED | 1'b1 | Status of NPSS High Frequency Clock Dynamic Clock Mux 1 : Clock got switched and output clock can be used 0 : Clock switching is in progress |
| 14:5 | R | Reserved | NA | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------------|-------------|---|
| 4:2 | R/W | MCUULP_VBAT_HF_CLK_S EL | 2'd0 | 0 : Clock is Gated 1 : ro_20m_clk 2 : rc_32m_clk After programming, wait for NPSS_HFCLK_SWITCHED to be 1'b1 |
| 1:0 | R | Reserved | NA | Reserved |

181 . NPSS High Frequency Clock Select Register

4.14 ULP Clock Oscillators

4.14.1 General Description

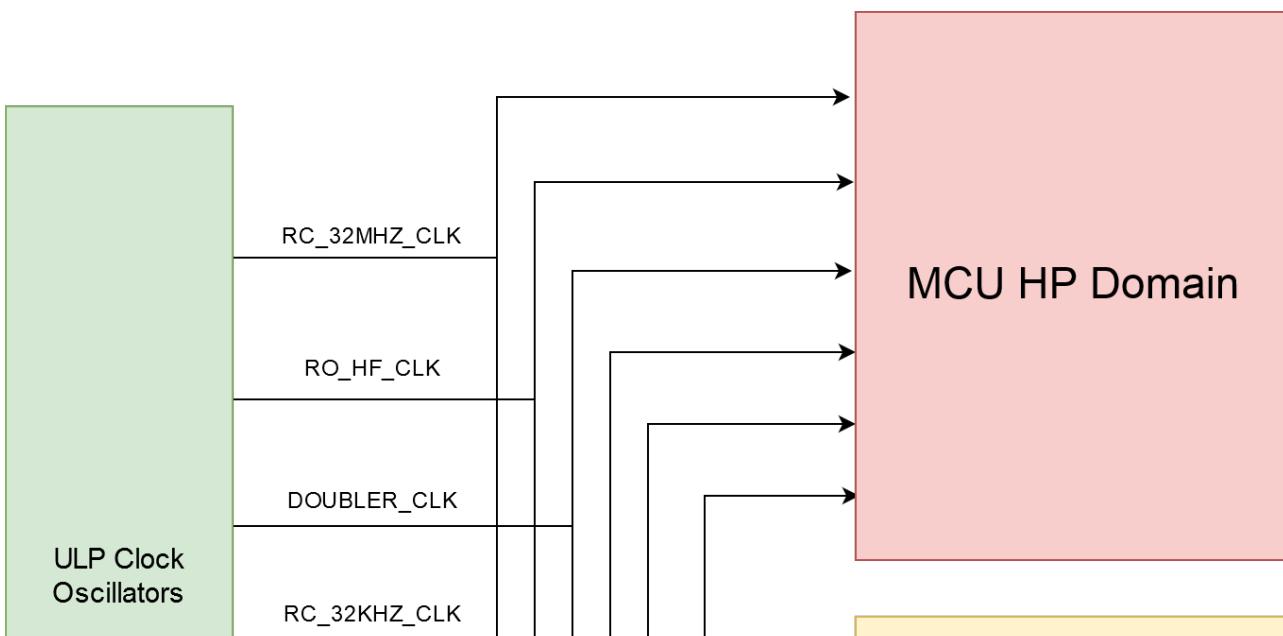
ULP Clock Oscillators is the source of the Low frequency RC/RO and High frequency RC/RO clocks which are used by MCU HP, MCU ULP and MCU ULP VBat domains as one of the clock sources for multiple peripherals.

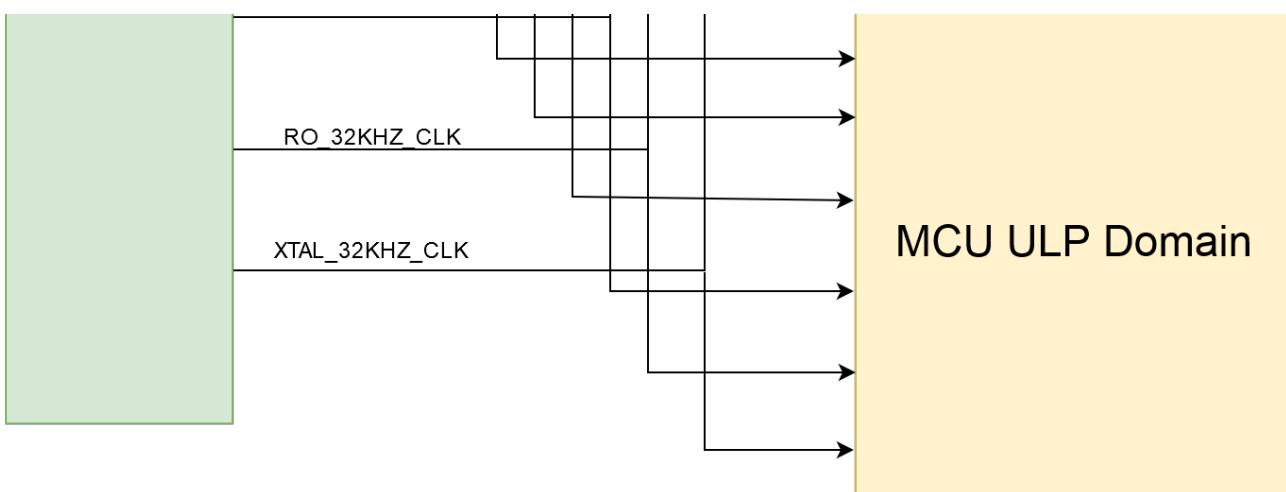
4.14.2 Features

1. Low-Power Clock Oscillators for generation of RC-Based and RO-Based clocks.
2. Fast Start-up times. The table below shows the start-up times for each clock

| S.No | Clock Source | Start-Up time |
|------|----------------|-------------------|
| 1 | RC_32MHZ_CLK | 10 Micro-seconds |
| 2 | RO_HF_CLK | 30 Micro-seconds |
| 3 | DOUBLER_CLK | 30 Micro-seconds |
| 4 | RC_32KHZ_CLK | 500 Micro-seconds |
| 5 | RO_32KHZ_CLK | 500 Micro-seconds |
| 6 | XTAL_32KHZ_CLK | 2.5 seconds. |

4.14.3 Functional Description





Clock Generation

There are different Oscillators to generate the Low-Frequency and High-Frequency clocks. The clock oscillators can be controlled individually through Register programming which are described below

4.14.4 Register Summary

Base Address: 0x2404_8100

| Register Name | Offset | Description |
|-------------------------------------|--------|--|
| ULP_CLKOSC_CTRL_REG | 0x20 | ULP Clock Oscillators Control Register 182 . Register summary |

4.14.5 Register Description

ULP_CLKOSC_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:22 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | R/W | XTAL_40MHZ_CLK_EN | 1 | Writing 1 to this enables the XTAL-40MHz Clock Writing 0 to this disables the XTAL-40MHz Clock |
| 21 | R/W | DOUBLER_CLK_EN | 0 | Writing 1 to this enables the Doubler Clock Writing 0 to this disables the Doubler Clock |
| 20 | R/W | RO_HF_CLK_EN | 0 | Writing 1 to this enables the RO High-Frequency Clock Writing 0 to this disables the RO High-Frequency Clock |
| 19 | R/W | RC_32MHZ_CLK_EN | 1 | Writing 1 to this enables the RC 32MHz Clock Writing 0 to this disables the RC 32MHz Clock |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 18 | R/W | XTAL_32KHZ_CLK_EN | 0 | Writing 1 to this enables the XTAL 32KHz Clock Writing 0 to this disables the XTAL 32KHz Clock |
| 17 | R/W | RO_32KHZ_CLK_EN | 1 | Writing 1 to this enables the RO 32KHz Clock Writing 0 to this disables the RO 32KHz Clock |
| 16 | R/W | RC_32KHZ_CLK_EN | 1 | Writing 1 to this enables the RC 32KHz Clock Writing 0 to this disables the RC 32KHz Clock |
| 15:0 | - | Reserved | - | |

183 .ULP_CLKOSC_CTRL_REG

5 Resets

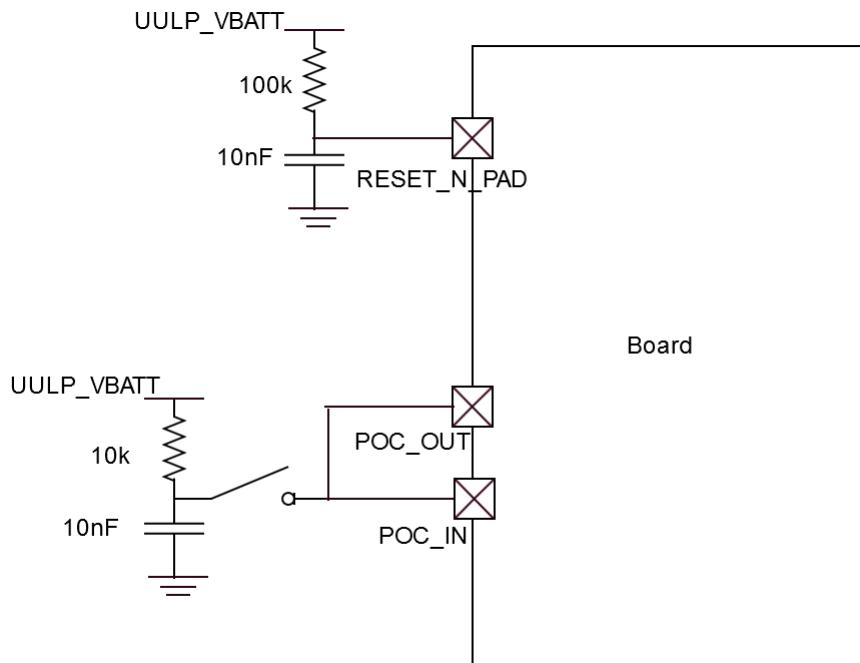
5.1 General Description

The device is brought into initial known state by applying the reset. The main resets in the chip are described in this section.

5.2 Features

1. Resets the design and set it to an initial state according to the reset source
2. Reset Resources
 - a. Primary reset , RESET_N_PAD
 - b. Power on reset, POC_IN
 - c. Black out Monitor
 - d. Watch Dog Reset
 - e. Reset request from host or Processor

5.3 Functional Description



External Connections on Board for Reset

The diagram above shows the required off chip components and the external board connections required for reset.

5.3.1 RESET_N_PAD

This is the primary reset to the chip. The external resistor and capacitor components are as shown in the diagram above. The complete design can be reset using this reset.

5.3.2 POC_IN

There is an internally generated Power On Reset in the design. That reset comes out as POC_OUT. This has to be looped back to POC_IN on board.

When the UULP_VBATT is less than or equal to 1.8V, external resistor and capacitor have to be connected on board as shown the diagram above.

5.3.3 Black Out Monitor

By default Blackout Monitor is disabled. It has to be enabled by enabling BLACKOUT_EN.

This is used to discharge RESET_N_PAD. Blackout occurs when UULP_VBATT is lower than 1.8V. Upon blackout event, RESET_N_PAD is autonomously pulled low. When the voltage on UULP_VBATT becomes greater than 1.825V, RESET_N_PAD charges again and the design will be out of reset.

The value of UULP_VBATT at which Black out Monitor triggers is programmable and is defined by the register bit BLACKOUT_TRIM

5.3.4 WatchDog Reset

Timer value is programmed in the watch dog timer. Once the timeout happens it generates an interrupt to the processor. If the processor does not service the interrupt, the watch dog reset is generated and the entire design is reset. The detailed description is present in WatchDog Timer block description

5.4 Reset request from host or Processor

Cortex M4 can request for reset on SYSRESETREQ pin. There can be reset requests from the host SDIO, SPI or USB. When there are any such requests from Cortex M4 or host, most of the design except for very few blocks are reset.

5.5 Register Summary

| Register Name | Offset | Description |
|------------------------|--------|-----------------------------------|
| BLACK_OUT_MON_EN_REG | 1E3 | Black Out Monitor Enable Register |
| BLACK_OUT_MON_TRIM_REG | 140 | Black Out Monitor Trim Register |

184 . Register Summary

5.6 Register Description

5.6.1 BLACK_OUT_MON_EN_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|---|
| 21:15 | R | Reserved | NA | Reserved |
| 14 | R/W | BLACKOUT_EN | 1'b0 | 0: Black Out Monitor is Disabled 1: Black Out Monitor is Enabled |
| 13:0 | R | Reserved | NA | Reserved |

185 . Black Out Monitor Enable Register Description

5.6.2 BLACK_OUT_MON_TRIM_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 21:12 | R | Reserved | NA | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|--|
| 11:10 | R/W | BLACKOUT_TRIM | 2'b00 | Used for trimming the voltage threshold which triggers the Blackout based reset. 0: 2.11V 1: 2.22V 2: 2.37V 3: 1.88V |
| 9:0 | R | Reserved | NA | Reserved |

186 . Black Out Monitor Trim Register Description

6 Interrupts

6.1 General Description

MCU uses Nested vectored interrupt controller (NVIC) for interrupts handling. It is programmable and its registers are located in the M4 System Control Space (SCS) of the memory map. The NVIC handles the exceptions and interrupt configurations, prioritization, and interrupt masking.

6.2 Features

The NVIC has the following features:

- Supports 99 interrupts
- Flexible exception and interrupt management
- Nested exception/interrupt support
- Vectored exception/interrupt entry
- Interrupt masking

6.3 Functional Description

6.3.1 Flexible exception and interrupt management

Each interrupt (apart from the NMI) can be enabled or disabled and can have its pending status set or cleared by software. The NVIC can handle various types of interrupt sources:

- Pulsed interrupt request e the interrupt request is at least one clock cycle long. When the NVIC receives a pulse at its interrupt input, the pending status is set and held until the interrupt gets serviced.
- Level triggered interrupt request e the interrupt source holds the request high until the interrupt is serviced.

The signal level at the NVIC input is active high. However, the actual external interrupt input on the micro-controller could be designed differently and is converted to an active high signal level by on-chip logic.

6.3.2 Nested exception/interrupt support

Each exception has a priority level. Some exceptions, such as interrupts, have programmable priority levels and some others (e.g., NMI) have a fixed priority level. When an exception occurs, the NVIC will compare the priority level of this exception to the current level. If the new exception has a higher priority, the current running task will be suspended. Some of the registers will be stored on the stack memory, and the processor will start executing the exception handler of the new exception. This process is called “preemption.” When the higher priority exception handler is complete, it is terminated with an exception return operation and the processor automatically restores the registers from stack and resumes the task that was running previously. This mechanism allows nesting of exception services without any software overhead.

6.3.3 Vectored exception/interrupt entry

When an exception occurs, the processor will need to locate the starting point of the corresponding exception handler. Traditionally software handles this step. The Cortex-M4 processor automatically locate the starting point of the exception handler from a vector table in the memory. As a result, the delays from the start of the exception to the execution of the exception handlers are reduced.

6.3.4 Interrupt masking

The NVIC in processor provide several interrupt masking registers such as the PRIMASK special register. Using the PRIMASK register, all exceptions excluding HardFault and NMI are disabled. This masking is useful for operations that should not be interrupted, like time critical control tasks or real-time multimedia codecs. Alternatively we can also use the BASEPRI register to select mask exceptions or interrupts which are below a certain priority level. The flexibility and capability of the NVIC also make the Cortex-M4 processors very easy to use, and provide better a system response by reducing the software overhead in interrupt processing, which also leads to smaller code size.

6.3.5 Vector table

When an exception event takes place and is accepted by the processor core, the corresponding exception handler is executed. To determine the starting address of the exception handler, a vector table mechanism is used. The vector table is an array of word data inside the system memory, each representing the starting address of one exception type. The vector table is relocatable and the relocation is controlled by a programmable register in the NVIC called the Vector Table Offset Register (VTOR). After reset, the VTOR is reset to 0; therefore, the vector table is located at address 0x0 after reset. The beginning of the memory space contains the vector table, and the first two words in the vector table are the initial value for the Main Stack Pointer (MSP), and the reset vector, which is the starting address of the reset handler. After these two words are read by the processor, the processor then sets up the MSP and the Program Counter (PC) with these values.

For example, if the reset is exception type 1, the address of the reset vector is 1 times 4 (each word is 4 bytes), which equals 0x00000004, and the NMI vector (type 2) is located at $2 \times 4 = 0x00000008$. The address 0x00000000 is used to store the starting value of the Main Stack Pointer.

6.3.6 Vectored Interrupt Table (VIT)

99 interrupts are mapped on NVIC. MCU HP peripheral interrupts, MCU ULP peripheral interrupts, MCU UULP peripheral interrupts and NWP peripheral interrupts are mapped into following Vectored interrupt table.

| Interrupt Number | Interrupt |
|------------------|--------------------------------|
| 19 : 0 | MCU ULP peripheral Interrupts |
| 29: 20 | MCU UULP peripheral Interrupts |
| 74:30 | MCU HP peripheral interrupts |
| 98:75 | NWP peripheral interrupts |

MCU HP Peripheral Interrupts

There are 45 MCU HP peripheral interrupts in 9116 chip. Following table provides the list of MCU HP peripheral interrupts and their interrupt number in vector interrupt table.

| Interrupt Number in VIT | MCU HP Peripheral interrupt |
|-------------------------|-----------------------------|
| 30 | Reserved |
| 31 | GPDMA interrupt |
| 32 | Reserved |
| 33 | MCU HP UDMA interrupt |
| 34 | SCT interrupt |
| 35 | HIF Interrupt 1 |
| 36 | HIF Interrupt 2 |
| 37 | SIO Interrupt |
| 38 | USART 1 Interrupt |
| 39 | UART 2 Interrupt |
| 40 | Reserved |

| Interrupt Number in VIT | MCU HP Peripheral interrupt |
|-------------------------|-----------------------------|
| 41 | EGPIO wakeup interrupts |
| 42 | I2C Interrupt |
| 43 | Reserved |
| 44 | SSI Slave Interrupt |
| 45 | Reserved |
| 46 | GSPI Master 1 Interrupt |
| 47 | SSI Master Interrupt |
| 48 | MCPWM Interrupt |
| 49 | QEI Interrupt |
| 51 : 50 | GPIO Group Interrupt |
| 59 : 52 | GPIO Pin Interrupt |
| 60 | QSPI2 Interrupt |
| 61 | I2C 2 Interrupt |
| 62 | Ethernet Interrupt |
| 63 | Ethernet PMT Interrupt |
| 64 | I2S master Interrupt |
| 65 | Reserved |
| 66 | Can 1 Interrupt |
| 67 | Reserved |
| 68 | SDMEM Interrupt |
| 69 | PLL clock ind Interrupt |
| 70 | Reserved |
| 71 | CCI system Interrupt Out |
| 72 | Reserved |
| 73 | USB interrupt |
| 74 | NWP P2P interrupt |

Register summary

Base Address: 0x4611_0000

| Register Name | Offset | Description |
|---------------------|--------|--|
| RESERVED | 0x00 | Reserved |
| M4SS_gpdma_INTR_SEL | 0x04 | MCU GPDMA interrupt selection register |
| RESERVED | 0x08 | Reserved |

| Register Name | Offset | Description |
|-------------------------|--------|---|
| M4SS_UDMA_INTR_SEL | 0x0C | MCU HP uDMA interrupt selection register |
| M4SS_SCT_INTR_SEL | 0x10 | SCT interrupt selection register |
| M4SS_gdma_INTR_SEL_TASS | 0x2C | The is used to select the GPDMA interrupts(8) that to be passed to NWP. Selected interrupts are ored mapped on to gdma bit in below MCU to NWP Interrupts |
| M4SS_UDMA_INTR_SEL_TASS | 0x34 | The is used to select the UDMA interrupts(32) that to be passed to NWP. Selected interrupts are ored mapped on to UDMA bit in below MCU to NWP Interrupts |
| M4SS_SCT_INTR_SEL_TASS | 0x38 | The is used to select the SCT interrupts(32) that to be passed to NWP. Selected interrupts are ored mapped on to SCT bit in below MCU to NWP Interrupts |

187 .MCU multi channel interrupt selection registers

Register Description

M4SS_gdma_INTR_SEL

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------------|-------------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | dma_m_interrupt_se 0 l | | This bit unmasks m th the GPDMA channel interrupt '1' – Unmasked '0' – Masked Upon read, If '0' seen upon reading this bit, this indicates that the interrupt is masked If '1' is read, this indicates interrupt is not masked. |

188 .M4SS_RPDMA_INTR_SEL Register Description

M4SS_UDMA_INTR_SEL

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------------|-------------|---|
| 31:0 | R/W | udma_m_interrupt_s 0 el | | This bit unmasks the m th UDMA channel interrupt '1' – Unmasked '0' – Masked Upon read, If '0' seen upon reading this bit, this indicates that the interrupt is masked If '1' is read, this indicates interrupt is not masked. |

189 .M4SS_UDMA_INTR_SEL Register Description

M4SS_SCT_INTR_SEL

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|---------------------|--------------------|---|
| 31:0 | R/W | sct_m_interrupt_sel | 0 | This bit unmasks the m th SCT channel interrupt '1' – Unmasked '0' – Masked Upon read, If '0' seen upon reading this bit, this indicates that the interrupt is masked If '1' is read, this indicates interrupt is not masked. |

190 . M4SS_SCT_INTR_SEL Register Description

NWP peripheral interrupts

There are 24 NWP peripheral interrupts are mapped on MCU NVIC. Following table provides the list of NWP peripheral interrupts and their interrupt number in vector interrupt table.

| Interrupt Number in VIT | NWP Peripheral interrupt |
|--------------------------------|----------------------------------|
| 75 | WLAN Band1 intr0 |
| 76 | WLAN Band1 intr1 |
| 77 | Reserved |
| 78 | Reserved |
| 79 | BT intr |
| 80 | ZB intr |
| 81 | Reserved |
| 82 | Modem disabled mode trigger intr |
| 83 | gpio intr |
| 84 | uart intr |
| 85 | watch dog level intr |
| 86 | Reserved |
| 87 | TASS - ECDH intr |
| 88 | TASS - DH intr |
| 89 | TASS - QSPI intr |
| 90 | Reserved |
| 91 | Sys Tick Timer |
| 92 | Real Timer interrupt |
| 93 | PLL lock interrupt |

| Interrupt Number in VIT | NWP Peripheral interrupt |
|-------------------------|--------------------------|
| 94 | Reserved |
| 95 | UART2 Interrupt |
| 96 | I2S Interrupt |
| 97 | I2C Interrupt |
| 98 | Reserved |

UULP Interrupt controller (Accessed only from MCU)

The group of UULP Peripheral interrupts will be converted to 10 UULP interrupt and mapped to 29-20 interrupt in MCU.

Features

- Supports 19 UULP peripheral interrupts and mapped them into 10 MCU interrupts.
- UULP GPIO interrupts have additional functionality of rise edge, fall edge and level detection.
- UULP interrupts other than GPIO interrupts will have only rise edge detection.
- Each interrupts can be masked or unmasked based on the requirement.
- UULP GPIO pins unmasked status can be read from register.

NPSS Interrupt numbers

19 NPSS interrupts mapping and the description of the interrupts are given below table. The priority of the interrupt decreases as the interrupt number increases.

| NPSS interrupt number | Interrupt Number in VIT | NPSS Interrupt |
|-----------------------|-------------------------|---------------------|
| 0 | 20 | uulp_wdt_intr |
| 1-5 | 21 | uulp_gpio_intr |
| 6-9 | 22 | uulp_cmp_intr |
| 10 | 22 | uulp_rf wakeup_intr |
| 11 | 23 | uulp_bod_intr |
| 12 | 24 | uulp_button_intr |
| 13 | 25 | uulp_sdc_intr |
| 14 | 26 | uulp_wireless_intr |
| 15 | 27 | uulp_wakeup_intr |
| 16 | 28 | uulp_alarm_intr |
| 17 | 29 | uulp_sec_intr |
| 18 | 29 | uulp_msec_intr |

Programming Sequence

- All UULP interrupts to the Processor are masked by default. To unmask any interrupt, set the corresponding bit in the `UULP_INTR_MASK_CLR_REG` register.
- To mask the interrupts, set the corresponding bit in the `UULP_INTR_MASK_SET_REG` register.
- When the interrupt is raised, check `UULP_INTR_STATUS_REG` to find out which interrupt among the mapped intr is raised.
- Clear it by setting the corresponding bit in the `UULP_INTR_CLEAR_REG` register so that the interrupt will not be seen again.
- UULP GPIO interrupts will have additional functionality of fall edge detection and level detection. GPIO interrupts will be rise detection by default.
- Set the required interrupt detection mode for each GPIO in `UULP_GPIO_CONFIG_REG`.
- To clear fall/rise edge interrupt `UULP_INTR_CLEAR_REG[5:0]` can be used. But to clear level interrupts, level0/level1 enable bits in `UULP_GPIO_CONFIG_REG` as to be reset.

Register Summary

All the below registers are 16 bit and 32 bit accessible

Base Address: 0x1208_0000

| Register Name | Offset | Description |
|-------------------------------------|--------|------------------------------------|
| <code>UULP_INTR_MASK_SET_REG</code> | 0x0 | UULP interrupt mask set register |
| <code>UULP_INTR_MASK_CLR_REG</code> | 0x4 | UULP interrupt mask clear register |
| <code>UULP_INTR_CLEAR_REG</code> | 0x8 | UULP interrupt clear register |
| <code>UULP_INTR_STATUS_REG</code> | 0xC | UULP interrupt status register |
| <code>UULP_GPIO_CONFIG_REG</code> | 0x10 | UULP GPIO configuration register |
| <code>UULP_GPIO_STATUS</code> | 0x14 | UULP GPIO status register |
| <code>M4_WIC_CLEAR_REG</code> | 0x18 | MCU WIC clear register |
| <code>M4_ULP_SLP_STATUS_REG</code> | 0x1C | MCU ULP sleep status register |
| <code>M4ULP_ISO_ENABLE_REG</code> | 0x20 | MCU ULP isolation enable register |
| <code>M4ULP_RST_CTRL_REG</code> | 0x24 | MCU ULP reset control register |

191 . UULP Peripheral Interrupt controller Register Table

Register Description

`UULP_INTR_MASK_SET_REG`

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|--------------------------|
| 31:19 | R | Reserved | 1b0 | Reserved for future use. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------------|-------------|---|
| 18:0 | R/W | UULP_INTR_n_MASK1b1 | | This bit is used to mask UULP interrupt 'n' For write operation, '1'- Mask Interrupt '0'- Writing a zero into this has no effect. For read operation, '1' - Interrupt masked '0' - Not masked |

192 . UULP_INTR_MASK_SET_REG Description

UULP_INTR_MASK_CLR_REG

| Bit | Access | Function | Reset Value | Description |
|----------|--------|---------------------|-------------|---|
| [31:19] | R | Reserved | 1b0 | Reserved for future use. |
| n (0-18) | R/W | UULP_INTR_n_MASK1b1 | | This bit is used to mask UULP interrupt 'n' For write operation, '1'- Unmask Interrupt '0'- Writing a zero into this has no effect. For read operation, '1' – Interrupt masked '0' – Not masked |

193 . UULP_INTR_MASK_CLR_REG description

UULP_INTR_CLEAR_REG

| Bit | Access | Function | Reset Value | Description |
|----------|--------|----------------------|-------------|--|
| [31:19] | R | Reserved | 1b0 | Reserved for future use. |
| n (0-18) | WO | UULP_INTR_n_CLEAR1b1 | | This bit is used to clear UULP interrupt 'n' Main interrupt has to cleared at the source For write operation, '1'- Clears the Interrupt '0'- Writing a zero into this has no effect. |

194 . UULP_INTR_CLEAR_REG Description

UULP_INTR_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|---------|--------|----------|-------------|--------------------------|
| [31:19] | R | Reserved | 1b0 | Reserved for future use. |

| Bit | Access | Function | Reset Value | Description |
|----------|--------|---------------------|-------------|--|
| n (0-18) | RO | UULP_INTR_n_STAT US | 1b0 | This bit is used to read the masked UULP interrupt 'n' status For Read operation, '1'- indicates that the 'n'th UULP interrupt has been raised '0'- indicates that the interrupt is masked or not been raised |

195 .UULP_INTR_STATUS_REG Description

UULP_GPIO_CONFIG_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|---|
| 31:29 | R | Reserved | 5b0 | Reserved for future use. |
| 28 | W/R | level1 enable 4 | 1'b0 | '1'- Enables level 1 interrupt detection for UULP_VBAT_GPIO_4 '0'- Disables level 1 interrupt detection for UULP_VBAT_GPIO_4 |
| 27 | W/R | level1 enable 3 | 1'b0 | '1'- Enables level 1 interrupt detection for UULP_VBAT_GPIO_3 '0'- Disables level 1 interrupt detection for UULP_VBAT_GPIO_3 |
| 26 | W/R | level1 enable 2 | 1'b0 | '1'- Enables level 1 interrupt detection for UULP_VBAT_GPIO_2 '0'- Disables level 1 interrupt detection for UULP_VBAT_GPIO_2 |
| 25 | W/R | level1 enable 1 | 1'b0 | '1'- Enables level 1 interrupt detection for UULP_VBAT_GPIO_1 '0'- Disables level 1 interrupt detection for UULP_VBAT_GPIO_1 |
| 24 | W/R | level1 enable 0 | 1'b0 | '1'- Enables level 1 interrupt detection for UULP_VBAT_GPIO_0 '0'- Disables level 1 interrupt detection for UULP_VBAT_GPIO_0 |
| 23:21 | R | Reserved | 5b0 | Reserved for future use. |
| 20 | W/R | level0 enable 4 | 1'b0 | '1'- Enables level 0 interrupt detection for UULP_VBAT_GPIO_4 '0'- Disables level 0 interrupt detection for UULP_VBAT_GPIO_4 |
| 19 | W/R | level0 enable 3 | 1'b0 | '1'- Enables level 0 interrupt detection for UULP_VBAT_GPIO_3 '0'- Disables level 0 interrupt detection for UULP_VBAT_GPIO_3 |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|---|
| 18 | W/R | level0 enable 2 | 1'b0 | '1'- Enables level 0 interrupt detection for UULP_VBAT_GPIO_2 '0'- Disables level 0 interrupt detection for UULP_VBAT_GPIO_2 |
| 17 | W/R | level0 enable 1 | 1'b0 | '1'- Enables level 0 interrupt detection for UULP_VBAT_GPIO_1 '0'- Disables level 0 interrupt detection for UULP_VBAT_GPIO_1 |
| 16 | W/R | level0 enable 0 | 1'b0 | '1'- Enables level 0 interrupt detection for UULP_VBAT_GPIO_0 '0'- Disables level 0 interrupt detection for UULP_VBAT_GPIO_0 |
| 15:13 | R | Reserved | 5b0 | Reserved for future use. |
| 12 | W/R | Fall edge enable 4 | 1'b0 | '1'- Enables fall edge interrupt detection for UULP_VBAT_GPIO_4 '0'- Disables fall edge interrupt detection for UULP_VBAT_GPIO_4 |
| 11 | W/R | Fall edge enable 3 | 1'b0 | '1'- Enables fall edge interrupt detection for UULP_VBAT_GPIO_3 '0'- Disables fall edge interrupt detection for UULP_VBAT_GPIO_3 |
| 10 | W/R | Fall edge enable 2 | 1'b0 | '1'- Enables fall edge interrupt detection for UULP_VBAT_GPIO_2 '0'- Disables fall edge interrupt detection for UULP_VBAT_GPIO_2 |
| 9 | W/R | Fall edge enable 1 | 1'b0 | '1'- Enables fall edge interrupt detection for UULP_VBAT_GPIO_1 '0'- Disables fall edge interrupt detection for UULP_VBAT_GPIO_1 |
| 8 | W/R | Fall edge enable 0 | 1'b0 | '1'- Enables fall edge interrupt detection for UULP_VBAT_GPIO_0 '0'- Disables fall edge interrupt detection for UULP_VBAT_GPIO_0 |
| 7:5 | R | Reserved | 5b0 | Reserved for future use. |
| 4 | W/R | Rise edge enable 4 | 1'b1 | '1'- Enables rise edge interrupt detection for UULP_VBAT_GPIO_4 '0'- Disables rise edge interrupt detection for UULP_VBAT_GPIO_4 |
| 3 | W/R | Rise edge enable 3 | 1'b1 | '1'- Enables rise edge interrupt detection for UULP_VBAT_GPIO_3 '0'- Disables rise edge interrupt detection for UULP_VBAT_GPIO_3 |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------|-------------|---|
| 2 | W/R | Rise edge enable 2 | 1'b1 | '1'- Enables rise edge interrupt detection for UULP_VBAT_GPIO_2 '0'- Disables rise edge interrupt detection for UULP_VBAT_GPIO_2 |
| 1 | W/R | Rise edge enable 1 | 1'b1 | '1'- Enables rise edge interrupt detection for UULP_VBAT_GPIO_1 '0'- Disables rise edge interrupt detection for UULP_VBAT_GPIO_1 |
| 0 | W/R | Rise edge enable 0 | 1'b1 | '1'- Enables rise edge interrupt detection for UULP_VBAT_GPIO_0 '0'- Disables rise edge interrupt detection for UULP_VBAT_GPIO_0 |

196 .UULP_GPIO_CONFIG Reg Description

UULP_GPIO_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------|-------------|---|
| 31:3 | R | Reserved | 29b0 | Reserved for future use. |
| 4 | R | GPIO4 status | 1'b0 | Gives the pin status for UULP_VBAT_GPIO_4 |
| 3 | R | GPIO3 status | 1'b0 | Gives the pin status for UULP_VBAT_GPIO_3 |
| 2 | R | GPIO2 status | 1'b0 | Gives the pin status for UULP_VBAT_GPIO_2 |
| 1 | R | GPIO1 status | 1'b0 | Gives the pin status for UULP_VBAT_GPIO_1 |
| 0 | R | GPIO0 status | 1'b0 | Gives the pin status for UULP_VBAT_GPIO_0 |

197 .UULP_GPIO_STATUS_REG Description

M4_WIC_CLEAR_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------|-------------|---|
| 31:2 | R | Reserved | 31'd0 | |
| 2 | R/W | enable_negedge_ulp | 1'b1 | Enables the negedge path in ULP Mode. This needs to be programmed before switching to ULP mode. This mode has to be enabled only if we intend to do supply switching for processor in ULP mode. When supply is switched level shifters will be enabled and we must use negedge path to save level shifters power due to combi toggles 0 - Posedge path 1 - Negedge path |
| 0 | R/W | WIC CLEAR | 1'b0 | Clear bit for WIC functionality |

198 .M4_WIC_CLEAR_REG Description

M4_ULP_SLP_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------------|-------------|--|
| 31:5 | R | Reserved | 28'd0 | |
| 4 | R | ulp_wakeup_por | 1'b9 | Indicates POR status. |
| 3 | R | ulp_mode_switched_npss | 1'b0 | Indicates the status of Physical switching to ULP Mode operation 0 - PS4 State 1 - PS2 State |
| 2 | R | ulp_mode_aftr_clk_sw | 1'b0 | Indicates the status of functional switching to ULP Mode operation 0 - PS4 state 1 - PS2 state |
| 1 | R | RAM RETENTION STATUS | 1'b0 | Indicates the status of Ram retention on Wakeup 0 - RAM not retained 1 - RAM retained |
| 0 | R | ULP WAKEUP | 1'b0 | Status Indication for Wakeup mode 0 - First Bootup 1 - ULP Wakeup |

199 . M4_ULP_SLP_STATUS_REG Description

M4ULP_ISO_ENABLE_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------|-------------|---|
| 31:23 | R | Reserved | 9'd0 | |
| 22:16 | R/W | ISO_ENABLE_REG_SRAM | 7'd0 | Enables isolation on outputs of M4 ULP-SRAM Power Domain. This is used on power domain controls bypass mode 0 - Disables isolation 1 - Enables isolation |
| 15:9 | R | Reserved | 7'd0 | |
| 8 | R/W | ISO_ENABLE_REG_M4_ROM | 1'b0 | Enables isolation on outputs of M4 ROM Power Domain. This is used on power domain controls bypass mode 0 - Disables isolation 1 - Enables isolation |
| 7:3 | R | Reserved | 5'd0 | |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------------------|-------------|---|
| 2 | R/W | ISO_ENABLE_REG_M4_ FPU | 1'b0 | <p>Enables isolation on outputs of M4 FPU Power Domain.</p> <p>This is used on power domain controls bypass mode</p> <p>0 - Disables isolation</p> <p>1 - Enables isolation</p> |
| 1 | R/W | ISO_ENABLE_REG_M4_ DEBUG | 1'b0 | <p>Enables isolation on outputs of M4 Debug Power Domain.</p> <p>This is used on power domain controls bypass mode</p> <p>0 - Disables isolation</p> <p>1 - Enables isolation</p> |
| 0 | R | Reserved | 1'b0 | |

200 . M4ULP ISO_ENABLE REG Description

M4ULP_RST_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------------------|-------------|--|
| 31:9 | R | Reserved | 23'd0 | |
| 8 | R/W | RS_CTRL_REG_M4_RO M | 1'b0 | <p>Enables Reset for M4 ROM Power Domain.</p> <p>This is used on power domain controls bypass mode</p> <p>0 - Out of Reset</p> <p>1 - In Reset</p> |
| 7:3 | R | Reserved | 5'd0 | |
| 2 | R/W | RST_CTRL_REG_M4_F PU | 1'b0 | <p>Enables Reset for M4 FPU Power Domain.</p> <p>This is used on power domain controls bypass mode</p> <p>0 - Out of Reset</p> <p>1 - In Reset</p> |
| 1 | R/W | RST_CTRL_REG_M4_D EBUG | 1'b0 | <p>Enables Reset for M4 DEBUG Power Domain.</p> <p>This is used on power domain controls bypass mode</p> <p>0 - Out of Reset</p> <p>1 - In Reset</p> |
| 0 | R | Reserved | 1'b0 | |

201 . M4ULP_RST_CTRL_REG Description

MCU ULP peripheral interrupts

There are 20 MCU ULP peripheral interrupts mapped on MCU NVIC. Following table provides the list of MCU ULP peripheral interrupts and their interrupt number in vector interrupt table.

| Interrupt Number in VIT | MCu ULP Peripheral interrupt |
|-------------------------|---|
| 19 | egpio_group interrupt |
| 18 | egpio_pin interrupt (OR'ed egpio_pin interrupt) |
| 17 | FIM interrupt |
| 16 | ssi_mst_intr |
| 15 | ir_wakeup_intr |
| 14 | i2s_intr |
| 13 | i2c_intr |
| 12 | uart_intr |
| 11 | aux_adc_dac_intr |
| 10 | udma_intr |
| 9 | ulp_gpio_wakeup_intr |
| 8:7 | comp_intr (comparator Interrupt) |
| 6 | cap_sense_intr (Cap_sense_wake_up or Cap_sense_intr) |
| 5:2 | ulp_timer_intr_status (Timer0, Timer 1, Timer2, Timer3) |
| 1:0 | vad_intr (0: Vad_intr_ping, 1: Vad_intr_pong) |

7 Power Architecture

7.1 General Description

The Power Control Hardware implements the control sequences for transitioning between different power states (Active/Standy/Sleep/Shutdown) and the power control for different Group of Peripherals. In-addition, wakeup from any of the Standby/Sleep/Shutdown states based on hardware events or peripheral interrupts is supported.

The Standby and Shutdown states can be reached from Active mode only and through a WFI instruction. Wakeup from Standby/Sleep/Shutdown states is through a hardware event or interrupt (Peripheral or External). The different wakeup interrupts are listed in the [NVIC section of the Processor](#).

Different SRAM sizes and Peripherals are available in each Active/Sleep states which is described in the functional description.

7.2 Features

- Two integrated buck switching regulators (High performance and ULP) to enable efficient Dynamic Voltage Scaling across wide operating mode currents ranging from <1uA to 300mA
- High performance and ultra-low-power MCU peripheral subsystems and buses.
- Multiple voltage domains with Independent voltage scaling of each domain.
- Fine grained power-gating including peripherals, buses and pads, thereby reducing power consumption when the peripheral/buses/pads are inactive.
- Multiple Active states using "gear-shifting" approach based on processing requirements, thereby reducing power consumption for low-power applications.
- Flexible switching between different Active states with controls from Software.
- Hardware based wakeup from Standby/Sleep/Shutdown states.
- All the peripherals are clock gated by default thereby reducing the power consumption in inactive state.
- Low wakeup times as configurable by Software.

7.3 Functional Description

7.3.1 Power Domains

All the Applications, High Speed Interfaces and Peripherals are segregated into multiple power domains to achieve lower current consumption when they are inactive. At reset, all the domains are powered ON.

The programming for power control for PLL Core is described in Clocking section.

The table below describes the different group of peripherals for which power is controlled through software

| S.No | Section | Domain Name | Functionality of the Power Domain |
|------|----------------------|--------------|--|
| 1 | APPLICATIONS | DEBUG | Debug Functionality for Cortex-M4F |
| 2 | | FPU | Floating Point Unit for Cortex-M4F |
| 3 | | ICACHE | ICache for the Cortex-M4F Processor |
| 4 | | ROM | ROM Core/Interface |
| 5 | | SRAM | SRAM Banks |
| 6 | HIGH SPEED INTERFACE | QSPI | Quad/Octa SPI SDR/DDR Flash Interface |
| 7 | | ETHERNET | 10/100 Ethernet Controller with RMII |
| 8 | | SDMEM | SDMEM, eMMC, SDIO 3.0 Interface |
| 9 | | CCI | Chip to Chip Interface |
| 10 | HP-PERIPHERALS | USB | USB 2.0 OTG |
| 11 | | PERI-DOMAIN1 | SPI/SSI Master, I2C, USART and Micro-DMA Controller. |

| S.No | Section | Domain Name | Functionality of the Power Domain |
|------|-------------------------|-----------------|--|
| 12 | | PERI-DOMAIN2 | UART, SPI/SSI Slave, Generic-SPI Master, Config Timer, CAN, Random-Number Generator and CRC Accelerator. |
| 13 | | PERI-DOMAIN3 | SIO, I2C, I2S Master/Slave, QEI and MCPWM. |
| 14 | | DMA | General Purpose DMA Controller |
| 15 | | SDIO-SPI | SDIO 2.0 Slave, SPI Slave. |
| 16 | | EFUSE | EFUSE for configuration information |
| 17 | HIGH SPEED FLASH MEMORY | FLASH-LDO | LDO-FL 1.8 for Flash Memory |
| 18 | HIGH-FREQ-PLL | PLL-REGISTERS | PLL Programming Registers for High frequency clocks. |
| 19 | DDR-FLASH-DLL | QSPI-DLL | DLL for Quad/Octa DDR Flash Interface |
| 20 | ULP-PERIPHERALS | DMA | Micro-DMA Controller |
| 21 | | IR | IR Receiver |
| 22 | | ADC-DAC | ADC and DAC Controller |
| 23 | | I2C | I2C Master/Slave |
| 24 | | SSI | SPI/SSI Master |
| 25 | | UART | UART |
| 26 | | VAD | Voice Activity Detection |
| 27 | | TOUCH | Capacitive Touch Sensor Controller |
| 28 | | FIM | IIR/FIR Filter, Interpolation, Matrix Multiplication, etc. |
| 29 | | TIMER | Timers |
| 30 | | WDT | Window-Watch Dog Timer |
| 31 | | TS | Temperature Sensor Controller |
| 32 | | PS | Process Sensor Controller |
| 33 | | RTC | Real-Time Clock |
| 34 | UULP-PERIPHERALS | STORAGE-DOMAIN1 | Storage Flops - Set1. Contains 8bytes |
| 35 | | STORAGE-DOMAIN2 | Storage Flops - Set2. Contains 8bytes |
| 36 | | STORAGE-DOMAIN3 | Storage Flops - Set3. Contains 16bytes |
| 37 | | SLEEP-FSM | FSM for Sleep/Wakeup |
| 38 | | CLOCK-CALIB | Calibration block for Sleep Clock. |
| 39 | | BBFFS | Programming Registers which can be retained during sleep. |
| 40 | | DS-TIMER | DEEP SLEEP Timer. |
| 41 | | TIMESTAMP | Timestamping Controller. |
| 42 | | LP-FSM | Low-Power FSM |
| 43 | | RETEN | Retention Flops which can be retained during sleep. |
| 44 | Analog-PERIPHERALS | WuRx | Wake-Fi Rx |
| 45 | | Aux-ADC | Auxillary ADC |
| 46 | | Aux-DAC | Auxillary DAC |
| 47 | | BOD | Brown-Out Detector |
| 48 | | RC-32MHz | High Frequency RC Oscillator |
| 49 | | RO-HF | High Frequency RO Oscillator |
| 50 | | FREQ-DOUBLER | Frequency Doubler |
| 51 | | XTAL-32KHz | Low Frequency XTAL Oscillator |
| 52 | | RO-32KHz | Low Frequency RO Oscillator |
| 53 | | RC-32KHz | Low Frequency RC Oscillator |
| 54 | | TS-RO | RO based Temperature Sensor |

| S.No | Section | Domain Name | Functionality of the Power Domain |
|------|---------|-------------|-----------------------------------|
| 55 | | TS-BJT | BJT based Temperature Sensor |

202 . List of Power Domains

The SRAM is also segregated into multiple power domains to achieve lower current consumption as per the Memory requirement. The power for the SRAM domains in active states can be controlled in the following manners

- SRAM Domains as described in the table below can be powered down for unused SRAM sections. The RAM contents are not retained in this mode
- Deep-Sleep (Lower power consumption) mode. The RAM contents are retained in this mode. The SRAM is not accessible in this state. This is configurable on a Bank granularity.

The table below describes the segregation of power domains for SRAM (400KB). The addressing for these banks are described in [Memory Architecture section](#).

| S.No | Section | Domain Name | Functionality of the Power Domain |
|------|----------|-------------|-----------------------------------|
| 1 | HP-SRAM1 | HP-SRAM1-1 | 16KB of SRAM (1x Banks) |
| 2 | | HP-SRAM1-2 | 32KB of SRAM (2x Banks) |
| 3 | | HP-SRAM1-3 | 16KB of SRAM (1x Banks) |
| 4 | HP-SRAM2 | HP-SRAM2-1 | 16KB of SRAM (1x Banks) |
| 5 | | HP-SRAM2-2 | 32KB of SRAM (2x Banks) |
| 6 | | HP-SRAM2-3 | 80KB of SRAM (5x Banks) |
| 7 | | HP-SRAM2-4 | 64KB of SRAM (4x Banks) |
| 8 | LP-SRAM | LP-SRAM-1 | 4KB of SRAM (1x Banks) |
| 9 | | LP-SRAM-2 | 4KB of SRAM (1x Banks) |
| 10 | | LP-SRAM-3 | 4KB of SRAM (1x Banks) |
| 11 | | LP-SRAM-4 | 4KB of SRAM (1x Banks) |
| 12 | | LP-SRAM-5 | 32KB of SRAM (2x Banks) |
| 13 | | LP-SRAM-6 | 64KB of SRAM (4x Banks) |
| 14 | | LP-SRAM-7 | 16KB of SRAM (1x Banks) |
| 15 | ULP-SRAM | ULP-SRAM-1 | 2KB of SRAM (1x Banks) |
| 16 | | ULP-SRAM-2 | 2KB of SRAM (1x Banks) |
| 17 | | ULP-SRAM-3 | 2KB of SRAM (1x Banks) |
| 18 | | ULP-SRAM-4 | 2KB of SRAM (1x Banks) |
| 19 | | ULP-SRAM-5 | 2KB of SRAM (1x Banks) |
| 20 | | ULP-SRAM-6 | 2KB of SRAM (1x Banks) |
| 21 | | ULP-SRAM-7 | 2KB of SRAM (1x Banks) |
| 22 | | ULP-SRAM-8 | 2KB of SRAM (1x Banks) |

203 . List of SRAM Power Domains

7.3.2 Programming Sequence

The power for the above domains except for SRAM can be controlled as described below

- APPLICATIONS, HIGH-SPEED INTERFACES and HP-PERIPHERALS
 - Program the particular bit in "[Power_Control_SET](#)" and "[Power_Control_CLEAR](#)" Register.
- HIGH SPEED FLASH MEMORY
 - Program the particular bit in "[PMU_LDO_CTRL_REG_SET](#)" and "[PMU_LDO_CTRL_REG_CLEAR](#)" Register
- HIGH-FREQ-PLL
 - Program the particular bit in "[PLL_Power_Control](#)" Register.

- DDR-FLASH-DLL
 - Program the particular bit in "[DLL_Power_Control](#)" Register.
- ULP-PERIPHERALS
 - Program the particular bit in "[ULP_Peripheral_Power_Control_SET](#)" and "[ULP_Peripheral_Power_Control_CLEAR](#)" Register.
- UULP-PERIPHERALS
 - Program the particular bit in "[UULP_Peripheral_Power_Control_SET](#)" and "[UULP_Peripheral_Power_Control_CLEAR](#)" Register.
 - Program the particular bit in "[FSM_CTRL_POWER_DOMAINS](#)" Register.
- Analog-PERIPHERALS
 - Program the particular bit in "[Analog_Power_Control](#)" Register. <TBD>

The programming for the power controls of the SRAM domains are described below

- SRAM Domain Power Up
 - For HP-SRAM1, HP-SRAM2 and LP-SRAM, configure the particular bit in "[SRAM_Power_Control_REG1_SET](#)" and "[SRAM_Power_Control_REG2_SET](#)" Register.
 - For ULP-SRAM, configure the particular bit in "[ULP_SRAM_Power_Control_REG1_SET](#)" and "[ULP_SRAM_Power_Control_REG2_SET](#)" Register.
 - Both the Registers need to be configured for achieving the functionality.
- SRAM Domain Power Down
 - For HP-SRAM1, HP-SRAM2 and LP-SRAM, configure the particular bit in "[SRAM_Power_Control_REG1_CLEAR](#)" and "[SRAM_Power_Control_REG2_CLEAR](#)" Register.
 - For ULP-SRAM, configure the particular bit in "[ULP_SRAM_Power_Control_REG1_CLEAR](#)" and "[ULP_SRAM_Power_Control_REG2_CLEAR](#)" Register.
 - Both the Registers need to be configured for achieving the functionality.
- SRAM Standby
 - For HP-SRAM1, HP-SRAM2 and LP-SRAM, configure the particular bit in "[SRAM_Power_Control_REG4_SET](#)" Register. This has to be cleared for SRAM accesses by configuring the particular bit in "[SRAM_Power_Control_REG4_CLEAR](#)" Register.
 - For ULP-SRAM, configure the particular bit in "[ULP_SRAM_Power_Control_REG4_SET](#)" Register. This has to be cleared for SRAM accesses by configuring the particular bit in "[ULP_SRAM_Power_Control_REG4_CLEAR](#)" Register.

The details of the above Registers are provided in the Register Description Section below.

7.4 Voltage Domains

All the Applications, High Speed Interfaces and Peripherals are segregated into multiple voltage domains to configure the operating voltages in different power states. This section describes the voltage domains and voltage source options available for each domain. These are configured based on the Power state which the device is operating in. The voltage for each domain can be shut-off during sleep by configuring the source to LDO SoC 1.1 (This supply is turned OFF during Sleep).

The table below lists down the different voltage sources and the possible output voltages of each source at different Power states. The voltage sources are described in detail in the Power Management Section.

| S.No | Voltage Source | Possible O/P Voltage |
|------|----------------|----------------------|
| 1 | LDO SoC 1.1 | 1.1V |
| | | 1.0V |
| 2 | DC-DC 0.95 | 1.0V |
| 3 | LDO 0.7V | 0.7V |

204 . List of Voltage Sources

The table below lists down the different voltage domains and the possible voltage sources for each domain. The voltage source for each domain in different power-states are defined in the [Power States](#) section below.

| S.No | Voltage Domain | Functionality | LDO SoC 1.1 | DC-DC 0.95 | LDO 0.7V |
|------|------------------------|---|-------------|------------|----------|
| 1 | PROC-DOMAIN | Processor, DEBUG, FPU | Yes | Yes | Yes |
| 2 | HIGH-VOLTAGE-DOMAIN | ICACHE, HIGH-SPEED- INTERFACES, HP-PERIPHERALS, HP-SRAM | Yes | No | No |
| 3 | LOW-VOLTAGE-LPRAM-16KB | LP-SRAM-1, LP-SRAM-2, LP-SRAM-3, LP-SRAM-4, | Yes | Yes | No |
| 4 | LOW-VOLTAGE-LPRAM | ROM LP-SRAM-5, LP-SRAM-6, LP-SRAM-7 | Yes | Yes | No |
| 5 | LOW-VOLTAGE-ULPPERIPH | ULP-PERIPHERALS | Yes | Yes | No |
| 6 | LOW-VOLTAGE-ULPRAM | ULP-SRAM | Yes | Yes | No |
| 7 | LOW-VOLTAGE-UULPPERIPH | UULP-PERIPHERALS | No | Yes | No |

205 . List of Voltage Domains

7.5 Power States

The power states available in different modes including the power variants of the processor are listed below

- Reset State
- Active States
 - PS1
 - PS2
 - PS3
 - PS4
- Standby States
 - PS2-STANDBY
 - PS3-STANDBY
 - PS4-STANDBY
- Sleep States
 - PS2-SLEEP
 - PS3-SLEEP
 - PS4-SLEEP

- Shutdown States
 - PS0

After reset, the processor starts in PS4 state which is the highest activity state where the full functionality is available. The other Active states (PS2/PS3) will have limited functionality or Processing power.

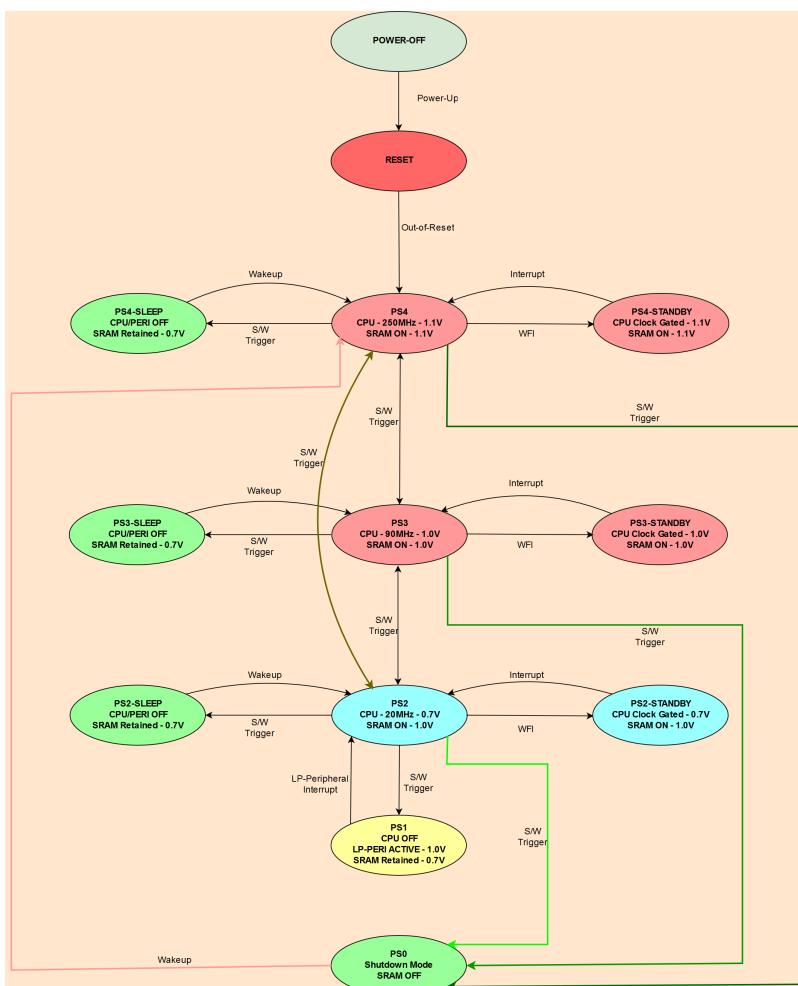
A transition from Active states (PS2/PS3/PS4) to any other state can only be triggered by software.

A transition from Standby/Sleep/Shutdown states can be triggered by an enabled interrupt as configured by software before entering these states.

A transition from Standby/Sleep to Active state is possible from where these states are entered.

There are different wakeup sources available in each Standby/Sleep/Shutdown states which is listed in the [Wakeup Sources](#) section below.

The figure below shows the transitions between different power states. The programming sequence for each transition is mentioned in the [Transitions](#) section below.



Power State Transitions

7.5.1 PS4

This is an Active state where the complete functionality is available. The CPU, Peripherals and SRAM operate on the LDO SoC 1.1V Supply at voltage of 1.1V.

This functionality available in this state includes the following

- Maximum CPU Operating frequency of 180MHz. The CPU can operate on the HIGH-FREQ-PLL output clocks.

- APPLICATIONS - DEBUG, FPU, ICACHE and ROM.
- HIGH SPEED INTERFACE - as listed in the Power Domains table above.
- HIGH-FREQ-PLL - as listed in the Power Domains table above.
- DDR-FLASH-DLL - as listed in the Power Domains table above.
- All the Peripherals consisting of HP-PERIPHERALS, ULP-PERIPHERALS, UULP-PERIPHERALS and Analog-PERIPHERALS - as listed in the [Power Domains](#) section above.
- All the GPIO's - 64(GPIO) + 16(ULP-GPIO) + 5(UULP Vbat GPIO)
- Complete SRAM of 400KB (HP-SRAM1, HP-SRAM2, LP-SRAM and ULP-SRAM).

7.5.2 PS3

This is an Active state where the complete functionality is available similar to PS4 state and operates at a lower voltage thereby reducing current consumption. The CPU, Peripherals and SRAM operate on the LDO SoC 1.1 Supply with output voltage of 1.0V. The Maximum CPU frequency is limited to 90MHz in this state.

7.5.3 PS2

This is an Active state where a limited set of functionality is available and operates a much lower voltage compared to PS3/PS4 thereby achieving lower current consumption. The CPU, Peripherals and SRAM can operate at different voltages and are configurable by software before entering this state.

The functionality available in this state includes the following

- CPU Operating frequency depends on the voltage source selected for PS2 state. The CPU operates on the ULP-Peripheral AHB Interface clock which is described in [ULP Clocking](#) section.
 - If LDO 0.7V is used, Maximum frequency is 20MHz.
 - If DC-DC 0.95 is used, Maximum frequency is 50MHz.
- APPLICATIONS - DEBUG, FPU and ROM.
- Limited peripherals consisting of ULP-PERIPHERALS, UULP-PERIPHERALS and Analog-PERIPHERALS - as listed in the [Power Domains](#) table above.
- 21 GPIO's are available - 16(ULP-GPIO) + 5(UULP Vbat GPIO)
- Total SRAM of 144KB (LP-SRAM and ULP-SRAM).

7.5.4 PS1

This state can be entered from PS2 only through a Software Instruction. The CPU is power-gated and a limited set of peripherals are active. The peripheral interrupts are used as wakeup source or to trigger sleep once the peripheral functionality is complete. The Peripherals and SRAM operate at the same voltage as PS2 state. The peripherals need to be configured by the Software for the defined functionality in the PS2 state before entering this state.

The functionality available in this state includes the following

- Limited peripherals consisting of ULP-PERIPHERALS, UULP-PERIPHERALS and Analog-PERIPHERALS - as listed in the [Power Domains](#) table above.
- 21 GPIO's are available - 16(ULP-GPIO) + 5(UULP Vbat GPIO)
- SRAM of 128KB (LP-SRAM) can be retained in this state.
- SRAM of 16KB (ULP-SRAM) is active for Peripheral functionality.

Wakeup sources for this state are defined in the [Wakeup source](#) section below.

7.5.5 STANDBY

This includes multiple states like PS4-STANDBY, PS3-STANDBY and PS2-STANDBY. These are Standby states entered from PS4/PS3/PS2 state through a WFI instruction. CPU is clock gated in this state.

All the Interrupts in the NVIC table as described in [NVIC](#) Section will act as a wakeup source in PS4-STANDBY and PS3-STANDBY state. Wakeup sources for PS2-STANDBY state are defined in the [Wakeup source](#) section below.

7.5.6 SLEEP

This includes multiple states like PS4-SLEEP PS3-SLEEP and PS2-SLEEP/PS1-SLEEP which can be entered from PS4, PS3 and PS2 state respectively through a Software instruction. In addition, PS2-SLEEP state can be entered from PS1 state through a peripheral interrupt. The CPU is power-gated and a much lower set of peripherals are available.

The status of resources in this state are

- UULP-PERIPHERALS and Analog-PERIPHERALS are available and are configured before entering this state.
- 5 GPIO's are available - 5(UULP Vbat GPIO)
- SRAM needs to be retained.

Wakeup sources for these states are defined in the [Wakeup source](#) section below.

7.5.7 PS0

This is a Shutdown state entered from PS4 state through a Software instruction. The CPU is power-gated and a much smaller set of peripherals are available.

The status of resources in this state are

- UULP-PERIPHERALS and Analog-PERIPHERALS are available and are configured before entering this state.
- 5 GPIO's are available - 5(UULP Vbat GPIO)
- SRAM need not be retained.

Wakeup sources for this state are defined in the [Wakeup source](#) section below

7.5.8 Programming Sequence for transitions

PS4 → PS3

The programming sequence for this transitions is described below

1. Configure the LDO SoC 1.1 output voltage to a lower voltage of 1.0V.
 - a. Program corresponding bit in [PMU_LDO_CTRL_REG_CLEAR](#) Register.
2. Configure the DC-DC 1.35 output voltage to a lower voltage of 1.25V
 - a. Program corresponding bit in [PMU_LDO_CTRL_REG_CLEAR](#) Register.

PS3 → PS4

The programming sequence for this transitions is described below

1. Configure the LDO SoC 1.1 output voltage to a lower voltage of 1.1V.
 - a. Program corresponding bit in [PMU_LDO_CTRL_REG_SET](#) Register.
2. Configure the DC-DC 1.35 output voltage to a lower voltage of 1.35V
 - a. Program corresponding bit in [PMU_LDO_CTRL_REG_SET](#) Register.

PS4/PS3 → PS2

This includes the following transitions and the programming sequence is same for all these transitions.

- PS4 → PS2
- PS3 → PS2

The programming sequence for these transitions is described below

1. Switch the Processor clock to MCUHP_REF_CLK as described in [MCU Clocking](#) section.

2. The following settings need to be configured for PS4-PS2 transition. The switching times between PS4->PS2 is determined by the state of the LDO SoC 1.1 and DC-DC 1.35. The OFF delays for these are described in the Power Management section
 - a. Configure the voltage source for the domains listed in [Voltage Domains](#) section above as per the requirement in PS2 state.
 - i. The voltage sources need to be selected as per the CPU performance required.
 - ii. These can be configured through VOLTAGE_SEL_ULP_SRAM, VOLTAGE_SEL_LP_SRAM, VOLTAGE_SEL_LP_SRAM_16KB, VOLTAGE_SEL_PROC and VOLTAGE_SEL_ULP_PERIPH parameters in [FSM_POWER_CTRL_DELAY](#) Register.
 - b. The LDO SoC 1.1 and DC-DC 1.35 state during PS2 state.
 - i. These supplies needs to be configured to ON state during PS2 to achieve lower switching time from PS2 to PS4/PS3 states
 - ii. These can be configured through DCDC_EN and LDoSoC_EN parameters in [FSM_POWER_CTRL_DELAY](#) Register.
3. If HP-SRAM1/HP-SRAM2 needs to be retained in PS2 state, please refer "Retention" section of the MCU SAPI Manual.
4. Configure the ULP AHB Interface clock source and output frequency as described in [ULP Clocking](#) section.
5. Dummy read for flushing the pending transactions of ULP-PERIPHERAL accesses.
6. If the required SRAM is less than 32KB, then a lower current consumption can be achieved by configuring the ULP MODE (Refer to [ULP_MODE_CONFIG](#) Register).
7. Enable functional switching to PS2 state (Refer to [ULP_MODE_CONFIG](#) Register).
8. Poll for the status bit for the functional switching done above (Refer M4_ULP_SLP_STATUS_REG Register described in [NVIC](#) section).
9. Disable the clock used for ULP-PERIPHERAL accesses in PS4 state
 - a. Program MCUULP_BRIDGE_CLK_EN in MCUULP_PROC_CLK_CONFIG Register described in [ULP Clocking](#) section.
10. Enable Isolation for all the interfaces with HIGH-VOLTAGE-DOMAIN which are not operational in PS2 state. (Refer to [ULP_ISOLATION_CTRL](#) Register).
11. Enable physical switching for PS2 state (Refer to [ULP_MODE_CONFIG](#) Register described below).
12. Poll for the status bit for the voltage switching done above (Refer M4_ULP_SLP_STATUS_REG Register described in [NVIC](#) section).

CPU will be operating on MCU-ULP AHB Interface Clock as described in [MCU Clocking](#) section once we switch to PS2 state.

PS2 -> PS4/PS3

This includes the following transitions and the programming sequence is same for all these transitions.

- PS4 -> PS2
- PS3 -> PS2

The programming sequence for these transitions is described below

1. Disable the clock used for ULP-PERIPHERAL accesses in PS4 state
 - a. Program MCUULP_BRIDGE_CLK_EN in MCUULP_PROC_CLK_CONFIG Register described in [ULP Clocking](#) section.
2. The following settings need to be configured for PS2-PS4 transition (Refer [FSM_POWER_CTRL_DELAY](#) Register described below). The switching times from PS2 to PS4 is determined by the state of the LDO SoC 1.1 and DC-DC 1.35 configured when entering the PS2 state. The ON delays for these are described in the Power Management section
 - a. Configure the voltage source for the domains listed in [Voltage Domains](#) section above to LDO-SOC 1.1.

- i. These can be configured through VOLTAGE_SEL_ULP_SRAM, VOLTAGE_SEL_LP_SRAM, VOLTAGE_SEL_LP_SRAM_16KB, VOLTAGE_SEL_PROC and VOLTAGE_SEL_ULP_PERIPH parameters in [FSM_POWER_CTRL_DELAY](#) Register.
- b. The ON times for DC-DC 1.35 and LDO-SoC 1.1 needs to be configured based on their state before entering PS2 state.
 - i. These can be configured to lowest possible value if there are maintained in ON state during PS2.
 - ii. These can be configured through DCDC_ON_TIME and LDoSoC_ON_TIME parameters in [FSM_POWER_CTRL_DELAY](#) Register.
3. Switch the CPU clock to Reference clock as described in [MCU Clocking](#) section.
4. Enable voltage switching for PS4 state (Refer to [ULP_MODE_CONFIG](#) Register).
5. Poll for the status bit for the voltage switching done above (Refer M4_ULP_SLP_STATUS_REG Register described in [NVIC](#) section).
6. If HP-SRAM1/HP-SRAM2 was retained before transition to PS2 state, then the isolation on the HP-SRAM1/HP-SRAM2 needs to be disabled (Refer to [SRAM_Power_Control_REG3_SET](#) Register).
7. Disable Isolation for all the interfaces with HIGH-VOLTAGE Domains. (Refer to [ULP_ISOLATION_CTRL](#) Register).
8. Disable the clock used for ULP-PERIPHERAL accesses in PS4 state
 - a. Program MCUULP_BRIDGE_CLK_EN in MCUULP_PROC_CLK_CONFIG Register described in [ULP Clocking](#) section.
9. Enable functional switching to PS4 state (Refer ULP_MODE_CONFIG Register described below).
10. Poll for the status bit for the functional switching done above (Refer M4_ULP_SLP_STATUS_REG Register described in [NVIC](#) section).

CPU will be operating on Reference Clock as described in [MCU Clocking](#) section once we switch PS3/PS4 state.

PS2 -> PS1

The programming sequence for this transition is described below. One of the ULP-Peripheral or Sensor-Data-Collector needs to be configured for the required activity before initiating the transition to PS1. A wakeup source available in PS1 state as per the [Wakeup sources](#) section below needs to be configured.

1. Configure any of the ULP-Peripheral or Sensor-Data-Collector as wakeup source for transition from PS1 to PS2
 - a. This can be configured through BITS(29:16) in [FSM_SLEEP_CTRLS_AND_WAKEUP_MODE](#) Register described below.
2. Configure the SRAM domains to be retained during PS1 state
 - a. This can be configured through BITS[7:3] in [FSM_SLEEP_CTRLS_AND_WAKEUP_MODE](#) Register described below.
3. Enter Sleep state. Please refer "SLEEP" Section in MCU SAPI Manual.

ACTIVE -> SLEEP

This includes the following transitions and the programming sequence is same for all these transitions. The state of the LDO SoC 1.1, LDO FL 1.8 and DC-DC 1.35 during SLEEP state can be configured for achieving lower wakeup time from Sleep to Active state. The Power Domains will remain in the same state (Power-Up/Power-Down) upon Wakeup from Sleep.

- PS4 -> PS4-SLEEP (Switches to PS4 upon wakeup)
- PS3 -> PS3-SLEEP (Switches to PS3 upon wakeup)
- PS2 -> PS2-SLEEP (Switches to PS2 upon wakeup)

The programming sequence for these transitions is described below

1. Configure the state of LDO SoC 1.1, LDO FL 1.8 and DC-DC 1.35 during Sleep state

- a. This can be configured through BITS[11:8] in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
2. Configure the SRAM domains to be retained during Sleep state
 - a. This can be configured through BITS[7:3] in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
3. Configure the wakeup source for transition from SLEEP state (PS4-SLEEP, PS3-SLEEP, PS2-SLEEP) to ACTIVE state (PS4, PS3, PS2)
 - a. This can be configured through BITS[29:16] in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
4. Configure the ON times for DC-DC 1.35 and LDO SoC 1.1 if they are configured to OFF state during Sleep.
 - a. This can be configured through PMU_POWERGOOD_TIME in [FSM_ONTIME_CONFIG_REG](#) Register described below.
5. Configure the ON time for HF-Crystal clock
 - a. This can be configured through HF_CRYSTAL_SETTLING_TIME in [FSM_ONTIME_CONFIG_REG](#) Register described below.
6. The ON time for HF-Crystal clock can be skipped to reduce the wakeup time if the MCUHP_REF_CLK is not configured to XTAL_CLK before entering Sleep state as described in Clock Architecture section.
 - a. This can be configured through SKIP_XTAL_WAIT_TIME in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
7. Enter Sleep state. Please refer "SLEEP" Section in MCU SAPI Manual.

PS4/PS3 -> PS0

This includes the following transitions and the programming sequence is same for all these transitions. The state of the LDO SoC 1.1, LDO FL 1.8 and DC-DC 1.35 during SLEEP state can be configured for achieving lower wakeup time from Sleep to Active state. The Power Domains will remain in the same state (Power-Up/Power-Down) upon Wake-up from Sleep.

- PS4 -> PS0 (Switches to PS4 upon wakeup)
- PS3 -> PS0 (Switches to PS3 upon wakeup)

The programming sequence for this transition is described below

1. Reset the state of LDO SoC 1.1, LDO FL 1.8 and DC-DC 1.35 during Sleep state
 - a. This can be configured through BITS[11:8] in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
2. Reset the SRAM domains to be retained during Sleep state
 - a. This can be configured through BITS[7:3] in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
3. Configure the wakeup source for transition from SLEEP state (PS4-SLEEP, PS3-SLEEP, PS2-SLEEP) to ACTIVE state (PS4, PS3, PS2)
 - a. This can be configured through BITS[29:16] in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
4. Configure the ON times for DC-DC 1.35 and LDO SoC 1.1 if they are configured to OFF state during Sleep.
 - a. This can be configured through PMU_POWERGOOD_TIME in [FSM_ONTIME_CONFIG_REG](#) Register described below.
5. Configure the ON time for HF-Crystal clock
 - a. This can be configured through HF_CRYSTAL_SETTLING_TIME in [FSM_ONTIME_CONFIG_REG](#) Register described below.
6. The ON time for HF-Crystal clock can be skipped to reduce the wakeup time if the MCUHP_REF_CLK is not configured to XTAL_CLK before entering Sleep state as described in Clock Architecture section.
 - a. This can be configured through SKIP_XTAL_WAIT_TIME in [FSM_SLEEP_CTRL_AND_WAKEUP_MODE](#) Register described below.
7. Enter Sleep state. Please refer "SLEEP" Section in MCU SAPI Manual.

PS2 -> PS0

The Power Domains will be switched to reset state upon Wakeup from Sleep.

The programming sequence for this transition is described below

1. Reset the SRAM Voltage domains retention state
 - a. This can be configured through BITS[7:3] in [FSM_SLEEP_CTRLS_AND_WAKEUP_MODE](#) Register described below.
2. Configure the wakeup source for transition from PS0 to PS4/PS3
 - a. This can be configured through BITS[29:16] in [FSM_SLEEP_CTRLS_AND_WAKEUP_MODE](#) Register described below.
3. Enable Reset of Power Domain Control Battery FFs
 - a. This can be configured through RESET_BFF_EN in [FSM_CTRL_POWER_DOMAINS](#) Register described below.
4. Enter Sleep state. Please refer "SLEEP" Section in MCU SAPI Manual.

7.6 Memory Retention in Sleep / Shutdown states

The table below indicates the SRAM banks and Backup Register Array which can be retained in each Sleep/Shutdown state.

| S.No | Power State | HP-SRAM (256KB) | LP-SRAM (128KB) | ULP-SRAM (16KB) | Backup Register Array (32 bytes) |
|------|-------------|--------------------|--------------------|--------------------|-------------------------------------|
| 1 | PS4-SLEEP | Yes | Yes | Yes | Yes |
| 2 | PS3-SLEEP | Yes | Yes | Yes | Yes |
| 3 | PS2-SLEEP | Yes | Yes | Yes | Yes |
| 4 | PS0 | No | No | No | Yes |

206 . SRAM in different states

7.7 Wakeup Sources

The table below indicates the wakeup sources available in Standby/Sleep/Shutdown states.

| S. No | Wakeup Source | PS2- STANDBY | PS4-SLEEP | PS3-SLEEP | PS2-SLEEP | PS1 | PS0 |
|----------|--|-----------------|-----------|-----------|-----------|-----|-----|
| 1 | UULP Vbat GPIO | Yes | Yes | Yes | Yes | Yes | Yes |
| 2 | Watch-Dog Interrupt | Yes | Yes | Yes | Yes | Yes | Yes |
| 3 | Analog Comparator | Yes | Yes | Yes | Yes | Yes | Yes |
| 4 | Wake-Fi Rx | Yes | Yes | Yes | Yes | Yes | Yes |
| 5 | BOD | Yes | Yes | Yes | Yes | Yes | Yes |
| 6 | ULP-Peripheral SDC | Yes | No | No | No | Yes | No |
| 7 | Wireless Processor Interrupt | Yes | Yes | Yes | Yes | Yes | Yes |
| 8 | Deep-Sleep Timer Interrupt | Yes | Yes | Yes | Yes | Yes | Yes |
| 9 | Alarm Interrupt | Yes | Yes | Yes | Yes | Yes | Yes |
| 10 | Second Based Interrupt | Yes | Yes | Yes | Yes | Yes | Yes |
| 11 | Milli-Second Based Interrupt | Yes | Yes | Yes | Yes | Yes | Yes |
| 12 | ULP-Peripheral GPIO Group Interrupt | Yes | No | No | No | Yes | No |
| 13 | ULP-Peripheral GPIO Pin Interrupt | Yes | No | No | No | Yes | No |
| 14 | ULP-Peripheral FIM Interrupt | Yes | No | No | No | No | No |

| S. No | Wakeup Source | PS2- STANDBY | PS4-SLEEP | PS3-SLEEP | PS2-SLEEP | PS1 | PS0 |
|----------|---|-----------------|-----------|-----------|-----------|-----|-----|
| 15 | ULP-Peripheral SPI/SSI Master Interrupt | Yes | No | No | No | Yes | No |
| 16 | ULP-Peripheral IR Interrupt | Yes | No | No | No | Yes | No |
| 17 | ULP-Peripheral I2S Interrupt | Yes | No | No | No | Yes | No |
| 18 | ULP-Peripheral I2C Interrupt | Yes | No | No | No | Yes | No |
| 19 | ULP-Peripheral UART Interrupt | Yes | No | No | No | Yes | No |
| 20 | ULP-Peripheral ADC/DAC Interrupt | Yes | No | No | No | Yes | No |
| 21 | ULP-Peripheral DMA Interrupt | Yes | No | No | No | Yes | No |
| 22 | ULP-Peripheral GPIO Wakeup Interrupt | Yes | No | No | No | Yes | No |
| 23 | ULP-Peripheral Touch Sensor Interrupt | Yes | No | No | No | Yes | No |
| 24 | ULP-Peripheral Timer Interrupt | Yes | No | No | No | Yes | No |
| 25 | ULP-Peripheral VAD Interrupt | Yes | No | No | No | Yes | No |

207 . List of Wakeup Sources in different states

7.8 Register Summary

7.8.1 High-Performance Power Domains

This includes the APPLICATIONS, HIGH-SPEED-INTERFACES, HP-PERIPHERALS, HIGH-FREQ-PLL, DDR-FLASH-DLL, HP-SRAM and LP-SRAM.

Base Address: 0x2404_8400

| Register Name | Offset | Description |
|-------------------------------|--------|--|
| Power_Control_SET | 0x08 | Enables power for APPLICATIONS, HIGH SPEED INTERFACES, HP-PERIPHERALS domains |
| Power_Control_CLEAR | 0x0C | Disables power for APPLICATIONS, HIGH SPEED INTERFACES, HP-PERIPHERALS domains |
| SRAM_Power_Control_REG1_SET | 0x10 | Enables power for HP-SRAM1, HP-SRAM2 and LP-SRAM domains |
| SRAM_Power_Control_REG1_CLEAR | 0x14 | Disables power for HP-SRAM, HP-SRAM2 and LP-SRAM domains |
| SRAM_Power_Control_REG2_SET | 0x18 | Enables power for HP-SRAM, HP-SRAM2 and LP-SRAM domains |
| SRAM_Power_Control_REG2_CLEAR | 0x1C | Disables power for HP-SRAM, HP-SRAM2 and LP-SRAM domains |
| SRAM_Power_Control_REG3_SET | 0x20 | Disables isolation on HP-SRAM, HP-SRAM2 and LP-SRAM input signals |
| SRAM_Power_Control_REG3_CLEAR | 0x24 | Enables isolation on HP-SRAM, HP-SRAM2 and LP-SRAM input signals |
| SRAM_Power_Control_REG4_SET | 0x28 | Enables Deep-Sleep for HP-SRAM, HP-SRAM2 and LP-SRAM domains |
| SRAM_Power_Control_REG4_CLEAR | 0x2C | Disables Deep-Sleep for HP-SRAM, HP-SRAM2 and LP-SRAM domains |

| Register Name | Offset | Description |
|----------------------------|--------|--|
| PMU_LDO_CTRL_REG_SET | 0x68 | Control Register for PMU Supply Voltages |
| PMU_LDO_CTRL_REG_CLEA R | 0x6C | Control Register for PMU Supply Voltages |
| PLL_Power_Control | 0x80 | Controls power for HIGH-FREQ-PLL domains |
| DLL_Power_Control | 0x84 | Controls power for DDR-FLASH-DLL domains |

208 . High-Power Domains Control Registers

7.8.2 Low-Power Domains

This includes the ULP-PERIPHERALS and ULP-SRAM.

Base Address: 0x2404_8400

| Register Name | Offset | Description |
|------------------------------------|--------|--|
| ULP_Peripheral_Power_Control_SET | 0x44 | Enables power for ULP-PERIPHERALS domains |
| ULP_Peripheral_Power_Control_CLEAR | 0x48 | Disables power for ULP-PERIPHERALS domains |
| ULP_SRAM_Power_Control_REG1_SET | 0x4C | Enables power for ULP-SRAM domains |
| ULP_SRAM_Power_Control_REG1_CLEAR | 0x50 | Disables power for ULP-SRAM domains |
| ULP_SRAM_Power_Control_REG4_SET | 0x54 | Enables power for ULP-SRAM domains |
| ULP_SRAM_Power_Control_REG4_CLEAR | 0x58 | Disables power for ULP-SRAM domains |
| ULP_SRAM_Power_Control_REG2_SET | 0x5C | Enables Deep-Sleep for ULP-SRAM domains |
| ULP_SRAM_Power_Control_REG2_CLEAR | 0x60 | Disables Deep-Sleep for ULP-SRAM domains |

209 . Low-Power Domains Control Registers

7.8.3 Ultra Low-Power Domains

This includes UULP-PERIPHERALS.

Base Address: 0x2404_8000

| Register Name | Offset | Description |
|-------------------------------------|--------|---|
| UULP_Peripheral_Power_Control_SET | 0x00 | Enables power for UULP-PERIPHERALS domains |
| UULP_Peripheral_Power_Control_CLEAR | 0x04 | Disables power for UULP-PERIPHERALS domains |

210 . Ultra Low-Power Domains Control Registers

7.8.4 Analog Domains

This includes ANALOG-PERIPHERALS.

Base Address: 0x2404_xxxx

| Register Name | Offset | Description |
|----------------------------|--------|---|
| Analog_Power_Control_SET | 0x00 | Enables power for Analog Power domains |
| Analog_Power_Control_CLEAR | 0x04 | Disables power for Analog Power domains |

211 . Analog Power Domains Control Registers

7.8.5 SLEEP FSM

Base Address: 0x2404_8100

| Register Name | Offset | Description |
|---------------------------------|--------|---|
| FSM_SLEEP_CTRLS_AND_WAKEUP_MODE | 0x00 | Sleep Control Signals and Wakeup source selection |
| ULP_MODE_CONFIG | 0x04 | Configuration for Ultra Low-Power Mode of the processor (PS2 State) |
| FSM_ONTIME_CONFIG_REG | 0x10 | Configuration Register for PMU and HF-Crystal ON time |
| FSM_POWER_CTRL_DELAY | 0x14 | Power Control and Delay Configuration for Ultra Low-Power Mode of the processor (PS2 State) |
| FSM_CTRL_POWER_DOMAINS | 0x24 | Power Domains Controlled by Sleep FSM |

212 . SLEEP FSM Registers

7.8.6 ULP Configuration

Base Address: 0x2404_8400

| Register Name | Offset | Description |
|--------------------|--------|---|
| ULP_ISOLATION_CTRL | 0x3C | Isolation Configuration for Ultra Low-Power Mode of the processor (PS2 State) |

213 . ULP Configuration Register

7.9 Register Description

7.9.1 Power_Control_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|--|
| 31:23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | RW | PWRCTRL_ROM | 1 | Writing 1 to this enables power to the ROM. Writing 0 to this has no effect. |
| 21:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | - | Reserved | - | It is recommended to write these bits to 0. |
| 17 | RW | PWRCTRL_DEBUG | 1 | Writing 1 to this enables power to the DEBUG. Writing 0 to this has no effect. |
| 16 | RW | PWRCTRL_FPU | 1 | Writing 1 to this enables power to the FPU. Writing 0 to this has no effect. |
| 15 | RW | PWRCTRL_ICACHE | 1 | Writing 1 to this enables power to the ICACHE. Writing 0 to this has no effect. |
| 14 | - | Reserved | - | It is recommended to write these bits to 0. |
| 13 | RW | PWRCTRL_QSPI | 1 | Writing 1 to this enables power to the QSPI. Writing 0 to this has no effect. |
| 12 | RW | PWRCTRL_ETHERNET | 1 | Writing 1 to this enables power to the ETHERNET. Writing 0 to this has no effect. |
| 11 | RW | PWRCTRL_SDIO_SPI | 1 | Writing 1 to this enables power to the SDIO-SPI Slave. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------------|-------------|--|
| 10 | RW | PWRCTRL_USB | 1 | Writing 1 to this enables power to the USB. Writing 0 to this has no effect. |
| 9 | RW | PWRCTRL_DMA | 1 | Writing 1 to this enables power to the DMA. Writing 0 to this has no effect. |
| 8 | RW | PWRCTRL_PERI_DOM AIN1 | 1 | Writing 1 to this enables power to the PERI-DOMAIN1. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL_PERI_DOM AIN2 | 1 | Writing 1 to this enables power to the PERI-DOMAIN2. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL_PERI_DOM AIN3 | 1 | Writing 1 to this enables power to the PERI-DOMAIN3. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL_CCI | 1 | Writing 1 to this enables power to the CCI. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL_EFUSE | 1 | Writing 1 to this enables power to the EFUSE. Writing 0 to this has no effect. |
| 3 | - | Reserved | - | It is recommended to write these bits to 0. |
| 2 | RW | PWRCTRL_SDMEM | 1 | Writing 1 to this enables power to the SDMEM. Writing 0 to this has no effect. |
| 1:0 | - | Reserved | - | It is recommended to write these bits to 0. |

214 . Power Control SET Register

7.9.2 Power_Control_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|---|
| 31:23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | RW | PWRCTRL_ROM | 1 | Writing 1 to this disables power to the ROM. Writing 0 to this has no effect. |
| 21:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | - | Reserved | - | It is recommended to write these bits to 0. |
| 17 | RW | PWRCTRL_DEBUG | 1 | Writing 1 to this disables power to the DEBUG. Writing 0 to this has no effect. |
| 16 | RW | PWRCTRL_FPU | 1 | Writing 1 to this disables power to the FPU. Writing 0 to this has no effect. |
| 15 | RW | PWRCTRL_ICACHE | 1 | Writing 1 to this disables power to the ICACHE. Writing 0 to this has no effect. |
| 14 | - | Reserved | - | It is recommended to write these bits to 0. |
| 13 | RW | PWRCTRL_QSPI | 1 | Writing 1 to this disables power to the QSPI. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------------|-------------|---|
| 12 | RW | PWRCTRL_ETHERNET | 1 | Writing 1 to this disables power to the ETHERNET. Writing 0 to this has no effect. |
| 11 | RW | PWRCTRL_SDIO_SPI | 1 | Writing 1 to this disables power to the SDIO-SPI Slave. Writing 0 to this has no effect. |
| 10 | RW | PWRCTRL_USB | 1 | Writing 1 to this disables power to the USB. Writing 0 to this has no effect. |
| 9 | RW | PWRCTRL_DMA | 1 | Writing 1 to this disables power to the DMA. Writing 0 to this has no effect. |
| 8 | RW | PWRCTRL_PERI_DOM_AIN1 | 1 | Writing 1 to this disables power to the PERI-DOMAIN1. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL_PERI_DOM_AIN2 | 1 | Writing 1 to this disables power to the PERI-DOMAIN2. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL_PERI_DOM_AIN3 | 1 | Writing 1 to this disables power to the PERI-DOMAIN3. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL_CCI | 1 | Writing 1 to this disables power to the CCI. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL_EFUSE | 1 | Writing 1 to this disables power to the EFUSE. Writing 0 to this has no effect. |
| 3 | - | Reserved | - | It is recommended to write these bits to 0. |
| 2 | RW | PWRCTRL_SDMEM | 1 | Writing 1 to this disables power to the SDMEM. Writing 0 to this has no effect. |
| 1:0 | - | Reserved | - | It is recommended to write these bits to 0. |

215 .Power Control CLEAR Register

7.9.3 PLL_Power_Control

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------------|-------------|---|
| 31:8 | - | Reserved | - | It is recommended to write these bits to 0. |
| 7 | RW | SOCPLL_VDD13_ISO_EN | 1 | This is used for isolation of signals from DC-DC 1.35 to the VBATT supply in SoC-PLL to avoid leakage. Writing 1 to this enables isolation. Writing 0 to this disables isolation. |
| 6 | RW | PWRCTRL_PLL_REG | 0 | Writing 0 to this enables power to the PLL Programming Registers. Writing 1 to this disables power to the PLL Programming Registers. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|---|
| 5:0 | - | Reserved | - | It is recommended to write these bits to 0. |

216 .PLL Power Control Register

7.9.4 DLL_Power_Control

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------|-------------|---|
| 31:7 | - | Reserved | - | It is recommended to write these bits to 0. |
| 6 | RW | PWRCTRL_DLL_TX | 1 | Writing 1 to this enables power to the Tx Section of DLL Writing 0 to this disables power to the Tx Section of DLL |
| 5:3 | - | Reserved | - | It is not recommended to overwrite the content in this bit. |
| 2 | RW | PWRCTRL_DLL_RX | 1 | Writing 1 to this enables power to the Rx Section of DLL Writing 0 to this disables power to the Rx Section of DLL |
| 1:0 | - | Reserved | - | It is recommended to write these bits to 0. |

217 .DLL Power Control Register

7.9.5 SRAM_Power_Control_REG1_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|---|
| 31:20 | - | Reserved | - | It is recommended to write these bits to 0. |
| 19 | RW | PWRCTRL1_HPSRAM2_4 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-4. Writing 0 to this has no effect. |
| 18 | RW | PWRCTRL1_HPSRAM2_3 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-3. Writing 0 to this has no effect. |
| 17 | RW | PWRCTRL1_HPSRAM2_2 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-2. Writing 0 to this has no effect. |
| 16 | RW | PWRCTRL1_HPSRAM2_1 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-1. Writing 0 to this has no effect. |
| 15:10 | - | Reserved | - | It is recommended to write these bits to 0. |
| 9 | RW | PWRCTRL1_LPSRAM_7 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-7. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------|-------------|---|
| 8 | RW | PWRCTRL1_LPSRAM_6 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-6. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL1_LPSRAM_5 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-5. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL1_LPSRAM_4 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-4. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL1_LPSRAM_3 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-3. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL1_LPSRAM_2 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-2. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL1_LPSRAM_1 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-1. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL1_HPSRAM1_3 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM1-3. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL1_HPSRAM1_2 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM1-2. Writing 0 to this has no effect. |
| 0 | RW | PWRCTRL1_HPSRAM1_1 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM1-1. Writing 0 to this has no effect. |

218 . SRAM Power Control Register1 SET

7.9.6 SRAM_Power_Control_REG1_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:20 | - | Reserved | - | It is recommended to write these bits to 0. |
| 19 | RW | PWRCTRL1_HPSRAM2_4 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-4. Writing 0 to this has no effect. |
| 18 | RW | PWRCTRL1_HPSRAM2_3 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-3. Writing 0 to this has no effect. |
| 17 | RW | PWRCTRL1_HPSRAM2_2 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-2. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|--------------------|--------------------|--|
| 16 | RW | PWRCTRL1_HPSRAM2_1 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-1. Writing 0 to this has no effect. |
| 15:10 | - | Reserved | - | It is recommended to write these bits to 0. |
| 9 | RW | PWRCTRL1_LPSRAM_7 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-7. Writing 0 to this has no effect. |
| 8 | RW | PWRCTRL1_LPSRAM_6 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-6. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL1_LPSRAM_5 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-5. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL1_LPSRAM_4 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-4. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL1_LPSRAM_3 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-3. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL1_LPSRAM_2 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-2. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL1_LPSRAM_1 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-1. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL1_HPSRAM1_3 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM1-3. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL1_HPSRAM1_2 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM1-2. Writing 0 to this has no effect. |
| 0 | RW | PWRCTRL1_HPSRAM1_1 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM1-1. Writing 0 to this has no effect. |

219 . SRAM Power Control Register1 CLEAR

7.9.7 SRAM_Power_Control_REG2_SET

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 31:20 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|---|
| 19 | RW | PWRCTRL2_HPSRAM2_4 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-4. Writing 0 to this has no effect. |
| 18 | RW | PWRCTRL2_HPSRAM2_3 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-3. Writing 0 to this has no effect. |
| 17 | RW | PWRCTRL2_HPSRAM2_2 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-2. Writing 0 to this has no effect. |
| 16 | RW | PWRCTRL2_HPSRAM2_1 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM2-1. Writing 0 to this has no effect. |
| 15:10 | - | Reserved | - | It is recommended to write these bits to 0. |
| 9 | RW | PWRCTRL2_LPSRAM_7 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-7. Writing 0 to this has no effect. |
| 8 | RW | PWRCTRL2_LPSRAM_6 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-6. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL2_LPSRAM_5 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-5. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL2_LPSRAM_4 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-4. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL2_LPSRAM_3 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-3. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL2_LPSRAM_2 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-2. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL2_LPSRAM_1 | 1 | Writing 1 to this enables power to all Banks in LP-SRAM-1. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL2_HPSRAM1_3 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM1-3. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL2_HPSRAM1_2 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM1-2. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------|-------------|---|
| 0 | RW | PWRCTRL2_HPSRAM1_1 | 1 | Writing 1 to this enables power to all Banks in HP-SRAM1-1. Writing 0 to this has no effect. |

220 . SRAM Power Control Register2 SET

7.9.8 SRAM_Power_Control_REG2_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:20 | - | Reserved | - | It is recommended to write these bits to 0. |
| 19 | RW | PWRCTRL2_HPSRAM2_4 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-4. Writing 0 to this has no effect. |
| 18 | RW | PWRCTRL2_HPSRAM2_3 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-3. Writing 0 to this has no effect. |
| 17 | RW | PWRCTRL2_HPSRAM2_2 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-2. Writing 0 to this has no effect. |
| 16 | RW | PWRCTRL2_HPSRAM2_1 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM2-1. Writing 0 to this has no effect. |
| 15:10 | - | Reserved | - | It is recommended to write these bits to 0. |
| 9 | RW | PWRCTRL2_LPSRAM_7 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-7. Writing 0 to this has no effect. |
| 8 | RW | PWRCTRL2_LPSRAM_6 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-6. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL2_LPSRAM_5 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-5. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL2_LPSRAM_4 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-4. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL2_LPSRAM_3 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-3. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL2_LPSRAM_2 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-2. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------|-------------|--|
| 3 | RW | PWRCTRL2_LPSRAM_1 | 1 | Writing 1 to this disables power to all Banks in LP-SRAM-1. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL2_HPSRAM1_3 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM1-3. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL2_HPSRAM1_2 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM1-2. Writing 0 to this has no effect. |
| 0 | RW | PWRCTRL2_HPSRAM1_1 | 1 | Writing 1 to this disables power to all Banks in HP-SRAM1-1. Writing 0 to this has no effect. |

221 . SRAM Power Control Register2 CLEAR

7.9.9 SRAM_Power_Control_REG3_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|--|
| 31:17 | - | Reserved | - | It is recommended to write these bits to 0. |
| 16 | RW | INP_ISO_HP_SRAM2 | 1 | Writing 1 to this disables isolation on inputs for HP-SRAM2 Banks. Writing 0 to this has no effect. |
| 15:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | RW | INP_ISO_LP_SRAM | 1 | Writing 1 to this disables isolation on inputs for LP-SRAM Banks. Writing 0 to this has no effect. |
| 2:1 | - | Reserved | - | It is recommended to write these bits to 0. |
| 0 | RW | INP_ISO_HP_SRAM1 | 1 | Writing 1 to this disables isolation on inputs for HP-SRAM1 Banks. Writing 0 to this has no effect. |

222 . SRAM Power Control Register3 SET

7.9.10 SRAM_Power_Control_REG3_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|---|
| 31:17 | - | Reserved | - | It is recommended to write these bits to 0. |
| 16 | RW | INP_ISO_HP_SRAM2 | 1 | Writing 1 to this enables isolation on inputs for HP-SRAM2 Banks. Writing 0 to this has no effect. |
| 15:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | RW | INP_ISO_LP_SRAM | 1 | Writing 1 to this enables isolation on inputs for LP-SRAM Banks. Writing 0 to this has no effect. |
| 2:1 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------|-------------|--|
| 0 | RW | INP_ISO_HP_SRAM1 | 1 | <p>Writing 1 to this enables isolation on inputs for HP-SRAM1 Banks.</p> <p>Writing 0 to this has no effect.</p> |

223 . SRAM Power Control Register3 CLEAR

7.9.11 SRAM_Power_Control_REG4_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|--|
| 31:28 | - | Reserved | - | It is recommended to write these bits to 0. |
| 27 | RW | DS_HPSRAM2_4_B4 | 1 | <p>Writing 1 to this enables deep sleep mode to 4th Bank in HP-SRAM2-4.</p> <p>Writing 0 to this has no effect.</p> |
| 26 | RW | DS_HPSRAM2_4_B3 | 1 | <p>Writing 1 to this enables deep sleep mode to 3rd Bank in HP-SRAM2-4.</p> <p>Writing 0 to this has no effect.</p> |
| 25 | RW | DS_HPSRAM2_4_B2 | 1 | <p>Writing 1 to this enables deep sleep mode to 2nd Bank in HP-SRAM2-4.</p> <p>Writing 0 to this has no effect.</p> |
| 24 | RW | DS_HPSRAM2_4_B1 | 1 | <p>Writing 1 to this enables deep sleep mode to 1st Bank in HP-SRAM2-4.</p> <p>Writing 0 to this has no effect.</p> |
| 23 | RW | DS_HPSRAM2_3_B5 | 1 | <p>Writing 1 to this enables deep sleep mode to 5th Bank in HP-SRAM2-3.</p> <p>Writing 0 to this has no effect.</p> |
| 22 | RW | DS_HPSRAM2_3_B4 | 1 | <p>Writing 1 to this enables deep sleep mode to 4th Bank in HP-SRAM2-3.</p> <p>Writing 0 to this has no effect.</p> |
| 21 | RW | DS_HPSRAM2_3_B3 | 1 | <p>Writing 1 to this enables deep sleep mode to 3rd Bank in HP-SRAM2-3.</p> <p>Writing 0 to this has no effect.</p> |
| 20 | RW | DS_HPSRAM2_3_B2 | 1 | <p>Writing 1 to this enables deep sleep mode to 2nd Bank in HP-SRAM2-3.</p> <p>Writing 0 to this has no effect.</p> |
| 19 | RW | DS_HPSRAM2_3_B1 | 1 | <p>Writing 1 to this enables deep sleep mode to 1st Bank in HP-SRAM2-3.</p> <p>Writing 0 to this has no effect.</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|--|
| 18 | RW | DS_HPSRAM2_2_B2 | 1 | Writing 1 to this enables deep sleep mode to 2 nd Bank in HP-SRAM2-2. Writing 0 to this has no effect. |
| 17 | RW | DS_HPSRAM2_2_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in HP-SRAM2-2. Writing 0 to this has no effect. |
| 16 | RW | DS_HPSRAM2_1_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in HP-SRAM2-1. Writing 0 to this has no effect. |
| 15 | - | Reserved | - | It is recommended to write these bits to 0. |
| 14 | RW | DS_LPSRAM_7_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in LP-SRAM-7. Writing 0 to this has no effect. |
| 13 | RW | DS_LPSRAM_6_B4 | 1 | Writing 1 to this enables deep sleep mode to 4 th Bank in LP-SRAM-6. Writing 0 to this has no effect. |
| 12 | RW | DS_LPSRAM_6_B3 | 1 | Writing 1 to this enables deep sleep mode to 3 rd Bank in LP-SRAM-6. Writing 0 to this has no effect. |
| 11 | RW | DS_LPSRAM_6_B2 | 1 | Writing 1 to this enables deep sleep mode to 2 nd Bank in LP-SRAM-6. Writing 0 to this has no effect. |
| 10 | RW | DS_LPSRAM_6_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in LP-SRAM-6. Writing 0 to this has no effect. |
| 9 | RW | DS_LPSRAM_5_B2 | 1 | Writing 1 to this enables deep sleep mode to 2 nd Bank in LP-SRAM-5. Writing 0 to this has no effect. |
| 8 | RW | DS_LPSRAM_5_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in LP-SRAM-5. Writing 0 to this has no effect. |
| 7 | RW | DS_LPSRAM_4_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in LP-SRAM-4. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|--|
| 6 | RW | DS_LPSRAM_3_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in LP-SRAM-3. Writing 0 to this has no effect. |
| 5 | RW | DS_LPSRAM_2_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in LP-SRAM-2. Writing 0 to this has no effect. |
| 4 | RW | DS_LPSRAM_1_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in LP-SRAM-1. Writing 0 to this has no effect. |
| 3 | RW | DS_HPSRAM1_3_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in HP-SRAM1-3. Writing 0 to this has no effect. |
| 2 | RW | DS_HPSRAM1_2_B2 | 1 | Writing 1 to this enables deep sleep mode to 2 nd Bank in HP-SRAM1-2. Writing 0 to this has no effect. |
| 1 | RW | DS_HPSRAM1_2_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in HP-SRAM1-2. Writing 0 to this has no effect. |
| 0 | RW | DS_HPSRAM1_1_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in HP-SRAM1-1. Writing 0 to this has no effect. |

224 . SRAM Power Control Register4 SET

7.9.12 SRAM_Power_Control_REG4_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|---|
| 31:28 | - | Reserved | - | It is recommended to write these bits to 0. |
| 27 | RW | DS_HPSRAM2_4_B4 | 1 | Writing 1 to this disables deep sleep mode to 4 th Bank in HP-SRAM2-4. Writing 0 to this has no effect. |
| 26 | RW | DS_HPSRAM2_4_B3 | 1 | Writing 1 to this disables deep sleep mode to 3 rd Bank in HP-SRAM2-4. Writing 0 to this has no effect. |
| 25 | RW | DS_HPSRAM2_4_B2 | 1 | Writing 1 to this disables deep sleep mode to 2 nd Bank in HP-SRAM2-4. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|---|
| 24 | RW | DS_HPSRAM2_4_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in HP-SRAM2-4. Writing 0 to this has no effect. |
| 23 | RW | DS_HPSRAM2_3_B5 | 1 | Writing 1 to this disables deep sleep mode to 5 th Bank in HP-SRAM2-3. Writing 0 to this has no effect. |
| 22 | RW | DS_HPSRAM2_3_B4 | 1 | Writing 1 to this disables deep sleep mode to 4 th Bank in HP-SRAM2-3. Writing 0 to this has no effect. |
| 21 | RW | DS_HPSRAM2_3_B3 | 1 | Writing 1 to this disables deep sleep mode to 3 rd Bank in HP-SRAM2-3. Writing 0 to this has no effect. |
| 20 | RW | DS_HPSRAM2_3_B2 | 1 | Writing 1 to this disables deep sleep mode to 2 nd Bank in HP-SRAM2-3. Writing 0 to this has no effect. |
| 19 | RW | DS_HPSRAM2_3_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in HP-SRAM2-3. Writing 0 to this has no effect. |
| 18 | RW | DS_HPSRAM2_2_B2 | 1 | Writing 1 to this disables deep sleep mode to 2 nd Bank in HP-SRAM2-2. Writing 0 to this has no effect. |
| 17 | RW | DS_HPSRAM2_2_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in HP-SRAM2-2. Writing 0 to this has no effect. |
| 16 | RW | DS_HPSRAM2_1_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in HP-SRAM2-1. Writing 0 to this has no effect. |
| 15 | - | Reserved | - | It is recommended to write these bits to 0. |
| 14 | RW | DS_LPSRAM_7_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in LP-SRAM-7. Writing 0 to this has no effect. |
| 13 | RW | DS_LPSRAM_6_B4 | 1 | Writing 1 to this disables deep sleep mode to 4 th Bank in LP-SRAM-6. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|---|
| 12 | RW | DS_LPSRAM_6_B3 | 1 | Writing 1 to this disables deep sleep mode to 3 rd Bank in LP-SRAM-6. Writing 0 to this has no effect. |
| 11 | RW | DS_LPSRAM_6_B2 | 1 | Writing 1 to this disables deep sleep mode to 2 nd Bank in LP-SRAM-6. Writing 0 to this has no effect. |
| 10 | RW | DS_LPSRAM_6_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in LP-SRAM-6. Writing 0 to this has no effect. |
| 9 | RW | DS_LPSRAM_5_B2 | 1 | Writing 1 to this disables deep sleep mode to 2 nd Bank in LP-SRAM-5. Writing 0 to this has no effect. |
| 8 | RW | DS_LPSRAM_5_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in LP-SRAM-5. Writing 0 to this has no effect. |
| 7 | RW | DS_LPSRAM_4_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in LP-SRAM-4. Writing 0 to this has no effect. |
| 6 | RW | DS_LPSRAM_3_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in LP-SRAM-3. Writing 0 to this has no effect. |
| 5 | RW | DS_LPSRAM_2_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in LP-SRAM-2. Writing 0 to this has no effect. |
| 4 | RW | DS_LPSRAM_1_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in LP-SRAM-1. Writing 0 to this has no effect. |
| 3 | RW | DS_HPSRAM1_3_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in HP-SRAM1-3. Writing 0 to this has no effect. |
| 2 | RW | DS_HPSRAM1_2_B2 | 1 | Writing 1 to this disables deep sleep mode to 2 nd Bank in HP-SRAM1-2. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|---|
| 1 | RW | DS_HPSRAM1_2_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in HP-SRAM1-2. Writing 0 to this has no effect. |
| 0 | RW | DS_HPSRAM1_1_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in HP-SRAM1-1. Writing 0 to this has no effect. |

225 . SRAM Power Control Register4 CLEAR

7.9.13 PMU_LDO_CTRL_REG_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|--|
| 31:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | RW | DCDC_1P35_LVL | 1 | Writing 1 to this configures DC-DC 1.35 Supply voltage to 1.35V. Writing 0 to this has no effect. |
| 17 | RW | LDOSOC_1P1_LVL | 1 | Writing 1 to this configures LDO-SoC 1.1 Supply voltage to 1.1V. Writing 0 to this has no effect. |
| 16:1 | - | Reserved | - | It is recommended to write these bits to 0. |
| 0 | RW | FLASH_LDO_EN | 1 | Writing 1 to this enables LDO-FL 1.8 Output voltage. Writing 0 to this has no effect. |

226 . PMU LDO Control SET Register

7.9.14 PMU_LDO_CTRL_REG_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|--|
| 31:19 | - | Reserved | - | It is recommended to write these bits to 0. |
| 18 | RW | DCDC_1P35_LVL | 1 | Writing 1 to this configures DC-DC 1.35 Supply voltage to 1.25V. Writing 0 to this has no effect. |
| 17 | RW | LDOSOC_1P1_LVL | 1 | Writing 1 to this configures LDO-SoC 1.1 Supply voltage to 1.0V. Writing 0 to this has no effect. |
| 16:1 | - | Reserved | - | It is recommended to write these bits to 0. |
| 0 | RW | FLASH_LDO_EN | 1 | Writing 1 to this disables LDO-FL 1.8 Output voltage. Writing 0 to this has no effect. |

227 . PMU LDO Control CLEAR Register

7.9.15 ULP_Peripheral_Power_Control_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|---|
| 31:29 | - | Reserved | - | It is recommended to write these bits to 0. |
| 28 | RW | PWRCTRL_FIM | 1 | Writing 1 to this enables power to the FIM. Writing 0 to this has no effect. |
| 27 | RW | PWRCTRL_DMA | 1 | Writing 1 to this enables power to the DMA. Writing 0 to this has no effect. |
| 26 | RW | PWRCTRL_IR | 1 | Writing 1 to this enables power to the IR. Writing 0 to this has no effect. |
| 25 | RW | PWRCTRL_ADC_DAC | 1 | Writing 1 to this enables power to the ADC/DAC. Writing 0 to this has no effect. |
| 24 | RW | PWRCTRL_I2C | 1 | Writing 1 to this enables power to the I2C. Writing 0 to this has no effect. |
| 23 | RW | PWRCTRL_I2S | 1 | Writing 1 to this enables power to the I2S. Writing 0 to this has no effect. |
| 22 | RW | PWRCTRL_SSI | 1 | Writing 1 to this enables power to the SPI/SSI. Writing 0 to this has no effect. |
| 21 | RW | PWRCTRL_UART | 1 | Writing 1 to this enables power to the UART. Writing 0 to this has no effect. |
| 20 | RW | PWRCTRL_VAD | 1 | Writing 1 to this enables power to the VAD. Writing 0 to this has no effect. |
| 19 | RW | PWRCTRL_TOUCH | 1 | Writing 1 to this enables power to the TOUCH. Writing 0 to this has no effect. |
| 18 | RW | PWRCTRL_TIMER | 1 | Writing 1 to this enables power to the TIMER. Writing 0 to this has no effect. |
| 17:0 | - | Reserved | - | It is recommended to write these bits to 0. |

228 . ULP Peripheral Power Control SET Register

7.9.16 ULP_Peripheral_Power_Control_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|--|
| 31:29 | - | Reserved | - | It is recommended to write these bits to 0. |
| 28 | RW | PWRCTRL_FIM | 1 | Writing 1 to this disables power to the FIM. Writing 0 to this has no effect. |
| 27 | RW | PWRCTRL_DMA | 1 | Writing 1 to this disables power to the DMA. Writing 0 to this has no effect. |
| 26 | RW | PWRCTRL_IR | 1 | Writing 1 to this disables power to the IR. Writing 0 to this has no effect. |
| 25 | RW | PWRCTRL_ADC_DAC | 1 | Writing 1 to this disables power to the ADC/DAC. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| 24 | RW | PWRCTRL_I2C | 1 | Writing 1 to this disables power to the I2C. Writing 0 to this has no effect. |
| 23 | RW | PWRCTRL_I2S | 1 | Writing 1 to this disables power to the I2S. Writing 0 to this has no effect. |
| 22 | RW | PWRCTRL_SSI | 1 | Writing 1 to this disables power to the SPI/SSI. Writing 0 to this has no effect. |
| 21 | RW | PWRCTRL_UART | 1 | Writing 1 to this disables power to the UART. Writing 0 to this has no effect. |
| 20 | RW | PWRCTRL_VAD | 1 | Writing 1 to this disables power to the VAD. Writing 0 to this has no effect. |
| 19 | RW | PWRCTRL_TOUCH | 1 | Writing 1 to this disables power to the TOUCH. Writing 0 to this has no effect. |
| 18 | RW | PWRCTRL_TIMER | 1 | Writing 1 to this disables power to the TIMER. Writing 0 to this has no effect. |
| 17:0 | - | Reserved | - | It is recommended to write these bits to 0. |

229 . ULP Peripheral Power Control CLEAR Register

7.9.17 ULP_SRAM_Power_Control_REG1_SET

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|--------------------|--------------------|---|
| 31:8 | - | Reserved | - | It is recommended to write these bits to 0. |
| 7 | RW | PWRCTRL1_ULPSRAM_8 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-8. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL1_ULPSRAM_7 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-7. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL1_ULPSRAM_6 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-6. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL1_ULPSRAM_5 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-5. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL1_ULPSRAM_4 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-4. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL1_ULPSRAM_3 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-3. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------|-------------|---|
| 1 | RW | PWRCTRL1_ULPSRAM_2 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-2. Writing 0 to this has no effect. |
| 0 | RW | PWRCTRL1_ULPSRAM_1 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-1. Writing 0 to this has no effect. |

230 . ULP SRAM Power Control Register1 SET

7.9.18 ULP_SRAM_Power_Control_REG1_CLEAR

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------|-------------|--|
| 31:8 | - | Reserved | - | It is recommended to write these bits to 0. |
| 7 | RW | PWRCTRL1_ULPSRAM_8 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-8. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL1_ULPSRAM_7 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-7. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL1_ULPSRAM_6 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-6. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL1_ULPSRAM_5 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-5. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL1_ULPSRAM_4 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-4. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL1_ULPSRAM_3 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-3. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL1_ULPSRAM_2 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-2. Writing 0 to this has no effect. |
| 0 | RW | PWRCTRL1_ULPSRAM_1 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-1. Writing 0 to this has no effect. |

231 . ULP SRAM Power Control Register1 CLEAR

7.9.19 ULP_SRAM_Power_Control_REG2_SET

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:8 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|---------------------|--------------------|---|
| 7 | RW | PWRCTRL2_ULPSRA_M_8 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-8. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL2_ULPSRA_M_7 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-7. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL2_ULPSRA_M_6 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-6. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL2_ULPSRA_M_5 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-5. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL2_ULPSRA_M_4 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-4. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL2_ULPSRA_M_3 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-3. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL2_ULPSRA_M_2 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-2. Writing 0 to this has no effect. |
| 0 | RW | PWRCTRL2_ULPSRA_M_1 | 1 | Writing 1 to this enables power to all Banks in ULP-SRAM-1. Writing 0 to this has no effect. |

232 . ULP SRAM Power Control Register2 SET

7.9.20 ULP_SRAM_Power_Control_REG2_CLEAR

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|---------------------|--------------------|--|
| 31:8 | - | Reserved | - | It is recommended to write these bits to 0. |
| 7 | RW | PWRCTRL2_ULPSRA_M_8 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-8. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL2_ULPSRA_M_7 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-7. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL2_ULPSRA_M_6 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-6. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL2_ULPSRA_M_5 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-5. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------|-------------|--|
| 3 | RW | PWRCTRL2_ULPSRA_M_4 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-4. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL2_ULPSRA_M_3 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-3. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL2_ULPSRA_M_2 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-2. Writing 0 to this has no effect. |
| 0 | RW | PWRCTRL2_ULPSRA_M_1 | 1 | Writing 1 to this disables power to all Banks in ULP-SRAM-1. Writing 0 to this has no effect. |

233 . ULP SRAM Power Control Register2 CLEAR

7.9.21 ULP_SRAM_Power_Control_REG4_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|--|
| 31:24 | - | Reserved | - | It is recommended to write these bits to 0. |
| 23 | RW | DS_ULPSRAM_8_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-8 Writing 0 to this has no effect. |
| 22 | RW | DS_ULPSRAM_7_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-7. Writing 0 to this has no effect. |
| 21 | RW | DS_ULPSRAM_6_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-6. Writing 0 to this has no effect. |
| 20 | RW | DS_ULPSRAM_5_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-5. Writing 0 to this has no effect. |
| 19 | RW | DS_ULPSRAM_4_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-4. Writing 0 to this has no effect. |
| 18 | RW | DS_ULPSRAM_3_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-3. Writing 0 to this has no effect. |
| 17 | RW | DS_ULPSRAM_2_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-2. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|--|
| 16 | RW | DS_ULPSRAM_1_B1 | 1 | Writing 1 to this enables deep sleep mode to 1 st Bank in ULP-SRAM-1. Writing 0 to this has no effect. |
| 15:0 | - | Reserved | - | It is recommended to write these bits to 0. |

234 . ULP SRAM Power Control Register3 SET

7.9.22 ULP_SRAM_Power_Control_REG4_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|---|
| 31:24 | - | Reserved | - | It is recommended to write these bits to 0. |
| 23 | RW | DS_ULPSRAM_8_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-8 Writing 0 to this has no effect. |
| 22 | RW | DS_ULPSRAM_7_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-7. Writing 0 to this has no effect. |
| 21 | RW | DS_ULPSRAM_6_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-6. Writing 0 to this has no effect. |
| 20 | RW | DS_ULPSRAM_5_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-5. Writing 0 to this has no effect. |
| 19 | RW | DS_ULPSRAM_4_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-4. Writing 0 to this has no effect. |
| 18 | RW | DS_ULPSRAM_3_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-3. Writing 0 to this has no effect. |
| 17 | RW | DS_ULPSRAM_2_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-2. Writing 0 to this has no effect. |
| 16 | RW | DS_ULPSRAM_1_B1 | 1 | Writing 1 to this disables deep sleep mode to 1 st Bank in ULP-SRAM-1. Writing 0 to this has no effect. |
| 15:0 | - | Reserved | - | It is recommended to write these bits to 0. |

235 . ULP SRAM Power Control Register3 CLEAR

7.9.23 UULP_Peripheral_Power_Control_SET

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|---|
| 31:11 | - | Reserved | - | It is recommended to write these bits to 0. |
| 10 | RW | PWRCTRL_CLOCK_CALIB | 1 | Writing 1 to this enables power to the CLOCK-CALIB. Writing 0 to this has no effect. |
| 9 | RW | PWRCTRL_STORAGE_DOMAIN3 | 1 | Writing 1 to this enables power to the STORAGE-DOMAIN3. Writing 0 to this has no effect. |
| 8 | RW | PWRCTRL_STORAGE_DOMAIN2 | 1 | Writing 1 to this enables power to the STORAGE-DOMAIN2. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL_STORAGE_DOMAIN1 | 1 | Writing 1 to this enables power to the STORAGE-DOMAIN1. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL_TS | 1 | Writing 1 to this enables power to the TS. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL_PS | 1 | Writing 1 to this enables power to the PS. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL_WDT | 1 | Writing 1 to this enables power to the WDT. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL_RTC | 1 | Writing 1 to this enables power to the RTC. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL_FSM | 1 | Writing 1 to this enables power to the FSM. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL_BBFFS | 1 | Writing 1 to this enables power to the BBFFS. Writing 0 to this has no effect. |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

236 .UULP Peripheral Power Control SET Register

7.9.24 UULP_Peripheral_Power_Control_CLEAR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|--|
| 31:11 | - | Reserved | - | It is recommended to write these bits to 0. |
| 10 | RW | PWRCTRL_CLOCK_CALIB | 1 | Writing 1 to this disables power to the CLOCK-CALIB. Writing 0 to this has no effect. |
| 9 | RW | PWRCTRL_STORAGE_DOMAIN3 | 1 | Writing 1 to this disables power to the STORAGE-DOMAIN3. Writing 0 to this has no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------------|-------------|--|
| 8 | RW | PWRCTRL_STORAGE_DO_MAIN2 | 1 | Writing 1 to this disables power to the STORAGE-DOMAIN2. Writing 0 to this has no effect. |
| 7 | RW | PWRCTRL_STORAGE_DO_MAIN1 | 1 | Writing 1 to this disables power to the STORAGE-DOMAIN1. Writing 0 to this has no effect. |
| 6 | RW | PWRCTRL_TS | 1 | Writing 1 to this disables power to the TS. Writing 0 to this has no effect. |
| 5 | RW | PWRCTRL_PS | 1 | Writing 1 to this disables power to the PS. Writing 0 to this has no effect. |
| 4 | RW | PWRCTRL_WDT | 1 | Writing 1 to this disables power to the WDT. Writing 0 to this has no effect. |
| 3 | RW | PWRCTRL_RTC | 1 | Writing 1 to this disables power to the RTC. Writing 0 to this has no effect. |
| 2 | RW | PWRCTRL_FSM | 1 | Writing 1 to this disables power to the FSM. Writing 0 to this has no effect. |
| 1 | RW | PWRCTRL_BBFFS | 1 | Writing 1 to this disables power to the BBFFS. Writing 0 to this has no effect. |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

237 . UULP Peripheral Power Control CLEAR Register

7.9.25 FSM_SLEEP_CTRLS_AND_WAKEUP_MODE

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------|-------------|--|
| 31 | - | Reserved | - | It is recommended to write these bits to 0. |
| 30 | RW | ULP_Peripheral_Sleep | 0 | Writing 1 to this enables ULP Peripheral Interrupt as a Sleep source Writing 0 to this disables ULP Peripheral Interrupt as a Sleep source |
| 29 | RW | WDT_Wakeup | 0 | Writing 1 to this enables WDT Interrupt as a Wakeup source Writing 0 to this disables WDT Interrupt as a Wakeup source Note: Set "EN_WDT_SLEEP" in FSM_CTRL_POWER_DOMAINS register. |
| 28 | RW | Millisecond_Wakeup | 0 | Writing 1 to this enables Milli-Second Interrupt as a Wakeup source Writing 0 to this disables Milli-Second Interrupt as a Wakeup source |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-------------------------|-------------|--|
| 27 | RW | Second_Wakeup | 0 | Writing 1 to this enables Second Interrupt as a Wakeup source Writing 0 to this disables Second Interrupt as a Wakeup source |
| 26 | RW | Alarm_Wakeup | 0 | Writing 1 to this enables ALARM Interrupt as a Wakeup source Writing 0 to this disables ALARM Interrupt as a Wakeup source |
| 25 | RW | SDC_Wakeup | 0 | Writing 1 to this enables Sensor Data Collector Interrupt as a Wakeup source Writing 0 to this disables Sensor Data Collector Interrupt as a Wakeup source |
| 24 | RW | ULP_Peripheral_Wakeu p | 0 | Writing 1 to this enables ULP Peripheral Interrupt as a Wakeup source Writing 0 to this disables ULP Peripheral Interrupt as a Wakeup source |
| 23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | RW | WuRx_Wakeup | 0 | Writing 1 to this enables Wake-Fi Rx Interrupt as a Wakeup source Writing 0 to this disables Wake-Fi Rx Interrupt as a Wakeup source |
| 21 | RW | CMPR_BOD_BUTTON_ Wakeup | 0 | Writing 1 to this enables 4x-Comparator/BOD/ BUTTON Interrupt as a Wakeup source Writing 0 to this disables 4x-Comparator/BOD/ BUTTON Interrupt as a Wakeup source |
| 20 | RW | UULP_Vbat_GPIO_Wak eup | 0 | Writing 1 to this enables UULP Vbat GPIO Interrupt as a Wakeup source Writing 0 to this disables UULP Vbat GPIO Interrupt as a Wakeup source The Selection of UULP Vbat GPIO's for wakeup is described "GPIO_WAKEUP_CONFIG" Register (Refer GPIO Configuration Section). |
| 19 | - | Reserved | 0 | It is recommended to write these bits to 0. |
| 18 | RW | NWP_Wakeup | 0 | Writing 1 to this enables NWP Interrupt as a Wakeup source Writing 0 to this disables NWP Interrupt as a Wakeup source |
| 17 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------------|-------------|--|
| 16 | RW | TIMER_Wakeup | 0 | Writing 1 to this enables Deep-Sleep Timer Interrupt as a Wakeup source Writing 0 to this disables Deep-Sleep Timer Interrupt as a Wakeup source |
| 15:12 | - | Reserved | - | It is recommended to write these bits to 0. |
| 11 | RW | SKIP_XTAL_WAIT_TIME | 0 | Writing 1 to this skips the settling time for High Frequency XTAL during wakeup. Writing 0 to this includes the settling time for High Frequency XTAL during wakeup. |
| 10 | RW | DCDC_ON | 0 | Writing 1 to this maintains DC-DC 1.35 in ON state during Sleep. Writing 0 to this maintains DC-DC 1.35 in OFF state during Sleep. |
| 9 | RW | LDoFL_ON | 0 | Writing 1 to this maintains LDO FL 1.8 in ON state during Sleep. Writing 0 to this maintains LDO FL 1.8 in OFF state during Sleep. |
| 8 | RW | LDoSoC_ON | 0 | Writing 1 to this maintains LDO SoC 1.1 in ON state during Sleep. Writing 0 to this maintains LDO SoC 1.1 in OFF state during Sleep. |
| 7 | RW | LP_SRAM_16KB_RETENTION_EN | 0 | SRAM retention Control for 16KB of LP-SRAM (LP-SRAM-1, LP-SRAM-2, LP-SRAM-3, LP-SRAM-4) Writing 1 to this enables Retention during sleep Writing 0 to this disables Retention during sleep |
| 6 | RW | ULP_SRAM_RETENTION_EN | 0 | SRAM retention Control for 16KB of ULP-SRAM Writing 1 to this enables Retention during sleep Writing 0 to this disables Retention during sleep |
| 5 | RW | HP_SRAM2_RETENTION_EN | 0 | SRAM retention Control for 192KB of HP-SRAM2 Writing 1 to this enables Retention during sleep Writing 0 to this disables Retention during sleep |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------------|-------------|--|
| 4 | RW | LP_SRAM_RETENTION_0 EN | 0 | SRAM retention Control for 112KB of LP-SRAM (LP-SRAM-5, LP-SRAM-6, LP-SRAM-7) Writing 1 to this enables Retention during sleep Writing 0 to this disables Retention during sleep |
| 3 | RW | HP_SRAM1_RETENTION_0 EN | 0 | SRAM retention Control for 64KB of HP-SRAM1 Writing 1 to this enables Retention during sleep Writing 0 to this disables Retention during sleep |
| 2:0 | - | Reserved | - | It is recommended to write these bits to 0. |

238 .FSM SLEEP CTRLs and WAKEUP MODE Register

7.9.26 FSM_ONTIME_CONFIG_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|---|
| 31:21 | - | Reserved | - | It is recommended to write these bits to 0. |
| 20:16 | RW | PMU_POWERGOOD_TIME | 15 | Specifies the combined ON Time for DC-DC 1.35 and LDO-SoC 1.1 0 - 10us 1 - 20us 2 - 25us 3 - 50us 4 - 100us 5 - 150us 31 - 1450us |
| 15:5 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------------|-------------|---|
| 4:0 | RW | HF_CRYSTAL_SETTLING_TIME | 15 | <p>Specifies the Settling Time for HF-Crystal Clock</p> <p>0 - 10us</p> <p>1 - 20us</p> <p>2 - 25us</p> <p>3 - 50us</p> <p>4 - 100us</p> <p>5 - 150us</p> <p>.....</p> <p>31 - 1450us</p> |

239 .FSM ON-Time Configuration Register

7.9.27 ULP_MODE_CONFIG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------|-------------|--|
| 31:17 | - | Reserved | - | It is recommended to write these bits to 0. |
| 16 | RW | bgpmu_sampling_en | 0 | <p>Controls the mode of Band-Gap for DC-DC 1.35 during PS2 state.</p> <p>Writing 1 to this enables sampling mode of Band-Gap. This is described in Power Management Section.</p> <p>Writing 0 to this disables sampling mode of Band-Gap. This is described in Power Management Section.</p> |
| 15:3 | - | Reserved | - | It is recommended to write these bits to 0. |
| 2 | RW | ULP_MODE_MEM_CONFIG | 0 | <p>Writing 1 to this enables a maximum of 32KB of LP-SRAM for operation in PS2 state</p> <p>Writing 0 to this enables a maximum of 128KB of LP-SRAM for operation in PS2 state</p> |
| 1 | RW | ULP_MODE_FUNC_SWITCH | 0 | <p>Enable functional switching for PS2-PS4/PS3 and PS4/PS3-PS2 state transitions</p> <p>Writing 1 to this enables functional switching for PS4/PS3-PS2 state transition</p> <p>Writing 0 to this enables functional switching for PS2-PS4/PS3 state transition</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------|-------------|--|
| 0 | RW | ULP_MODE_VOLT_SWITCH | 0 | <p>Enables voltage switching for PS2-PS4/PS3 and PS4/PS3-PS2 state transitions</p> <p>Writing 1 to this enables voltage switching for PS4/PS3-PS2 state transition</p> <p>Writing 0 to this enables voltage switching for PS2-PS4/PS3 state transition</p> |

240 . ULP Mode Configuration Register

7.9.28 FSM_POWER_CTRL_DELAY

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------|-------------|---|
| 31:28 | - | Reserved | - | It is recommended to write these bits to 0. |
| 27:26 | RW | VOLTAGE_SEL_ULP_SRAM | | <p>Configures the Voltage source to be used for LOW-VOLTAGE-ULPRAM Domain in PS2 state</p> <p>3 – LDO SoC 1.1</p> <p>1 – DC-DC 0.95</p> <p>0 – Reserved</p> |
| 25:24 | RW | VOLTAGE_SEL_LP_SRAM | 3 | <p>Configures the Voltage source to be used for LOW-VOLTAGE-LPRAM Domain in PS2 state</p> <p>3 – LDO SoC 1.1</p> <p>1 – DC-DC 0.95</p> <p>0 – Reserved</p> |
| 23:22 | RW | VOLTAGE_SEL_LP_SRAM_16KB | 3 | <p>Configures the Voltage source to be used for LOW-VOLTAGE-LPRAM-16KB Domain in PS2 state</p> <p>3 – LDO SoC 1.1</p> <p>1 – DC-DC 0.95</p> <p>0 – Reserved</p> |
| 21:20 | RW | VOLTAGE_SEL_PROC | 3 | <p>Configures the Voltage source to be used for PROC-DOMAIN Domain in PS2 state</p> <p>3 – LDO SoC 1.1</p> <p>1 – DC-DC 0.95</p> <p>0 – LDO 0.7V</p> |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|---|
| 19 | RW | VOLTAGE_SEL_ULP_PER_IPH | 1 | Configures the Voltage source to be used for LOW-VOLTAGE-ULPPERIPH Domain in PS2 state 1 – LDO SoC 1.1 0 – DC-DC 0.95 |
| 18 | - | Reserved | 0 | Reserved |
| 17 | RW | DCDC_EN | 0 | Writing 1 to this configures DC-DC 1.35 to ON state during PS2 Writing 0 to this configures DC-DC 1.35 in OFF state during PS2 |
| 16 | RW | LDoSoC_EN | 0 | Writing 1 to this configures LDO SoC 1.1 to ON state during PS2 Writing 0 to this configures LDO SoC 1.1 in OFF state during PS2 |
| 15:12 | RW | DCDC_ON_TIME | 0 | Configures the time for switching ON the DC-DC 1.35 during transition from PS2 to PS4 state. 0 - 50us 1 - 100us 2 - 200us 3 - 300us .. 15 - 1500us |
| 11:8 | RW | LDoSoC_ON_TIME | 0 | Configures the time for switching ON the LDO SoC 1.1 during transition from PS2 to PS4 state. 0 - 50us 1 - 100us 2 - 200us 3 - 300us .. 15 - 1500us |
| 7:0 | - | Reserved | - | It is recommended to write these bits to 0. |

241 . FSM Power Control Delay Register

7.9.29 FSM_CTRL_POWER_DOMAINS

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|---|
| 31:20 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 19 | RW | PWRCTRL_RETEN | 1 | Writing 1 to this enables Power to Retention Flops during SHIP state. These Flops are used for storing Chip Configuration. Writing 0 to this disables Power to Retention Flops during SHIP state. These Flops are used for storing Chip Configuration. |
| 18 | RW | PWRCTRL_DS_TIME | 1 | Writing 1 to this enables Power to DEEP SLEEP Timer. Writing 0 to this disables Power to DEEP SLEEP Timer. |
| 17 | RW | PWRCTRL_TIMESTAMP | 1 | Writing 1 to this enables Power to TIMESTAMP. Writing 0 to this disables Power to TIMESTAMP. |
| 16 | RW | PWRCTRL_LP_FSM | 1 | Writing 1 to this enables Power to Low-Power FSM. Writing 0 to this disables Power to Low-Power FSM. |
| 15:3 | - | Reserved | - | It is recommended to write these bits to 0. |
| 2 | RW | RESET_BFF_EN | 0 | Writing 1 to this enables reset of Power Domain Control Battery FF's on wakeup Writing 0 to this disables reset of Power Domain Control Battery FF's on wakeup |
| 1 | RW | WakeFi_Rx_EN | 0 | Writing 1 to this enables detection of On-Air Pattern using Wake-Fi Rx Writing 0 to this disables detection of On-Air Pattern using Wake-Fi Rx |
| 0 | RW | EN_WDT_SLEEP | 0 | Writing 1 to this enables WDT during Sleep/Shutdown states. Writing 0 to this disables WDT during Sleep/Shutdown states. |

242 . FSM Controlled POWER Domains Register

7.9.30 ULP_ISOLATION_CTRL

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------|-------------|---|
| 31:6 | - | Reserved | - | It is recommended to write these bits to 0. |
| 5:0 | RW | ULP_ISOLATION_CTRL | 0 | Writing 'h3F to this enables Isolation between Low-Voltage and HIGH VOLTAGE Domains. Writing 'h00 to this disables Isolation between Low-Voltage and HIGH VOLTAGE Domains. |

243 . ULP ISOLATION Control Register

8 Power Management Unit

8.1 General Description

Power Management Unit (PMU) manages the power requirements of the SOC in different modes of operation. The PMU includes one DC-DC switching regulator (DC-DC 1.35) for powering LDO SOC and LDO ANA 1.2 . The DC-DC regulator supports a wide programmable output range of 0.8 to 1.5V. All the features of DC-DC 1.35 regulator can be controlled by the SOC, enabling dynamic power management. This regulator can handle up to 3.6V while ensuring high efficiency over a wide range of load currents. This PMU has two Low Drop Out regulators. The first LDO (LDO SOC 1.1) is used to power the SOC and can support 300mA load. This LDO provides 1.15V and can be programmed from 0.5 to 1.2V. The second LDO (LDO FL 1.8) can provide 1.6 to 2.8V programmable output voltage and support 125mA load current. The PMU also has internal bandgap reference generation which provides reference voltages to DC-DC 1.35 converter and LDOs.

8.2 Features

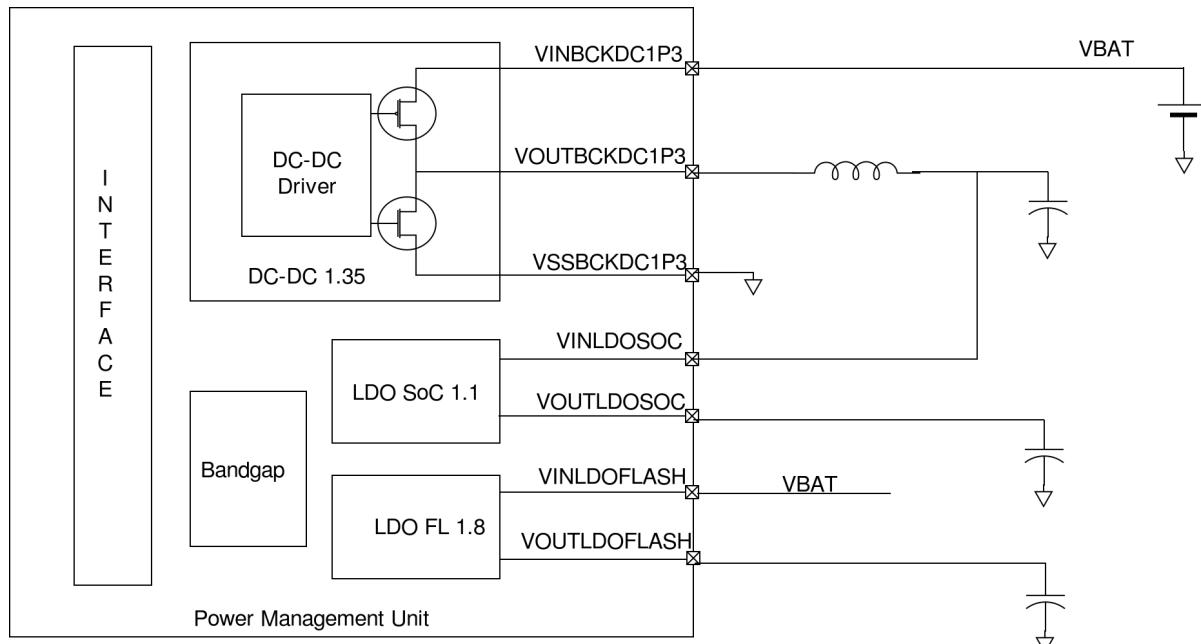
The PMU has following features,

1. PMU modes of Operation for high system efficiency
 - a. Active mode
 - b. Sleep mode
 - c. Ultra sleep mode
 - d. LDO switch mode
2. Bypass Options

8.3 Functional Description

8.3.1 Block Diagram

Following is the block diagram of Power Management Unit



Block Diagram of Power Management Unit

8.3.2 Modes of Operation

PMU can be configured to one of the following modes of operation to optimize the overall system efficiency for various system load requirements.

Active Mode

PMU can be configured to Active mode through APIs. In the active mode, PMU supports maximum power requirement of SOC. DC-DC 1.35 converter works in PWM mode with fixed frequency to achieve high efficiency at high load condition. The LDOs can support their maximum load current.

Sleep Mode

PMU can be configured to Sleep mode through APIs. In the sleep mode, PMU supports no more than 50mA current for DC-DC 1.35 and LDO SOC 1.1. DC-DC 1.35 converter works in PFM mode with variable frequency to achieve high efficiency at low load condition. In this mode, LDO FL 1.8 can support max current.

Ultra Sleep Mode

PMU can be configured to Ultra Sleep mode through APIs. In the Ultra sleep mode, both the LDOs are turned off and DC-DC 1.35 is configured in PFM mode. In this mode, the PMU consumes less than 1uA current and retain the buck output voltage to 1.2V.

LDO Switch Mode

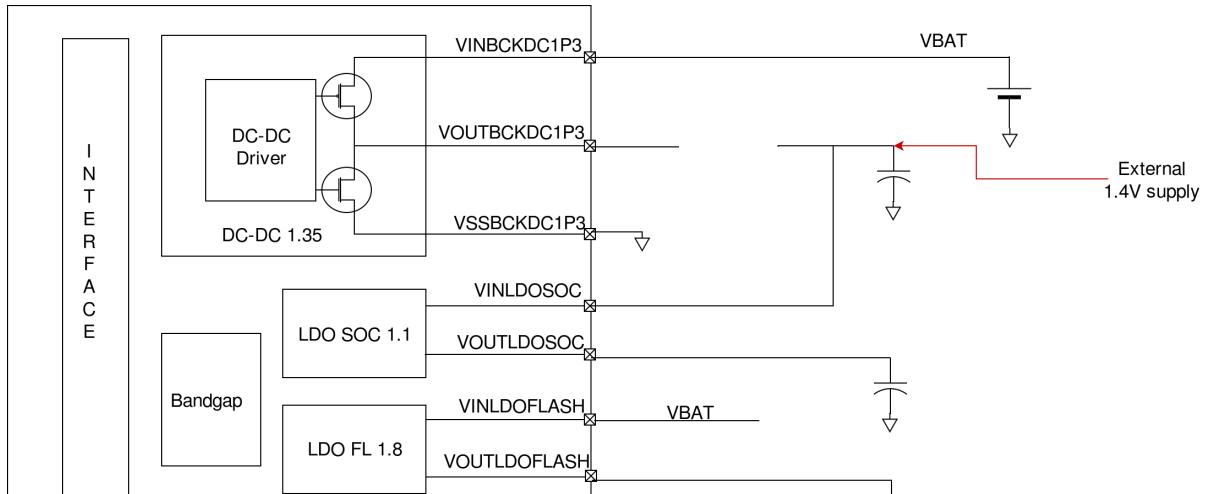
Both the LDOs can be configured in LDO switch mode through APIs. In LDO switch mode, LDO is bypassed and power MOSFET used as switch to pass input voltage directly to the output voltage. It can be configured in PMU active or PMU sleep mode.

8.3.3 Bypass Options

The PMU has a feature to use external supply instead of PMU internal supplies. In this case the PMU block can be bypassed by configuring the PMU external connection.

DC-DC 1.35 bypass

To bypass PMU internal DC-DC 1.35 and use external supply, the PMU should be connected as per the following configuration. In this configuration, DC-DC 1.35 inductor is removed and 1.35V external supply is given to VINLDO SOC.



Power Management Unit



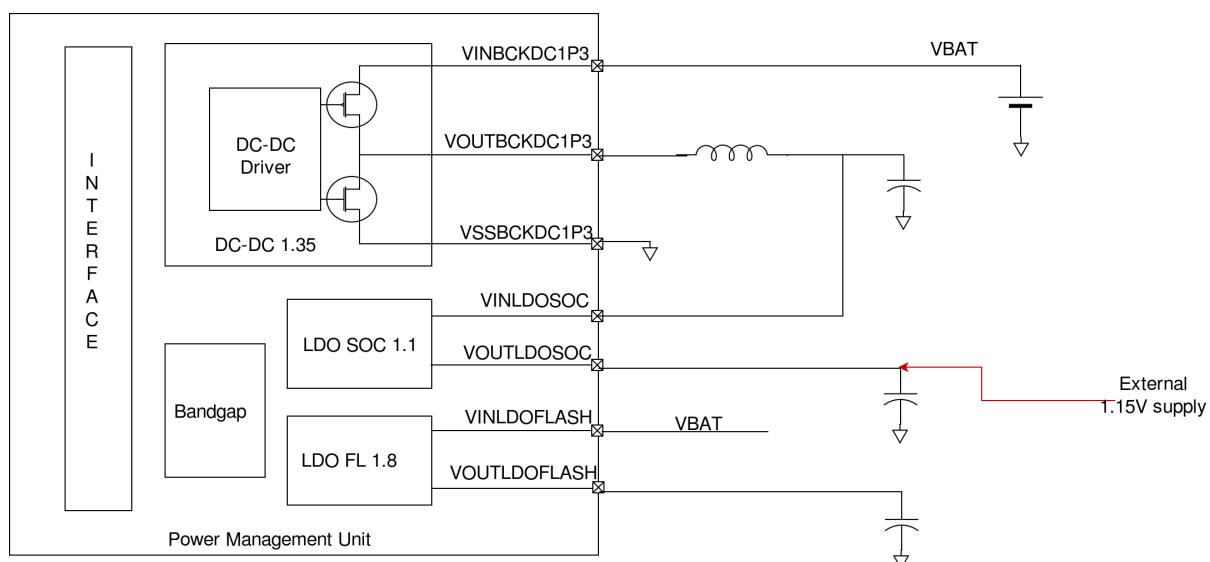
DC-DC 1.35 bypass

Following is the sequence to be followed for power up in DC-DC 1.35 bypass mode,

1. Remove buck inductor
2. Apply VBAT to VINBCKDC1P3 and VINLDOFLASH
3. Apply External 1.4V to VINLDOSOC after a delay of 250uS

LDO SOC 1.1 bypass

To bypass PMU internal LDO SOC 1.1 and use external supply, the PMU should be connected as per the following configuration. In this configuration, LDO SOC 1.1 output is connected to the external supply of 1.15V.



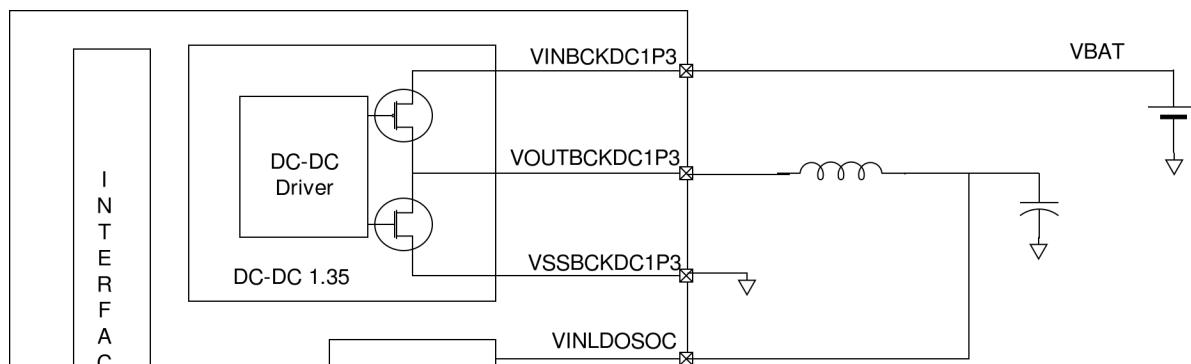
LDO SOC 1.1 bypass

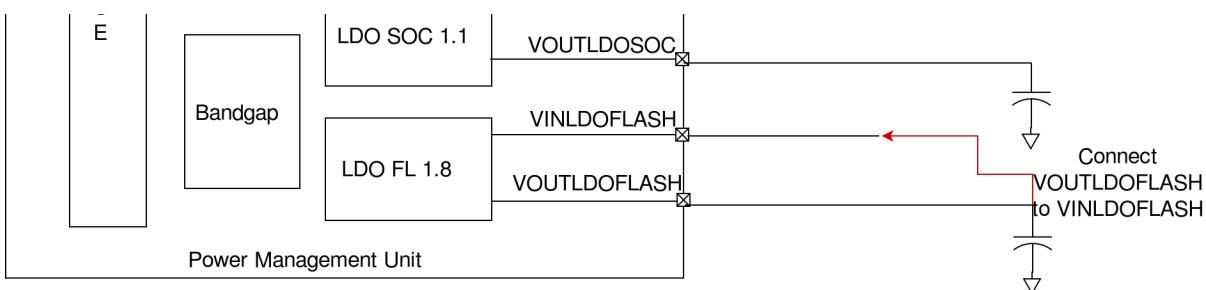
Following is the sequence to be followed for power up in LDO SOC 1.1 bypass mode,

1. Apply VBAT to VINBCKDC1P3 and VINLDOFLASH
2. Apply External 1.15V to VOUTLDOSOC after a delay of 600uS

LDO FL 1.8 bypass

To bypass PMU internal LDO FL 1.8. the PMU should be connected as per the following configuration.





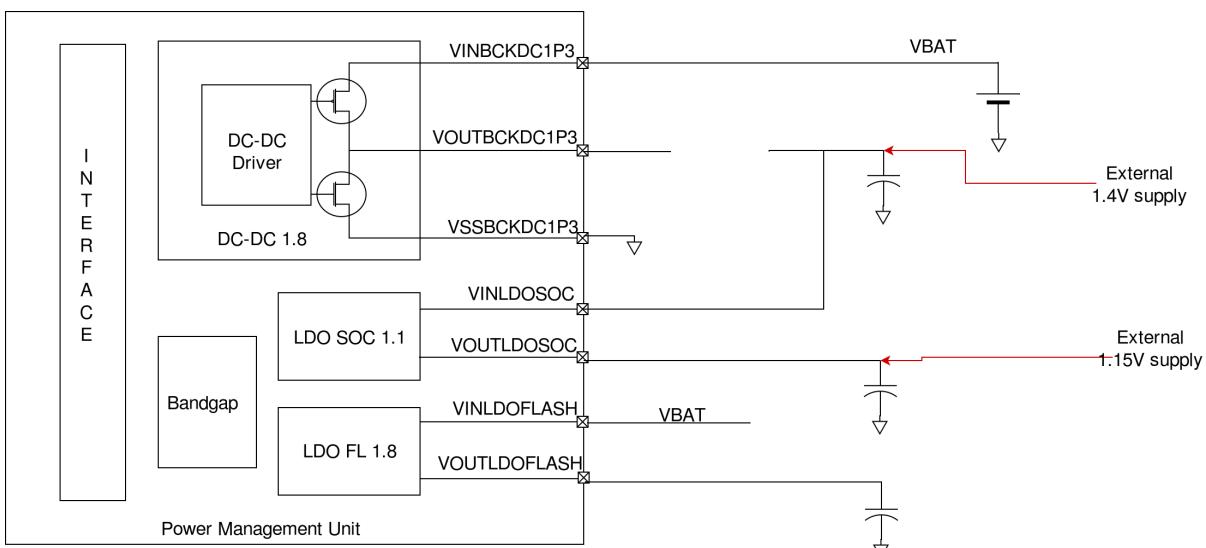
LDO FL 1.8 bypass

Following is the sequence to be followed for power up in LDO FL 1.8 bypass mode,

1. Apply VBAT to VINDCDC1P3, PMU_AVDD and VINLDOFLASH
2. Connect VOUTLDOFLASH to VINLDOFLASH in case of Flash supply is 3.3V
3. Connect VOUTLDOFLASH to 1.8V external supply after a delay of 250uS in case of Flash supply is 1.8V

DC-DC 1.35 + LDO SOC 1.1 bypass

To bypass PMU internal DC-DC 1.35 and LDO SOC 1.1. the PMU should be connected as per the following configuration.



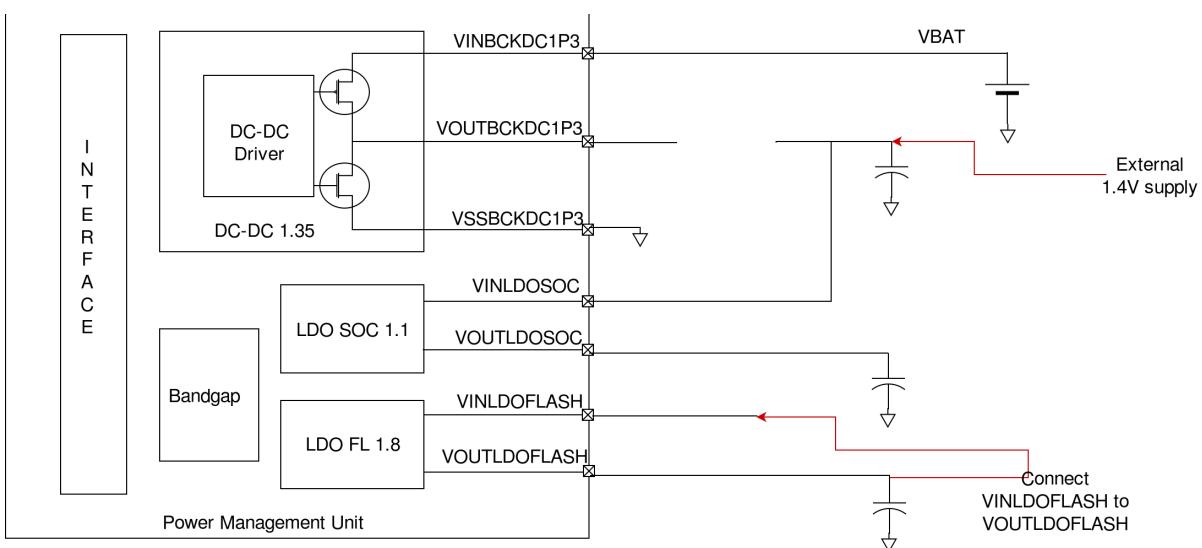
DC-DC 1.35 + LDO SOC 1.1 bypass

Following is the sequence to be followed for power up in DC-DC 1.35 and LDO SOC 1.1 bypass mode,

1. Remove the buck inductor
2. Apply VBAT to VINBCKDC1P3 and VINLDORF
3. Apply external supply of 1.4V to VINLDO SOC after a delay of 250uS
4. Apply external supply of 1.15V to VOUTLDO SOC after a delay of 600uS

DC-DC 1.35 + LDO FL 1.8 bypass

To bypass PMU internal DC-DC 1.35 and LDO FL 1.8. the PMU should be connected as per the following configuration.



DC-DC 1.35 + LDO FL 1.8 bypass

Following is the sequence to be followed for power up in DC-DC 1.35 and LDO FL 1.8 bypass mode,

1. Remove the buck inductor
2. Apply VBAT to VINBCKDC1P3 and VINLDOFLASH
3. Apply external supply of 1.4V to VINLDOSOC after a delay of 250uS
4. LDO FL 1.8 connection
 - a. Connect VOUTLDOFLASH to VINLDOFLASH in case of Flash supply is 3.3V
 - b. Connect VOUTLDOFLASH to 1.8V external supply after a delay of 250uS in case of Flash supply is 1.8V

LDO SOC 1.1 + LDO FL 1.8 bypass

To bypass PMU internal LDO SOC 1.1 and LDO FL 1.8. the PMU should be connected as per the following configuration.

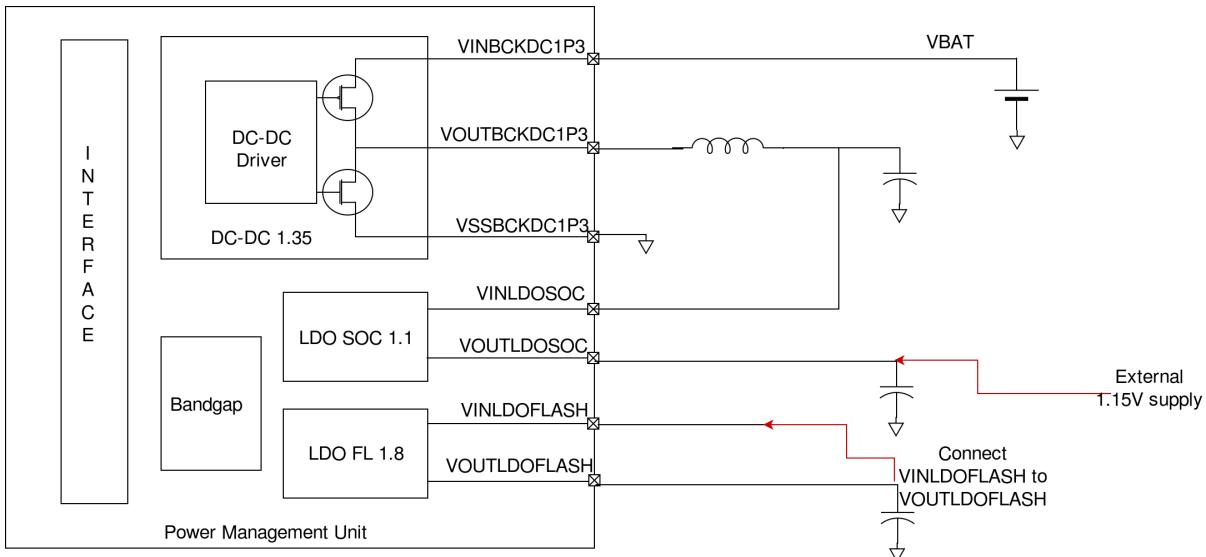


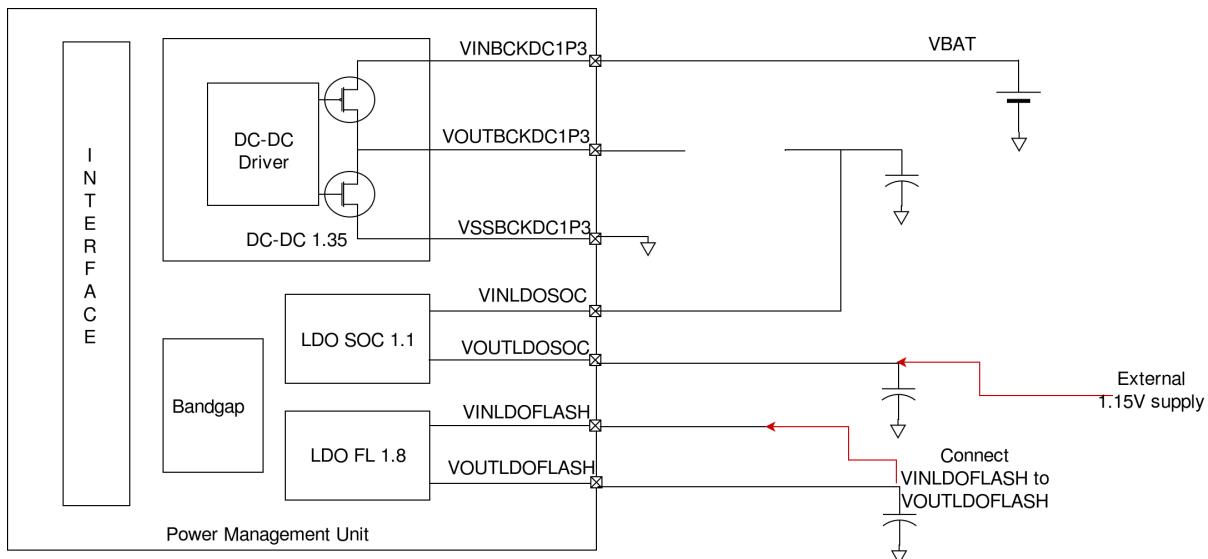
Figure 52: LDO SOC 1.1 + LDO FL 1.8 bypass

Following is the sequence to be followed for power up in LDO SOC 1.1 and LDO FL 1.8 bypass mode,

1. Apply VBAT to VINBCKDC1P3 and VINLDOFLASH
2. Apply external supply of 1.15V to VOUTLADOSOC after a delay of 600uS
3. LDO FL 1.8 connection
 - a. Connect VOUTLDOFLASH to VINLDOFLASH in case of Flash supply is 3.3V
 - b. Connect VOUTLDOFLASH to 1.8V external supply after a delay of 250uS in case of Flash supply is 1.8V

DC-DC 1.35 + LDO SOC 1.1 + LDO FL 1.8 bypass

To bypass PMU internal DC-DC 1.35, LDO SOC 1.1 and LDO FL 1.8. the PMU should be connected as per the following configuration.



DC-DC 1.35 + LDO SOC 1.1 + LDO FL 1.8 bypass

Following is the sequence to be followed for power up in DC-DC 1.35, LDO SOC 1.1 and LDO FL 1.8 bypass mode,

1. Remove the buck inductor
2. Apply VBAT to VINBCKDC1P3 and VINLDOFLASH
3. Apply external supply of 1.4V to VINLADOSOC after a delay of 250uS
4. Apply external supply of 1.15V to VOUTLADOSOC after a delay of 600uS
5. LDO FL 1.8 connection
 - a. Connect VOUTLDOFLASH to VINLDOFLASH in case of Flash supply is 3.3V
 - b. Connect VOUTLDOFLASH to 1.8V external supply after a delay of 250uS in case of Flash supply is 1.8V

8.4 Register Summary

Base Address: 0x2405_8000

| Register Name | offset | Description |
|------------------|--------|-------------|
| PMU_IP3_CTRL_REG | 0x740 | |
| PMU_LDOSOC_REG | 0x758 | |

8.5 Register Description

8.5.1 PMU_IP3_CTRL_REG

| Bit | Access | Function | Default Value | Description | Dynamic Controllable |
|-------|--------|-------------|---------------|---|----------------------|
| 20:17 | R/W | set_vref1p3 | 4'd11 | Set DC-DC 1.35 output voltage. 0000 - 0.8V 0001 - 0.86V 0010 - 0.91V 0011 - 0.96V 0100 - 1.01V 0101 - 1.06V 0110 - 1.11V 0111 - 1.16V 1000 - 1.21V 1001 - 1.26V 1010 - 1.31V 1011 - 1.36V 1100 - 1.41V 1101 - 1.46V 1110 - 1.51V 1111 - 1.56V | Yes |

8.5.2 PMU_LDOSOC_REG

| Bit | Access | Function | Default Value | Description | Dynamic Controllable |
|-----|--------|---------------|---------------|---|----------------------|
| 9:6 | R/W | CLRL_LDOFLASH | 4'd3 | Set LDO FL 1.8 Output voltage 0000 - 1.6V 0001 - 1.68V 0010 - 1.76V 0011 - 1.84V 0100 - 1.92V 0101 - 2V 0110 - 2.08V 0111 - 2.16V 1000 - 2.24V 1001 - 2.32V 1010 - 2.4V 1011 - 2.48V 1100 - 2.56V 1101 - 2.64V 1110 - 2.72V 1111 - 2.8V | Yes |
| 3:0 | R/W | CTRL_LDOSOC | 4'd11 | Set LDO SOC 1.1 output voltage 0000 - 0.50 V 0001 - 0.55 V 0010 - 0.60 V 0011 - 0.65 V 0100 - 0.70 V 0101 - 0.75 V 0110 - 0.80 V 0111 - 0.90 V 1000 - 0.95 V 1001 - 1 V 1010 - 1.05 V 1011 - 1.10 V 1100 - 1.15 V 1101 - 1.2 V 1110 - 1.25 V 1111 - 1.3 V | Yes |

8.6 PMU Good Time

8.6.1 Direct Battery Connected PMU Good Time

If the battery is directly connected to VINBCKDC and VINLDOFLASH then PMU takes following time to generate PMU power good. This data is based on simulation results

| Blocks | Up Time (us) | Up Time (us) | Up Time (us) |
|------------------|---------------------------------------|---|--------------------------------|
| | (From Supply Rampup to PMU On) | (From Ultra Sleep mode to PMU Active mode) | (From LDO FL Off to On) |
| Supply Rampup* | 10 | - | - |
| Bandgap | 225 | 225 | - |
| Buck | 170 | 25 | - |
| LDOSOC | 15 | 15 | - |
| LDORF | 20 | 20 | 20 |
| PMU Total | 430 | 285 | 20 |

- If supply rampup time increased, PMU powergood time will also increase. Bandgap will start once supply reached to 1.6V.

8.6.2 Cascaded Power Supply PMU Good Time

If VINBCKDC and VINLDOFLASH are connected through an external power gate (PGATE) to battery or are powered by an external BOOST regulator, then add the above numbers to the Tstab (the voltage stabilization time of the BOOST or power gate) to arrive at the final PMU_GOOD_TIME. Tstab is defined as the time it takes for the BOOST or PGATE to charge the VINBCKDC and VINDLOFLASH to 1.6V.

For example, if Tstab is 20us, then the final PMU_Good_Time from supply ramp up to PMU On is 450us (430us+20us).

9 ULP Regulators

9.1 General Description

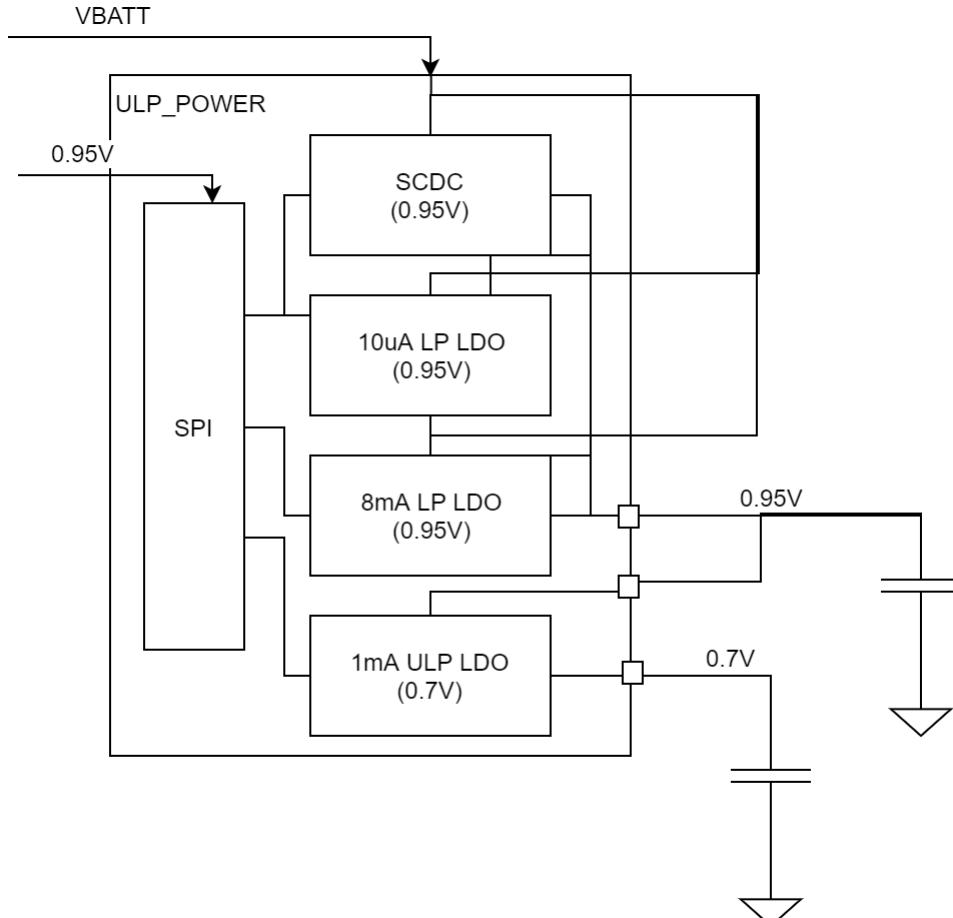
ULP (Ultra Low Power) regulators are used to power low power Always-ON (AON) digital and analog power management circuitry inside the IC. In addition to this, the ULP regulators can also be used to power the SRAMs and M4 core. The ULP regulators include two high power LDOs , a Low power LDO, and a switched capacitor DC-DC regulator. These regulators operate directly off of VBATT.

9.2 Features

1. Support wide programmable output voltage range at very low quiescent current
2. Support wide current ranges (0.4uA to 8mA)
3. Have two modes of operation:
 - a. High Power Mode
 - b. Low Power Mode
4. Support modes to bypass internal regulators

9.3 Functional Description

9.3.1 Block Diagram



Block Diagram of ULP Regulators

Power Up:

Upon Battery insertion, voltage VBATT ramps up there by enabling the Bandgap (UULP_VBAT_Peripheral). The Bandgap generates a stable voltage reference to the 8mA LP LDO which in turn generates a stable 1.0 V output. Power On Control (POC), another UULP_VBAT_Peripheral, monitors the output of the regulator and holds the IC under reset till the output voltage of this regulator reaches a value that is high enough to facilitate a safe operation. Upon release of the POC reset, the LDO 0.7V is enabled. This LDO can support a maximum load current of 1mA.

Low Power State:

When the IC needs to enter a low power state, then the provided APIs can be used to switch OFF the 8mA LDO and transition over to the 10uA LDO. This transition would enable lowering the quiescent current consumed by the LDO.

SC DC-DC Mode:

A switched capacitor DC-DC regulator can be used, instead of the 8mA LP LDO, to improve the power conversion efficiency of the IC. The DC-DC converter loop automatically tracks the VBATT and changes the built in gain to provide a high efficiency across a wide input voltage range (2.1 to 3.6V). One can use the provided API, to automatically transition over to the SC DC-DC mode after the first power up and continue to remain in this mode till VBATT falls below 2.1V. At VBATT lower than 2.1V, the DC-DC regulator begins to operate in linear mode, hence the API automatically transitions back to the 8mA LDO.

Bypass Options:

Each of the LDOs can be bypassed by over-driving the outputs with either a high efficiency buck regulator or a linear regulator

9.4 Register Summary

Base Address: 0x2405_A000

| Register Name | Offset | Reset Value | Description |
|--------------------|--------|-------------|--|
| SCDC_CTRL_REG_0 | 0x498 | 22'h1E002F | SCDC-DC Algorithm Control Register |
| BG_SCDC_PROG_REG_1 | 0x49C | 22'h200498 | DC-DC / LDO Output Programmability Selection |
| BG_SCDC_PROG_REG_2 | 0x4A0 | 22'h000050 | Enable Controls |
| BG_LDO_REG | 0x4A4 | 22'h088000 | LDO 0.7V Controls |

9.5 Register Description

9.5.1 SCDC_CTRL_REG_0

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------------|---------------|---|
| 21 | R/W | ext_cap_en | 1'b0 | To change current trim bits to high or low through spi, based on high power or low power mode. When 0, curr prog value is 0. |
| 20:17 | R/W | fixed_curr_prog_high | 4'd15 | Current prog value to take when ext cap en is high and sel_high freq_ext_b is 0 |
| 16:13 | R/W | fixed_curr_prog_low | 4'd0 | Current prog value to take when ext cap en is high and sel_high freq_ext_b is 1 |
| 12 | R/W | bypass_trim_ro | 1'b0 | To program the trim value manually, irrespective of the fsm |

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|---|
| 11:7 | R/W | fixed_trim_ro | 5'd0 | Manual trim word |
| 6 | R/W | fixed_mode | 1'b0 | fixed mode |
| 5:4 | R/W | max_mode | 2'd2 | maximum mode it can go to |
| 3:0 | R/W | count_reset | 4'hF | Count reset value, count threshold will be doubler this value |

9.5.2 BG_SCDC_PROG_REG_1

| Bit | Access | Function | Default Value | Description |
|------------|---------------|------------------|----------------------|--|
| 21:19 | R/W | bg_r_ptat | 3'd2 | Bandgap voltage programming |
| 18:16 | R/W | bg_r | 3'd0 | Bandgap voltage programming |
| 15 | R/W | bg_en | 1'b0 | bg_en from spi |
| 14 | R/W | bg_sh_en | 1'b0 | bg_sh_en from spi |
| 13 | -- | Reserved | 0 | Reserved |
| 12:10 | R/W | ref_sel_dcdc | 3'd1 | DCDC output programming vref_1p1/vref_1p05 3'd0 - 1.1/1.05 3'd1 - 1.0/0.95 3'd2 - 0.9/0.85 3'd3 - 0.8/0.75 3'd4 - 1.05/1.0 3'd5 - 0.95/0.9 |
| 9:7 | R/W | ref_sel_lp_dcdc | 3'd1 | DCDC output programming in low power mode 3'd0 - 1.05 3'd1 - 1.0 3'd2 - 0.95 3'd3 - 0.9 3'd4 - 0.85 3'd5 - 0.8 |
| 6:5 | - | Reserved | 0 | Reserved |
| 4 | R/W | bod_clks_ptat_en | 1'b1 | 1 - To enable pstat currents to clocks and bod(cmp_npss) |
| 3 | R/W | an_perif_ptat_en | 1'b1 | 1 - To enable pstat currents to analog peripherals |
| 2:0 | R/W | ref_sel_PMU | 3'd0 | 3'd0 - 1.2V 3'd1 - 1.15V 3'd2 - 1.1V 3'd3 - 1.05V 3'd4 - 1.0V 3'd5 - 0.95V 3'd6 - 0.9V 3'd7 - 0.85V |

244.

9.5.3 BG_SCDC_PROG_REG_2

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|--|
| 21 | R/W | scdcdc_sel | 1'b0 | To switch to SCDCDC mode from LDO mode. 1 - SCDC mode 0 - LDO mode |
| 20 | R/W | testmode_0_en | 1'b0 | Enable for output on to BG_TESTMODE0 |

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------------|---------------|---|
| 19:18 | R/W | testmode_0_sel | 2'd0 | 2'd0: bg_sw_active 2'd1: scdcdc_sown 2'd2: scdcdc_lp_mode (sel_high_freq_ext_b) 2'd3: scdcdc_sel (To select ldo - scdcdc) |
| 17 | R/W | testmode_1_en | 1'b0 | To enable test mux for BG_TESTMODE1 |
| 16:15 | R/W | testmode_1_sel | 2'd0 | 2'd0: bg_sh_en 2'd1: scdcdc_up 2'd2: scdcdc_en (Enable for scdcdc block) 2'd3: scdcdc_lp_en (enable for 10uA LDO) |
| 14 | R/W | testmode_2_en | 1'd0 | To enable testmux for BG_TESTMODE2 |
| 13:11 | R/W | testmode_2_sel | 3'd0 | 3'd0: bg_en 3'd1: bg_comp_clk 3'd2: en_ldo_5m_b 3'd3: comp_clk 3'd4: scdcdc_conv_1b1 3'd5: scdcdc_conv_1b2 3'd6: scdcdc_conv_1b3 3'd7: 0 |
| 10:6 | R/W | trim_clamp_lp | 5'd1 | trim value lower clamp value when sel high freq_b is 1 |
| 5:1 | R/W | trim_clamp_hp | 5'd16 | trim value lower clamp value when sel high freq_b is 0 |
| 0 | R/W | scdcdc_soft_reset | 0 | soft reset signal for scdcdc fsm |

9.5.4 BG_LDO_REG

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|--|
| 21 | R/W | LDO_0P6_BYPASS | 1'b0 | bypass signal for DCDC1p1_lp_500uA |
| 20:18 | R/W | LDO_0P6_CTRL | 3'd2 | vref for DCDC1p1_lp_500uA 3'd0 - 0.8V 3'd1 - 0.75V 3'd2 - 0.7V 3'd3 - 0.65V 3'd4 - 0.6V 3'd5 - 0.55V |
| 17 | -- | Reserved | 0 | Reserved |
| 16 | R/W | LDO_0P6_LP_MODE | 1'b0 | enable low power mode, otherwise in high power mode |
| 15 | R/W | LDO_0P6_ENABLE | 1'b1 | enable digital LDO |
| 14:5 | -- | Reserved | 0 | Reserved |
| 4 | R/W | test_amux_en | 1'b0 | Enable analog mux to test reference voltages |

| Bit | Access | Function | Default Value | Description |
|-----|--------|---------------|---------------|--|
| 3:1 | R/W | test_amux_sel | 3'd0 | Select for analog mux 3'd0: vref_1p1 3'd1: vref_1p05 3'd2: vref_0p6 3'd3: vref_ulp 3'd4: vref_pmu |
| 0 | -- | Reserved | -- | Reserved |

10 Pad Configurations

10.1 General Description

There are total of 85 GPIOs present. The number of GPIOs available varies between different packages. For example, QMS package has 22 GPIO pins exposed, whereas BTS package has all GPIOs exposed. Please refer to GPIO available vs package table in the product datasheet for more details. Registers for GPIO pins that are not available on package are reserved. There are multiple processor sub-systems containing SZP (Secure Zone Processor), MCU HP (High Performance) and MCU ULP (Ultra Low Power) which share these common set of GPIO pads. These GPIO pads are controllable by either SZP, MCU HP or MCU ULP. PAD selection register has to be programmed to control the PAD behavior for each GPIO.

In addition to these, there are few High Speed PADs used for DDR-Flash Interface labeled as DDR Pads. The configuration for these PADs are controlled by the MCU HP (High Performance) Domain.

The list below provides the registers to be configured for accessing any of the GPIO pads.

- PAD selection Register.
- PAD configuration Register.
- GPIO mode Register.

More details about pad selection and pad configuration are described below.

10.2 Features

The 85 GPIOs are divided into 64 SoC GPIOs, 16 ULP GPIOs and 5 UULP (Ultra ULP) Vbat GPIOs. The SoC GPIOs are available only in PS4/PS3 power states (as described in [Power Architecture Section](#)) whereas ULP GPIOs are available in all the power states except sleep modes. The UULP Vbat GPIOs are available in all power states.

The SoC GPIOs and ULP GPIOs PAD are programmable, multi-voltage (1.8V, 2.5V, 2.8V, 3.0V, 3.3V) general purpose, bi-directional I/O buffer with a selectable LVCMOS (Low Voltage CMOS) input or LVCMOS Schmitt trigger input and programmable pull-up / pull-down. In the full-drive mode, this buffer can operate in excess of 100MHz frequency with 15pF external load and 125 MHz with 10pF load, but actual frequency is load and system dependent. A maximum of 200MHz can be achieved under small capacitive loads.

The following PAD configurations can be controlled by software for SoC GPIOs and ULP GPIOs.

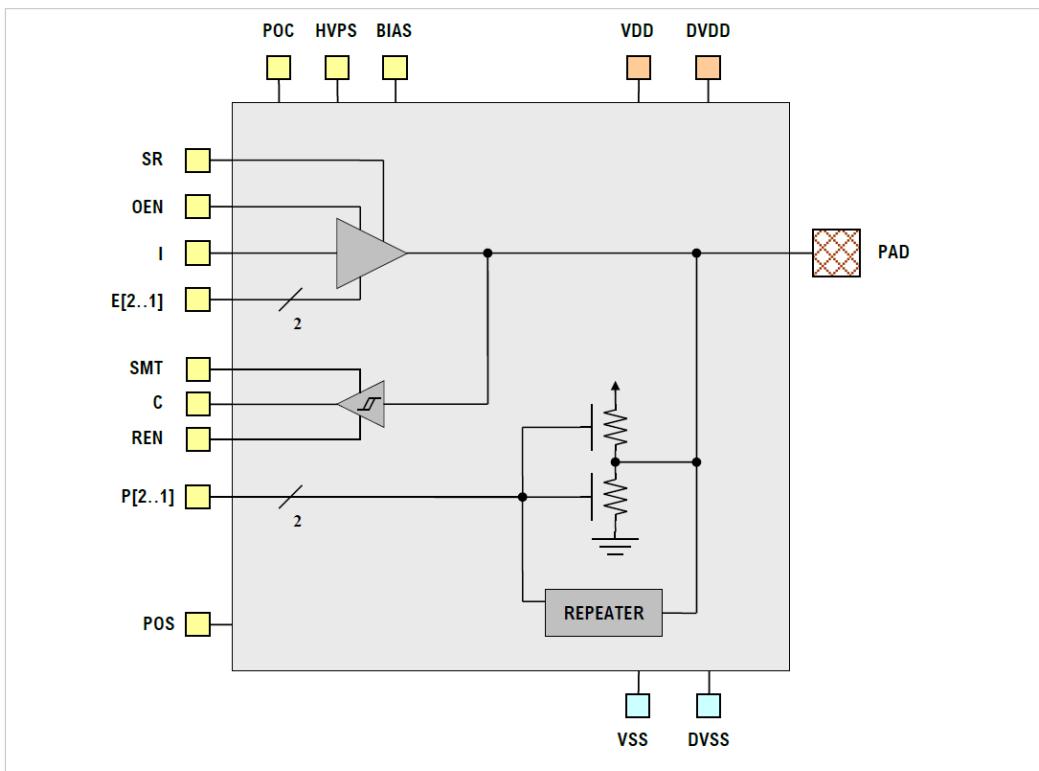
- Bi-directional IO capability
- Multi-voltage DVDD capability (1.8V, 2.5V, 2.8V, 3.0V, 3.3V)
- Power-on-Start (POS) capable
- Optimized for EMC (low di/dt switching supply noise) with SSO (Simultaneous Switching Output) factor of 8
- Four (4) Programmable output drive strengths (rated 2mA, 4mA, 8mA, and 12mA)
- Selectable output slew-rate (slow / fast)
- Open drain output mode (Logic low or high on input and use OEN as data input)
- LVCMOS/LVTTL compatible input with selectable hysteresis
- Programmable input options (pull-up, pull-down, repeater, or plain input)
- No power sequence requirements, I/Os are tri-stated when core power is not valid (POC control). These are tri-stated even if the system is under reset or in the deep sleep power state.

The following PAD configurations can be controlled by software for UULP Vbat GPIOs.

- Bi-directional IO capability
- Multi-voltage DVDD capability (1.8V, 2.5V, 2.8V, 3.0V, 3.3V)

10.3 Functional Description

The figure below depicts the PAD model used for SoC-GPIOS and ULP-GPIOS.



6 .Pad Model

10.3.1 PAD Description

Ports

| Port Name | Direction | Description |
|-----------|-----------|---|
| PAD | INOUT | Pad pin (Bond pad) |
| C | OUTPUT | Data output to the core. The value on PAD will be assigned to C when REN is 1. |
| I | INPUT | Data input from the core logic. This value is assigned to PAD when OEN is 0. |
| OEN | INPUT | Active low output driver enable. 1 - Driver is Disabled. 0 - Driver is Enabled. |

| Port Name | Direction | Description |
|-----------|-----------|---|
| P[2..1] | INPUT | Driver disabled state control. 0 - Hi-Z 1 - Pull-up 2 - Pull-down 3 - Repeater |
| E[2..1] | INPUT | Drive strength selector. 0: 2 mA 1: 4 mA 2: 8 mA 3: 12 mA |
| SR | INPUT | Slew Rate Control. 0 - Slow (half frequency) 1 - Fast |
| REN | INPUT | Active High Receiver Enable. 0 - Receiver disabled. 1 - Receiver enabled |
| SMT | INPUT | Active High Schmitt Trigger (Hysteresis) select. 0 - No hysteresis |
| POS | INPUT | Power-on-Start enable. 1 - Enables Active pull-down for invalid power. 0 - Disables Active pull-down capability. When one of the power supplies is invalid and active-high POS is set to 1, PAD is pulled to weak 0. When POS is set to 0, PAD remains in a high-Z state |
| POC | INPUT | Power-on control |
| HVPS | INPUT | High voltage power supply signal |
| BIAS | INPUT | Bias signal |
| VDD | INPUT | Core VDD |
| VSS | INPUT | Core VSS |
| DVDD | INPUT | I/O VDD |
| DVSS | INPUT | I/O VSS |

245 .PAD Port Description

Transmit (OEN) and Driver Disabled State Control (P2, P1) Truth Table

| Inputs | | | | Output | |
|--------|---------|---------|---|--------|--|
| OEN | P2(MSB) | P1(LSB) | I | IO | |
| | | | | | |

| Inputs | | | | Output |
|--------|---|---|---|-----------------------|
| 0 | - | - | 0 | 0 |
| | | | 1 | 1 |
| 1 | 0 | 0 | - | Z(Normal operation) |
| | 0 | 1 | - | weak 1 (Pull-up) |
| | 1 | 0 | - | weak 0 (Pull-down) |
| | 1 | 1 | - | Repeater (Bus keeper) |

246 . Truth Table for OEN, P1, P2

Receiver Enable (REN) Truth Table

| Inputs | | Output |
|--------|-------|--------|
| REN | PAD C | |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | - | 0 |

247 . Truth Table for REN

10.3.2 Programming Sequence

The SoC GPIOs (GPIO_0 to GPIO_63) and ULP GPIOs (ULP_GPIO_0 to ULP_GPIO_15) are shared between SZP, MCU HP and MCU ULP. The GPIOs configuration and functionality can be independently controlled by them. The UULP Vbat GPIOs (UULP_VBAT_GPIO_0 to UULP_VBAT_GPIO_5) are controlled by MCU ULP.

PAD Configuration

The PAD Configuration for each GPIO can be done through SZP, MCU HP or MCU ULP.

- GPIO_21 to GPIO_24 and GPIO_38 to GPIO_41 are configured by MCU HP
- The remaining SoC GPIOs are shared and are configured by either MCU HP or SZP
- ULP GPIOs are configured by MCU ULP
- The PADs are configured through [PAD_CONFIG_REG_n](#) (n=0:63), [ULP_PAD_CONFIG_REG0](#), [ULP_PAD_CONFIG_REG1](#) and [ULP_PAD_CONFIG_REG2](#) Registers
- The UULP Vbat GPIOs are configured through [UULP_VBAT_GPIOn_CONFIG_REG](#) Register (n=0:4)

At power up, all shared GPIOs are controlled by SZP. The following control bits needs to be programmed corresponding to the particular PAD such that it can be configured by MCU HP.

- SZP_MCUHP_GPIO_CTRL1[15:0] control bits are configured through [MCUHP_PAD_SELECTION](#) Register. The contents of this register are retained during sleep.
- SZP_MCUHP_GPIO_CTRL2 control bits accessible by [MEM_GPIO_ACCESS_CTRL_SET](#) and [MEM_GPIO_ACCESS_CTRL_CLEAR](#) Register. The contents of this register are retained during sleep.

The table below indicates the PAD Selection control for each GPIO.

| GPIO Index | Selection Control Signal |
|------------|--------------------------|
| GPIO_0 | SZP_MCUHP_GPIO_CTRL1[0] |
| GPIO_1 | SZP_MCUHP_GPIO_CTRL1[0] |
| GPIO_2 | SZP_MCUHP_GPIO_CTRL1[0] |
| GPIO_3 | SZP_MCUHP_GPIO_CTRL1[0] |
| GPIO_4 | SZP_MCUHP_GPIO_CTRL1[0] |

| GPIO Index | Selection Control Signal |
|-------------------|---------------------------------|
| GPIO_5 | SZP_MCUHP_GPIO_CTRL1[0] |
| GPIO_6 | SZP_MCUHP_GPIO_CTRL1[1] |
| GPIO_7 | SZP_MCUHP_GPIO_CTRL1[1] |
| GPIO_8 | SZP_MCUHP_GPIO_CTRL1[1] |
| GPIO_9 | SZP_MCUHP_GPIO_CTRL1[1] |
| GPIO_10 | SZP_MCUHP_GPIO_CTRL1[2] |
| GPIO_11 | SZP_MCUHP_GPIO_CTRL1[2] |
| GPIO_12 | SZP_MCUHP_GPIO_CTRL1[3] |
| GPIO_13 | SZP_MCUHP_GPIO_CTRL1[4] |
| GPIO_14 | SZP_MCUHP_GPIO_CTRL1[4] |
| GPIO_15 | SZP_MCUHP_GPIO_CTRL1[5] |
| GPIO_16 | SZP_MCUHP_GPIO_CTRL1[5] |
| GPIO_17 | SZP_MCUHP_GPIO_CTRL1[5] |
| GPIO_18 | SZP_MCUHP_GPIO_CTRL1[6] |
| GPIO_19 | SZP_MCUHP_GPIO_CTRL1[6] |
| GPIO_20 | SZP_MCUHP_GPIO_CTRL1[7] |
| GPIO_21 | Controlled by MCU HP |
| GPIO_22 | Controlled by MCU HP |
| GPIO_23 | Controlled by MCU HP |
| GPIO_24 | Controlled by MCU HP |
| GPIO_25 | SZP_MCUHP_GPIO_CTRL2 |
| GPIO_26 | SZP_MCUHP_GPIO_CTRL2 |
| GPIO_27 | SZP_MCUHP_GPIO_CTRL2 |
| GPIO_28 | SZP_MCUHP_GPIO_CTRL2 |
| GPIO_29 | SZP_MCUHP_GPIO_CTRL2 |
| GPIO_30 | SZP_MCUHP_GPIO_CTRL2 |
| GPIO_31 | SZP_MCUHP_GPIO_CTRL1[8] |
| GPIO_32 | SZP_MCUHP_GPIO_CTRL1[8] |
| GPIO_33 | SZP_MCUHP_GPIO_CTRL1[8] |
| GPIO_34 | SZP_MCUHP_GPIO_CTRL1[8] |
| GPIO_35 | SZP_MCUHP_GPIO_CTRL1[9] |
| GPIO_36 | SZP_MCUHP_GPIO_CTRL1[9] |
| GPIO_37 | SZP_MCUHP_GPIO_CTRL1[9] |
| GPIO_38 | Controlled by MCU HP |
| GPIO_39 | Controlled by MCU HP |
| GPIO_40 | Controlled by MCU HP |
| GPIO_41 | Controlled by MCU HP |
| GPIO_42 | SZP_MCUHP_GPIO_CTRL1[10] |
| GPIO_43 | SZP_MCUHP_GPIO_CTRL1[10] |
| GPIO_44 | SZP_MCUHP_GPIO_CTRL1[10] |
| GPIO_45 | SZP_MCUHP_GPIO_CTRL1[10] |
| GPIO_46 | SZP_MCUHP_GPIO_CTRL1[11] |
| GPIO_47 | SZP_MCUHP_GPIO_CTRL1[11] |
| GPIO_48 | SZP_MCUHP_GPIO_CTRL1[11] |
| GPIO_49 | SZP_MCUHP_GPIO_CTRL1[12] |
| GPIO_50 | SZP_MCUHP_GPIO_CTRL1[12] |
| GPIO_51 | SZP_MCUHP_GPIO_CTRL1[12] |
| GPIO_52 | SZP_MCUHP_GPIO_CTRL1[13] |

| GPIO Index | Selection Control Signal |
|-------------------|---------------------------------|
| GPIO_53 | SZP_MCUHP_GPIO_CTRL1[13] |
| GPIO_54 | SZP_MCUHP_GPIO_CTRL1[13] |
| GPIO_55 | SZP_MCUHP_GPIO_CTRL1[13] |
| GPIO_56 | SZP_MCUHP_GPIO_CTRL1[14] |
| GPIO_57 | SZP_MCUHP_GPIO_CTRL1[14] |
| GPIO_58 | SZP_MCUHP_GPIO_CTRL1[15] |
| GPIO_59 | SZP_MCUHP_GPIO_CTRL1[15] |
| GPIO_60 | SZP_MCUHP_GPIO_CTRL1[15] |
| GPIO_61 | SZP_MCUHP_GPIO_CTRL1[15] |
| GPIO_62 | SZP_MCUHP_GPIO_CTRL1[15] |
| GPIO_63 | SZP_MCUHP_GPIO_CTRL1[15] |
| ULP_GPIO_0 | Controlled by MCU ULP |
| ULP_GPIO_1 | Controlled by MCU ULP |
| ULP_GPIO_2 | Controlled by MCU ULP |
| ULP_GPIO_3 | Controlled by MCU ULP |
| ULP_GPIO_4 | Controlled by MCU ULP |
| ULP_GPIO_5 | Controlled by MCU ULP |
| ULP_GPIO_6 | Controlled by MCU ULP |
| ULP_GPIO_7 | Controlled by MCU ULP |
| ULP_GPIO_8 | Controlled by MCU ULP |
| ULP_GPIO_9 | Controlled by MCU ULP |
| ULP_GPIO_10 | Controlled by MCU ULP |
| ULP_GPIO_11 | Controlled by MCU ULP |
| ULP_GPIO_12 | Controlled by MCU ULP |
| ULP_GPIO_13 | Controlled by MCU ULP |
| ULP_GPIO_14 | Controlled by MCU ULP |
| ULP_GPIO_15 | Controlled by MCU ULP |
| UULP_VBAT_GPIO_0 | Controlled by MCU ULP |
| UULP_VBAT_GPIO_1 | Controlled by MCU ULP |
| UULP_VBAT_GPIO_2 | Controlled by MCU ULP |
| UULP_VBAT_GPIO_3 | Controlled by MCU ULP |
| UULP_VBAT_GPIO_4 | Controlled by MCU ULP |

248 . PAD Configuration Control Signals

GPIO Register Programming

The 80 general-purpose I/O (GPIO) pins are used in generating and capturing application-specific input and output signals. Each pin can be programmed as an output or as an input port for various functions. GPIO pins may have alternate input and output functions. A pin may be controlled by software or as an alternate function pin, but not as both at the same time.

Functionlity for all the GPIOs are shared between SZP, MCU HP and MCU ULP. Each GPIO can be programmed through respective GPIO registers after configuring the GPIO mode as per the functional usage.

- GPIOs 0:63 are controlled from the MCU HP GPIO Registers
- ULP GPIOs 0:15 are controlled from the MCU ULP GPIO Registers

Each GPIO pin has a register that controls the behavior of the pin. Information about the pin, like the mode, direction of the pin and type of signal detection required has to be programmed to this register. The GPIO mode for all GPIO pins are configured as per the Reset values table described below

MCU HP GPIO Registers

The GPIO programming for GPIO_n (n=0:63) which are controlled by MCU HP are programmed as described in [Enhanced GPIO](#) Section of MCU APB Peripherals Section.

MCU ULP GPIO Registers

The GPIO mode for ULP_GPIO_n (n=0:15) which are controlled by MCU ULP are programmed as described in [ULP Enhanced GPIO](#) Section of MCU ULP Peripherals Section.

MCU UULP Vbat GPIO Registers

The configuration of UULP_VBAT_GPIO_n (n=0:4) which are controlled by MCU ULP can be done through the [UULP_VBAT_GPIOOn_CONFIG_REG](#)

10.3.3 PAD Configuration and GPIO Mode Reset Values

The table below indicates the Reset values for the PAD configurations and GPIO Modes of each GPIO.

| GPPAD | P2 | P1 | SR | REN | SMT | POS | E2 | E1 | GPIO_MODE |
|---------|----|----|----|-----|-----|-----|----|----|-----------|
| GPIO_0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_6 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_7 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_8 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_9 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_10 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_11 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_12 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_13 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_14 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_15 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_16 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 15 |
| GPIO_17 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_18 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_19 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_20 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_21 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_22 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_23 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_24 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_25 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| GPIO_26 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| GPIO_27 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| GPIO_28 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| GPIO_29 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| GPPAD | P2 | P1 | SR | REN | SMT | POS | E2 | E1 | GPIO_MODE |
|-------------|----|----|----|-----|-----|-----|----|----|-----------|
| GPIO_30 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| GPIO_31 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 15 |
| GPIO_32 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 15 |
| GPIO_33 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 15 |
| GPIO_34 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 15 |
| GPIO_35 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_36 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_37 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_38 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_39 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_40 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_41 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GPIO_42 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_43 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_44 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_45 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_46 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_47 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_48 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_49 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_50 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_51 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_52 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_53 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_54 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_55 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_56 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_57 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_58 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_59 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_60 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_61 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_62 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| GPIO_63 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 15 |
| ULP_GPIO_0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| GPPAD | P2 | P1 | SR | REN | SMT | POS | E2 | E1 | GPIO_MODE |
|-------------------|----|----|----|-----|-----|-----|----|----|-----------|
| ULP_GPIO_14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| ULP_GPIO_15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| UULP_VBAT_GPIO_0- | - | - | 0 | - | - | - | - | - | 1 |
| UULP_VBAT_GPIO_1- | - | - | 0 | - | - | - | - | - | 1 |
| UULP_VBAT_GPIO_2- | - | - | 0 | - | - | - | - | - | 1 |
| UULP_VBAT_GPIO_3- | - | - | 0 | - | - | - | - | - | 0 |
| UULP_VBAT_GPIO_4- | - | - | 0 | - | - | - | - | - | 0 |

249 . PAD Configuration and GPIO Mode Reset Values

10.4 DDR PADS

10.4.1 Overview

These PADS are used for DDR-Flash Interface. These Pads are controlled only by MCU HP.

DDR PADS are:

- DDR_DATA_n (n=0:7) - Data Signals
- DDR_CSN - Chip Select Signal
- DDR_CLK - Clock Signal
- DDR_DQS - DQS Select Signal
- DDR_WP - Write Protect Signal

10.5 Register Summary

10.5.1 PAD Selection Registers

Base Address: 0x4130_0000

| Register Name | Offset | Description |
|-----------------------------|--------|---|
| MEM_GPIO_ACCESS_CTR_L_SET | 0x000 | Indicates the PAD Configuration Control for GPIO_25-GPIO_30. |
| MEM_GPIO_ACCESS_CTR_L_CLEAR | 0x004 | Indicates the PAD Configuration Control for GPIO_25-GPIO_30. |
| MCUHP_PAD_SELECTION | 0x610 | Indicates the PAD Configuration Control for shared GPIO's except for GPIO_25-GPIO_30. |

250 . PAD Control Registers Summary

10.5.2 MCU HP GPIO PAD Configuration Registers

Base Address: 0x4600_6000

| Register Name | Offset | Description |
|------------------|-----------|---|
| PAD_CONFIG_REG_n | 0x0 + 4*n | PAD Configuration Register for GPIO_n; n = 0,1,2, 63 |

251 . MCUHP PAD Configuration Registers Summary

10.5.3 MCU ULP GPIO PAD Configuration Registers

Base Address: 0x2404_A000

| Register Name | Offset | Description |
|---------------------|--------|--|
| ULP_PAD_CONFIG_REG0 | 0x00 | |
| ULP_PAD_CONFIG_REG1 | 0x04 | |
| ULP_PAD_CONFIG_REG2 | 0x08 | PAD Configuration Registers for ULP GPIOs (ULP_GPIO_0 to ULP_GPIO_15) |

252 . MCUULP PAD Configuration Registers Summary

10.5.4 MCU UULP Vbat GPIO PAD Configuration Registers

Base Address: 0x2404_861C

| Register Name | Offset | Description |
|----------------------------|-----------|---|
| UULP_VBAT_GPIOn_CONFIG_REG | 0x0 + 4*n | PAD Configuration Registers for UULP Vbat GPIOs UULP_VBAT_GPIO (n=0:4) |

253 . MCU UULP Vbat GPIO Configuration Registers

10.6 Register Description

10.6.1 MCUHP_PAD_SELECTION

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------|-------------|--|
| 31:21 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15:0 | RW | SZP_MCUHP_GPIO_C TRL1 | 0 | PAD Configuration Controls between NWP and MCU HP. Writing 1 to a particular bit enables the MCU HP to configure the corresponding PADs Writing 0 to a particular bit enables the SZP to configure the corresponding PADs Details of the PADs corresponding to each bit are described in GPIO Controls table above. |

254 . MCUHP_PAD_SELECTION Description

10.6.2 MEM_GPIO_ACCESS_CTRL_SET

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------------|-------------|--|
| 31:6 | - | Reserved | - | It is recommended to write these bits to 0. |
| 5 | RW | SZP_MCUHP_GPIO_C TRL2 | 1 | Writing 1 to this enables SZP to configure the GPIO_25 to GPIO_30 Writing 0 to this has no effect. |
| 4:0 | - | Reserved | - | It is recommended to write these bits to 0. |

255 . MEM_GPIO_ACCESS_CTRL_SET Description

10.6.3 MEM_GPIO_ACCESS_CTRL_CLEAR

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------|-------------|--|
| 31:6 | - | Reserved | - | It is recommended to write these bits to 0. |
| 5 | RW | SZP MCUHP GPIO_C TRL2 | 1 | Writing 1 to this enables MCU HP to configure the GPIO_25 to GPIO_30 Writing 0 to this has no effect. |
| 4:0 | - | Reserved | - | It is recommended to write these bits to 0. |

256 .MEM_GPIO_ACCESS_CTRL_CLEAR Description

10.6.4 PAD_CONFIG_REG_n

The Reset values for these registers are already provided in the [PAD Configuration and GPIO Mode Reset Values Table](#) above

| Bit | Access | Function | Description |
|------|--------|---------------|--|
| 31:8 | - | Reserved | It is recommended to write these bits to 0. |
| 7 | RW | PADCONFIG_P2 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater |
| 6 | RW | PADCONFIG_P1 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater |
| 5 | RW | PADCONFIG_SR | Slew Rate Control ; SR = 0 – Slow (half frequency); SR = 1 – Fast |
| 4 | RW | PADCONFIG_REN | Active high receiver enable ; REN = 0 – Receiver disabled, C driven to 0 - REN = 1 – Receiver enabled |
| 3 | RW | PADCONFIG_SMT | Active high Schmitt trigger (Hysteresis) select; SMT=0 – No hysteresis; Default value for reset is 1'b1 and others is 1'b0 |
| 2 | RW | PADCONFIG_POS | Power-on-Start enable; POS = 1 – Enables active pull-down for invalid power; POS = 0 – Active pull-down capability disabled . When one of the power supplies is invalid and active-high POS is set to 1, PAD is pulled to weak 0. When POS is set to 0, PAD remains in a high-Z state. : Default 0 |
| 1 | RW | PADCONFIG_EE2 | EE[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 mA |
| 0 | RW | PADCONFIG_EE1 | EE[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 mA |

257 .PAD_CONFIG_REG_n Description

10.6.5 ULP_PAD_CONFIG_REG0

| BIT | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15 | RW | PADCONFIG_P2_2 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_4 - ULP_GPIO_7 |
| 14 | RW | PADCONFIG_P1_2 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_4 - ULP_GPIO_7 |
| 13 | RW | PADCONFIG_SR_2 | 0 | Slew Rate Control ; SR = 0 – Slow (half frequency); SR = 1 – Fast for ULP_GPIO_4 - ULP_GPIO_7 |
| 12 | - | Reserved | - | It is recommended to write these bits to 0. |
| 11 | RW | PADCONFIG_SMT_2 | 0 | Active high Schmitt trigger (Hysteresis) select; SMT=0 – No hysteresis; Default value for reset is 1'b1 and others is 1'b0 for ULP_GPIO_4 - ULP_GPIO_7 |
| 10 | RW | PADCONFIG_POS_2 | 0 | Power-on-Start enable; POS = 1 – Enables active pull-down for invalid power; POS = 0 – Active pull-down capability disabled . When one of the power supplies is invalid and active-high POS is set to 1, PAD is pulled to weak 0. When POS is set to 0, PAD remains in a high-Z state. : Default 0 for ULP_GPIO_4 - ULP_GPIO_7 |
| 9 | RW | PADCONFIG_E2_2 | 0 | E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma for ULP_GPIO_4 - ULP_GPIO_7 |
| 8 | RW | PADCONFIG_E1_2 | 1 | E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma for ULP_GPIO_4 - ULP_GPIO_7 |
| 7 | RW | PADCONFIG_P2_1 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_0 - ULP_GPIO_3 |

| BIT | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|---|
| 6 | RW | PADCONFIG_P1_1 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_0 - ULP_GPIO_3 |
| 5 | RW | PADCONFIG_SR_1 | 0 | Slew Rate Control ; SR = 0 – Slow (half frequency); SR = 1 – Fast for ULP_GPIO_0 - ULP_GPIO_3 |
| 4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | RW | PADCONFIG_SMT_1 | 0 | Active high Schmitt trigger (Hysteresis) select; SMT=0 – No hysteresis; Default value for reset is 1'b1 and others is 1'b0 for ULP_GPIO_0 - ULP_GPIO_3 |
| 2 | RW | PADCONFIG_POS_1 | 0 | Power-on-Start enable; POS = 1 – Enables active pull-down for invalid power; POS = 0 – Active pull-down capability disabled . When one of the power supplies is invalid and active-high POS is set to 1, PAD is pulled to weak 0. When POS is set to 0, PAD remains in a high-Z state. : Default 0 for ULP_GPIO_0 - ULP_GPIO_3 |
| 1 | RW | PADCONFIG_E2_1 | 0 | E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma for ULP_GPIO_0 - ULP_GPIO_3 |
| 0 | RW | PADCONFIG_E1_1 | 1 | E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma for ULP_GPIO_0 - ULP_GPIO_3 |

258 . MCUULP_PAD_CONFIG_REG0 Description

10.6.6 ULP_PAD_CONFIG_REG1

| BIT | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|--|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |
| 15 | RW | PADCONFIG_P2_2 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_12 - ULP_GPIO_15 |
| 14 | RW | PADCONFIG_P1_2 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_12 - ULP_GPIO_15 |

| BIT | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 13 | RW | PADCONFIG_SR_2 | 0 | Slew Rate Control ; SR = 0 – Slow (half frequency); SR = 1 – Fast for ULP_GPIO_12 - ULP_GPIO_15 |
| 12 | - | Reserved | - | It is recommended to write these bits to 0. |
| 11 | RW | PADCONFIG_SMT_2 | 0 | Active high Schmitt trigger (Hysteresis) select; SMT=0 – No hysteresis; Default value for reset is 1'b1 and others is 1'b0 for ULP_GPIO_12 - ULP_GPIO_15 |
| 10 | RW | PADCONFIG_POS_2 | 0 | Power-on-Start enable; POS = 1 – Enables active pull-down for invalid power; POS = 0 – Active pull-down capability disabled . When one of the power supplies is invalid and active-high POS is set to 1, PAD is pulled to weak 0. When POS is set to 0, PAD remains in a high-Z state. : Default 0 for ULP_GPIO_12 - ULP_GPIO_15 |
| 9 | RW | PADCONFIG_E2_2 | 0 | E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma for ULP_GPIO_12 - ULP_GPIO_15 |
| 8 | RW | PADCONFIG_E1_2 | 1 | E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma for ULP_GPIO_12 - ULP_GPIO_15 |
| 7 | RW | PADCONFIG_P2_1 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_8 - ULP_GPIO_11 |
| 6 | RW | PADCONFIG_P1_1 | 0 | P[2,1] – Driver disabled state control, 0-Hi-Z / 1-Pull-up / 2-Pull-down / 3-Repeater for ULP_GPIO_8 - ULP_GPIO_11 |
| 5 | RW | PADCONFIG_SR_1 | 0 | Slew Rate Control ; SR = 0 – Slow (half frequency); SR = 1 – Fast for ULP_GPIO_8 - ULP_GPIO_11 |
| 4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | RW | PADCONFIG_SMT_1 | 0 | Active high Schmitt trigger (Hysteresis) select; SMT=0 – No hysteresis; Default value for reset is 1'b1 and others is 1'b0 for ULP_GPIO_8 - ULP_GPIO_11 |

| BIT | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|---|
| 2 | RW | PADCONFIG_POS_1 | 0 | <p>Power-on-Start enable;</p> <p>POS = 1 – Enables active pull-down for invalid power;</p> <p>POS = 0 – Active pull-down capability disabled .</p> <p>When one of the power supplies is invalid and active-high POS is set to 1, PAD is pulled to weak 0.</p> <p>When POS is set to 0, PAD remains in a high-Z state. : Default 0</p> <p>for ULP_GPIO_8 - ULP_GPIO_11</p> |
| 1 | RW | PADCONFIG_E2_1 | 0 | <p>E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma</p> <p>for ULP_GPIO_8 - ULP_GPIO_11</p> |
| 0 | RW | PADCONFIG_E1_1 | 1 | <p>E[2,1] – Drive strength selector, 0-2 mA / 1-4 mA / 2-8 mA / 3-12 ma</p> <p>for ULP_GPIO_8 - ULP_GPIO_11</p> |

259 . MCUULP_PAD_CONFIG_REG1 Description

10.6.7 ULP_PAD_CONFIG_REG2

| BIT | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|--|
| 31:16 | - | Reserved | 0 | It is recommended to write these bits to 0. |
| 15:0 | RW | PADCONFIG_REN | 0 | <p>Active high receiver enable ; REN = 0 – Receiver disabled, C driven to 0 - REN = 1 – Receiver enabled</p> <p>for ULP_GPIO_15:ULP_GPIO_0</p> |

260 . MCUULP_PAD_CONFIG_REG2 Description

10.6.8 UULP_VBAT_GPIOOn_CONFIG_REG

| BIT | Access | Function | Reset Value | Description |
|------|--------|---------------|-------------|--|
| 31:9 | - | Reserved | - | It is recommended to write these bits to 0. |
| 8 | RW | GPIO_POLARITY | 0 | <p>Indicates the polarity of the UULP_VBAT_GPIO to be considered when used as a Wakeup source from any of the Sleep States as described in Power Architecture</p> <p>1 - When GPIO Input is High</p> <p>0 - When GPIO Input is Low</p> |
| 7:6 | - | Reserved | - | It is recommended to write these bits to 0. |
| 5 | RW | GPIO_OUTPUT | 0 | Indicates the value to be driven on the PAD when configured to OUTPUT mode (GPIO Mode=0) for UULP_VBAT_GPIO_n (n=0:4) |

| BIT | Access | Function | Reset Value | Description |
|-----|--------|-----------|-------------|---|
| 4 | RW | GPIO_OEN | 1 | Indicates the direction of the PAD for UULP_VBAT_GPIO_n (n=0:4) if configured to GPIO mode = 0 0 - Output 1 - Input |
| 3 | RW | GPIO_REN | 0 | Enables the Receiver of the PAD for UULP_VBAT_GPIO_n (n=0:4) 0 - Receiver Disabled 1 - Receiver Enabled |
| 2:0 | RW | GPIO_MODE | - | Indicates the GPIO Mode for UULP_VBAT_GPIO_n (n=0:4) |

261 .UULP_VBAT_GPIOOn_CONFIG_REG Register Description

10.6.9 Register Summary

| Address | Register Name |
|-------------------------------|--|
| base address | 0x4600_4000 |
| addr[13:12] = 10 | |
| addr[11:0] = 0x0 + 4*n | PAD_CONFIG for Pad Index n = 0,1..7 for qspi_ddr_data_0,1..7 n = 8 for qspi_ddr_csn n = 9 for qspi_ddr_clk n = 10 for qspi_ddr_dqs n = 11 for smih_wp |

10.6.10 Register Description

| PAD_CONFIG_REG_n (n = 0) | | | | |
|---------------------------------|--------|----------------|---------------|---|
| Address: 0x0 + 4*n | | | | |
| Bit | Access | Function | Default Value | Description |
| 31:8 | - | Reserved | - | It is recommended to write these bits to 0. |
| 7 | RW | padconfig_RXEN | 1 | Active-low receiver enable |
| 6 | RW | padconfig_PD | 1 | Active-high receiver power down signal. This is applicable to all the DDR PADS. |

PAD_CONFIG_REG_n (n = 0)

Address: 0x0 + 4*n

| Bit | Access | Function | Default Value | Description |
|-----|--------|-----------------|---------------|---|
| 5 | RW | padconfig_DSEL | 0 | DDR2 drive select signal 0 - Full drive strength (Class II) 1 - 60% drive strength (Class I) |
| 4 | RW | padconfig_MODE | 0 | Mode select signal. This is applicable to all the DDR PADS. 0 – selects DDR2 mode 1 – selects DDR3 mode |
| 3 | RW | padconfig_ODTEN | 0 | Active-high ODT enable |
| 2 | RW | padconfig_ODT_2 | 0 | On-Die-Termination select pins |
| 1 | RW | padconfig_ODT_1 | 0 | On-Die-Termination select pins |
| 0 | RW | padconfig_ODT_0 | 0 | On-Die-Termination select pins |

PAD_CONFIG_REG_n (n = 1,2,...,11)

Address: 0x0 + 4*n

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------|---------------|--|
| 31:8 | - | Reserved | - | It is recommended to write these bits to 0. |
| 7 | RW | PADCONFIG_RXEN | 1 | Active-low receiver enable |
| 6 | - | Reserved | - | It is recommended to write these bits to 0. |
| 5 | RW | PADCONFIG_DSEL | 0 | DDR2 drive select signal 0 - Full drive strength (Class II) 1 - 60% drive strength (Class I) |
| 4 | - | Reserved | - | It is recommended to write these bits to 0. |

PAD_CONFIG_REG_n (n = 1,2,...,11)

Address: 0x0 + 4*n

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|--------------------------------|
| 3 | RW | PADCONFIG_ODTEN | 0 | Active-high ODT enable |
| 2 | RW | PADCONFIG_ODT_2 | 0 | On-Die-Termination select pins |
| 1 | RW | PADCONFIG_ODT_1 | 0 | On-Die-Termination select pins |
| 0 | RW | PADCONFIG_ODT_0 | 0 | On-Die-Termination select pins |

11 Pin MUX

11.1 Overview

The Pin MUX block supports many peripherals including digital and analog like I2C, I2S, UART, SPI, Ethernet, SDIO, SDMEM, PWM, QEI, CAN etc. However number of pins that can be exposed out is limited. This pin multiplexing module solves problem of accommodating all these multiple peripherals in smallest package without compromising any features. It multiplexes different functions on same I/O pins. Each I/O pin supports multiple peripheral functions. If a I/O pin is used for a peripheral function, it cannot be used as GPIO. At a time only one function can be used. The reset value for the GPIO mode of each of the GPIOs are provided in PAD Configurations section. The configuration of GPIO Mode is described in PAD Configurations section. For more details, refer to the chapter [Pad Configurations](#).

11.2 Functional Description

The following sections describe the Pin Multiplexing on different GPIOs (SoC GPIOs, ULP GPIOs and UULP VBAT GPIOs). This also includes the Analog functions mapped on these GPIOs.

11.2.1 SoC GPIOs

The SoC GPIOs below (GPIO_6 to GPIO_57) are available in the normal mode of operation (Power-states 4 and 3) as described in [Power Architecture](#).

Each of these GPIO's Pin function is configured through a 4-bit GPIO Mode of GPIO_CONFIG_REG_n(n=0:63) as described in [Enhanced-GPIO](#) Section of MCU APB Peripherals.

| GPIO | GPIO Mode = 0 | GPIO Mode = 1 | GPIO Mode = 2 | GPIO Mode = 3 | GPIO Mode = 4 | GPIO Mode = 5 | GPIO Mode = 6 | GPIO Mode = 7 | GPIO Mode = 8 | GPIO Mode = 9 | GPIO Mode = 10 | GPIO Mode = 14 |
|---------|---------------|---------------|-------------------|------------------------|------------------------|---------------|------------------|---------------------|---------------|----------------|----------------|----------------|
| GPIO_6 | GPIO_6 | SIO_0 | UART2_T_SSI_MST_X | I2C1_SD_A | I2S_2CH_DOUT_0 | SCT_OU | QEI_IDX | M4SS_S_MIH_CD_A_4_N | CCI_DAT | M4SS_Q_SPI_CLK | RMII_TX_D1 | |
| GPIO_7 | GPIO_7 | SIO_1 | UART2_R_SSI_MST_X | I2C1_SCL_I2S_2CH_DATA1 | I2S_2CH_CLK | SCT_OU_T_1 | QEI_PHB | M4SS_S_MIH_WP_T_1 | SCT_OU | M4SS_Q_SPI_CSN | RMII_TX_D0 | |
| GPIO_8 | GPIO_8 | SIO_2 | USART1_RX | USART1_SSI_MST_CLK | | SCT_OU_T_2 | QEI_PHA_PWM_1L | CCI_DAT_A_6 | M4SS_Q_SPI_D0 | RMII_RXE_F_CLK | | |
| GPIO_9 | GPIO_9 | SIO_3 | USART1_TX | USART1_SSI_MST_CS0 | SDMEM_PRESENTT_3 | SCT_OU | QEI_DIR_PWM_1H | CCI_DAT_A_7 | M4SS_Q_SPI_D1 | RMII_TXE_N | | |
| GPIO_10 | GPIO_10 | USART1_IR_TX | USART1_CLK | USART1_SSI_MST_CS1 | I2C2_SCL_I2S_2CH_DIN_0 | SCT_OU_T_4 | RMII_RX_D1 | PWM_2H | CCI_CLK | M4SS_Q_SPI_D2 | SSI_SLV_CS | |
| GPIO_11 | GPIO_11 | SIO_5 | USB_DR_VVBus | SSI_MST_CLK | I2C2_SD_A | I2S_2CH_WS | SCT_OU_T_5 | RMII_MD_PWM_2L_C | CCI_CS[0] | M4SS_Q_SPI_D3 | SSI_SLV_CLK | |
| GPIO_12 | GPIO_12 | USART1_IR_RX | UART2_T_SSI_MST_X | I2C1_SCL_USART1_SCT_OU | RTS | T_6 | RMII_MD_SCT_IN_O | C0 | CCI_DAT_A_0 | MCU_CL_K_OUT | SSI_SLV_MISO | |
| GPIO_13 | GPIO_13 | SIO_7 | USART1_TX | USART1_GSPI_MS_T1_MOSI | USB_XTASCT_IN_L_ON | XTAL_ON_1 | _IN | F_CLK | PWM_3L | CCI_DAT_A_4 | CCI_DAT_A_1 | |

| GPIO | GPIO Mode = 0 | GPIO Mode = 1 | GPIO Mode = 2 | GPIO Mode = 3 | GPIO Mode = 4 | GPIO Mode = 5 | GPIO Mode = 6 | GPIO Mode = 7 | GPIO Mode = 8 | GPIO Mode = 9 | GPIO Mode = 10 | GPIO Mode = 14 | |
|---------|------------------|------------------|------------------|---------------------------|---|--------------------------------|------------------|-------------------|------------------------|------------------|-------------------|-------------------|--|
| GPIO_14 | GPIO_14 | ULP_GPI_O_7 | USART1_RX | GSPI_MSSCT_IN_T1_MISO2 | USB_DR_VVBUS | SCT_OU_T_0 | RMII_CR_S_DV | PWM_3HCCI_DAT_A_5 | HCCI_DAT_A_2 | | | | |
| GPIO_15 | GPIO_15 | MCU_CL_K_OUT | UART2_RX | SSI_MST_I2C1_SD_DATA1_A | I2C1_SD_A | USART1_CTS | SCT_OU_T_7 | RMII_RX_D0 | GSPI_MS_CCI_DAT_T1_CLK | CCI_DAT_A_6 | SSI_SLV_A_3 | _MOSI | |
| GPIO_16 | GPIO_16 | SIO_4 | USB_DR_VVBUS | SSI_MST_CAN1_R_CS0_XD | CAN1_R_CS0_XD | GSPI_MS_I2S_2CH_T1_CS0_WS | XTAL_O_N_IN | SSI_SLV_CS | CCI_DAT_A_0 | ULP_GPI_O_1 | RMII_CR_S_DV | | |
| GPIO_17 | GPIO_17 | SIO_5 | USB_XT_AL_ON | SSI_MST_CAN1_T_CS1_XD | CAN1_T_CS1_XD | GSPI_MS_I2S_2CH_T1_CLK | SCT_IN_CLK_0 | SSI_SLV_CLK | CCI_INT_R[0] | | | | |
| GPIO_18 | GPIO_18 | SIO_6 | USART1_RTS | SSI_MST_I2C2_SD_DATA2_A | I2C2_SD_A | GSPI_MS_I2S_2CH_T1_MOSI_DOUT_1 | SSI_SLV_MOSI | USART1_TX | PWM_1H | | | | |
| GPIO_19 | GPIO_19 | SIO_7 | USART1_CTS | SSI_MST_DATA3 | | GSPI_MS_I2S_2CH_T1_MISO_DIN_1 | SSI_SLV_MISO | USART1_RX | PWM_1L | SCT_OU_T_0 | | | |
| GPIO_20 | GPIO_20 | | | SSI_MST_CS2_L | I2C2_SC_L | GSPI_MS_T1_CLK | | XTAL_O_N_IN | USART1_CLK | PWM_2L | SCT_OU_T_0 | | |
| GPIO_21 | GPIO_21 | | | USART1_RTS_T1_CS0_0 | GSPI_MSSCT_IN_USB_XT_0 | ULP_GPI_O_8 | PWM_SL_P_EVEN | M4SS_S_MIH_D4 | CCI_CLK_T_TRIGGER | | | | |
| GPIO_22 | GPIO_22 | | | USART1_CTS_L_CLOC_1_K | SOC_PL_SCT_IN_1 | USB_DR_VVBUS_O_9 | ULP_GPI_ULTA | PWM_FA_MIH_D5 | M4SS_S[0] | CCI_CS[0] | | | |
| GPIO_23 | GPIO_23 | SIO_2 | | SSI_MST_CS3_2 | SCT_IN_DIN_1 | I2S_2CH_O_10 | ULP_GPI_ULTB | PWM_FA_MIH_D6 | M4SS_S[0] | CCI_RDY | | Analog Function | |
| GPIO_24 | GPIO_24 | | | | SCT_IN_3 | I2S_2CH_DOUT_O_11 | ULP_GPI | | M4SS_S_MIH_D7 | CCI_DAT_A_VALID | | Analog Function | |
| GPIO_25 | GPIO_25 | SIO_4 | | USART1_CLK_L | SSI_MST_I2C1_SC_L | RMII_TX_D1 | SCT_IN_2 | SCT_OU_T_0 | M4SS_S_MIH_CLK | UART2_TX_K | | Analog Function | |
| GPIO_26 | GPIO_26 | SIO_5 | | USART1_TX | SSI_MST_I2C1_SD_DATA0_A | RMII_TX_D0 | SCT_IN_3 | SCT_OU_T_1 | M4SS_S_MIH_CM | UART2_RX_D | | Analog Function | |
| GPIO_27 | GPIO_27 | SIO_6 | | USART1_RX | SSI_MST_GSPI_MS_I2S_2CH_DATA1_T1_CS0_WS | SCT_IN_4 | USART1_RTS | M4SS_S_MIH_D0 | SSI_SLV_CS | QEI_IDX | | Analog Function | |
| GPIO_28 | GPIO_28 | SIO_7 | | USART1_CLK_CS0 | SSI_MST_GSPI_MS_I2S_2CH_T1_CLK | I2C2_SC_L | USART1_CTS | M4SS_S_MIH_D1 | SSI_SLV_CLK | QEI_PHB | Analog Function | | |
| GPIO_29 | GPIO_29 | CAN1_R_XD | USART1_RX | SSI_MST_CS1_T1_MISO_DIN_0 | GSPI_MS_I2S_2CH_A | I2C2_SD_A | SCT_OU_T_2 | M4SS_S_MIH_D2 | SSI_SLV_MOSI | QEI_PHA | Analog Function | | |

| GPIO | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode |
|-------------|------------------|------------------|--------------------|------------------------|-------------------------------|---------------------|-------------------------|-----------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| | = 0 | = 1 | = 2 | = 3 | = 4 | = 5 | = 6 | = 7 | = 8 | = 9 | = 10 | = 11 | = 12 | = 13 | = 14 |
| GPIO_30 | GPIO_30 | CAN1_T_XD | USART1_TX | SSI_MST_CS2 | GSPI_MSI2S_2CH_T1_MOSI_DOUT_0 | SCT_IN_6_RMII_TX_EN | SCT_OUT_4_M4SS_S_MIH_D3 | M4SS_S_MIH_D3 | SSI_SLAVE_MISO | QEI_DIR | Analog Function | | | | |
| GPIO_31 | GPIO_31 | | | | | | SCT_OUT_4 | QEI_PHASE_PWM_4L | | | | | | | |
| GPIO_32 | GPIO_32 | | | | | | SCT_OUT_5 | QEI_PHASE_PWM_4H | | | | | | | |
| GPIO_33 | GPIO_33 | | | | | | | | | | | | | | |
| GPIO_34 | GPIO_34 | | | | | | | | | | | | | | |
| GPIO_35 | GPIO_35 | SIO_0 | REF_CLKSSI_MST_OUT | SSI_MST_CLK | SSI_MST_CS2 | I2C2_SC_L | SCT_IN_3 | RMII_TX_D1 | PWM_4L[0] | CCI_RDY | QEI_PHA | | | | |
| GPIO_36 | GPIO_36 | SIO_1 | UART2_TX | SSI_MST_DATA0 | M4SS_S_MIH_WP_A | I2C2_SD | SCT_OUT_5 | RMII_TX_D0 | PWM_4H | CCI_DAT | QEI_PHB | | | | |
| GPIO_37 | GPIO_37 | XTAL_O_N_IN | UART2_RX | SSI_MST_DATA1 | CCI_INT_R[0] | M4SS_S_MIH_CD | SCT_OUT_5 | RMII_TX_EN | PWM_3L | CCI_DAT | QEI_IDX_A_0 | | | | |
| GPIO_38 | GPIO_38 | REF_CLK | UART2_CTS | SDMEM_PRESENT | | | ULP_GPIO_12 | | PWM_3L | CCI_DAT_A_7 | PWM_T_MR_EXT | | | | |
| GPIO_39 | GPIO_39 | XTAL_O_N_IN | UART2_RTS | | USB_PL_CLOCK | | ULP_GPIO_13 | PWM_T_MR_EXT | PWM_3H | CCI_DAT_A_1 | | | | | |
| GPIO_40 | GPIO_40 | | | I2S_PLL_CLOCK | I2C2_SC_L | ULP_GPIO_14 | PWM_T_MR_EXT | PWM_4L | CCI_DAT_A_2 | | | | | | |
| GPIO_41 | GPIO_41 | | | INTERFA_CE_PLL_A_CLOCK | I2C2_SD | ULP_GPIO_15 | PWM_T_MR_EXT | PWM_4H | CCI_DAT_A_3 | | | | | | |
| GPIO_42 | GPIO_42 | | CCI_DAT_A_4 | I2S_2CH_CLK | SSI_MST_CS0 | SCT_IN_4 | GSPI_MS_PWM_1L_T1_CS0 | | | | | | | | |
| GPIO_43 | GPIO_43 | | CCI_DAT_A_5 | UART2_RS485_E_WS_N | I2S_2CH_CS1 | SSI_MST_CS1 | SCT_IN_5 | GSPI_MS_PWM_1H_T1_CS1 | USB_XT_AL_ON | | | | | | |
| GPIO_44 | GPIO_44 | | CCI_DAT_A_6 | UART2_RS485_R_DIN_E | I2S_2CH_CS2 | SSI_MST_CS2 | SCT_IN_6 | GSPI_MS_PWM_2L_T1_CS2 | USB_DR_VVBUS | | | | | | |
| GPIO_45 | GPIO_45 | | CCI_DAT_A_0 | UART2_RS485_D_DOUT_E | I2S_2CH_CS3 | SSI_MST_CS3 | SCT_IN_7 | GSPI_MS_PWM_2H_T1_CS3 | | | | | | | |

| GPIO | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode | GPIO Mode |
|-------------|------------------|------------------|------------------|----------------------------|-------------------------|----------------------------|-------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| | = 0 | = 1 | = 2 | = 3 | = 4 | = 5 | = 6 | = 7 | = 8 | = 9 | = 10 | = 11 | = 12 | = 13 | = 14 |
| GPIO_46 | GPIO_46 | CAN1_R | UART2_RX | ULP_GPI_O_12 | GSPI_MS_T1_CS3 | USART1_DSR | SCT_OUT2 | I2S_2CH_DIN1 | M4SS_S_MIH_CLK | CCI_INT_R[1] | M4SS_Q_SPI_CLK | | | | |
| GPIO_47 | GPIO_47 | CAN1_T | UART2_TX | SSI_MST_CS3 | GSPI_MS_T1_CS2 | USART1_DCD | SCT_OUT3 | I2S_2CH_DOUT1 | M4SS_S_MIH_CM | CCI_RDY[0] | M4SS_Q_SPI_D0 | | | | |
| GPIO_48 | GPIO_48 | SIO_0 | | ULP_GPI_O_0_CS2 | SSI_MST_UART2_RS485_E_N | USART1_DTR | SCT_OUT4 | I2S_2CH_WS | M4SS_S_MIH_D0 | CCI_DAT_A_VALID | M4SS_Q_SPI_D1 | | | | |
| GPIO_49 | GPIO_49 | SIO_1 | | ULP_GPI_O_1_RT | UART2_T1_CS1 | GSPI_MS_I2S_2CH_CLK | SCT_OUT5 | RMII_TXD1 | M4SS_S_MIH_D1 | CCI_CS[1] | M4SS_Q_SPI_CSN0 | | | | |
| GPIO_50 | GPIO_50 | SIO_2 | | ULP_GPI_O_2_CTS | UART2_RS485_RLE | I2C1_SC | SCT_OUT0 | RMII_TXD0 | M4SS_S_MIH_D2 | CCI_RDY[1] | M4SS_Q_SPI_D2 | | | | |
| GPIO_51 | GPIO_51 | SIO_3 | | ULP_GPI_O_3_AL_ON | USB_XTAL_ON | UART2_RS485_DA_E | I2C1_SD | SCT_OUT1 | RMII_TXEN | M4SS_S_MIH_D3 | CCI_INT_R[0] | M4SS_Q_SPI_D3 | | | |
| GPIO_52 | GPIO_52 | SIO_4 | | M4SS_Q_SPI_CLK_RACE_C_LKIN | M4SS_T_O_13 | ULP_GPI_USART1_RI | SCT_OUT7 | RMII_RXD1 | M4SS_S_MIH_WP | CCI_CLK | M4SS_Q_SPI_DQS | | | | |
| GPIO_53 | GPIO_53 | SIO_5 | | M4SS_Q_SPI_CSN0_LK | M4SS_T_RACE_C | | USART1_IR_RX | SCT_IN3 | RMII_MD_C | M4SS_S_MIH_CD | CCI_CS[0] | M4SS_Q_SPI_CSN1 | | | |
| GPIO_54 | GPIO_54 | SIO_6 | | M4SS_Q_SPI_D0_RACE_D0 | M4SS_T_CCI_DAT_A_0 | USART1_IR_TX | ULP_GPI_O_4 | RMII_MD_O | M4SS_S_MIH_D4 | CCI_DAT_A_4 | M4SS_Q_SPI_D4 | | | | |
| GPIO_55 | GPIO_55 | SIO_7 | | M4SS_Q_SPI_D1_RACE_D1 | M4SS_T_CCI_DAT_A_1 | USART1_RS485_O_5_EN | ULP_GPI_O_5_F_CLK | RMII_RE | M4SS_S_MIH_D5 | CCI_DAT_A_5 | M4SS_Q_SPI_D5 | | | | |
| GPIO_56 | GPIO_56 | | | M4SS_Q_SPI_D2_RACE_D2 | M4SS_T_CCI_DAT_A_2 | USART1_RS485_O_6_RS485_O_6 | ULP_GPI_O_6_S_DV | RMII_CR | M4SS_S_MIH_D6 | CCI_DAT_A_6 | M4SS_Q_SPI_D6 | | | | |
| GPIO_57 | GPIO_57 | | | M4SS_Q_SPI_D3_RACE_D3 | M4SS_T_CCI_DAT_A_3 | USART1_RS485_O_7_DE | ULP_GPI_O_7_D0 | RMII_RXD0 | M4SS_S_MIH_D7 | CCI_DAT_A_7 | M4SS_Q_SPI_D7 | | | | |

GPIO Pin Multiplexing

Note that GPIO's 23 to 30 can be used for Analog functions when GPIO Mode = 14. The analog function available on these GPIOs is further determined by GPIO Analog Mode (2-bit programmable value) for each of these GPIOs.

| | | | |
|-------------|-----------------------------|-----------------------------|-----------------------------|
| GPIO | GPIO Analog Mode = 0 | GPIO Analog Mode = 1 | GPIO Analog Mode = 4 |
|-------------|-----------------------------|-----------------------------|-----------------------------|

| | | | |
|---------|----------------|----------------|------------|
| GPIO_23 | ADCP9 | | |
| GPIO_24 | ADCP19 / ADCN9 | | |
| GPIO_25 | ADCP6 | | |
| GPIO_26 | ADCP16 / ADCN6 | | |
| GPIO_27 | ADCP7 | TOUCH_VREF_EXT | OPAMP3OUT0 |
| GPIO_28 | ADCP17 / ADCN7 | | |
| GPIO_29 | ADCP8 | | |
| GPIO_30 | ADCP18 / ADCN8 | | |

GPIO Pin Multiplexing - Analog Functions

11.2.2 ULP GPIOs

The ULP GPIOs listed in the table below (ULP_GPIO_0 to ULP_GPIO_15) are available in the normal mode of operation (Power-states 4 and 3) and also in Ultra-low power mode of operation of the Microcontroller (Power-states 2 and 1). The power-states are described in [Power Architecture](#).

Each of these GPIO's Pin function is configured through a 4-bit GPIO Mode of GPIO_CONFIG_REG_n(n=0:15) as described in [ULP Enhanced-GPIO](#) Section of MCU APB Peripherals.

| ULP_GPIO | ULP GPIO Mode = 0 | ULP GPIO Mode = 1 | ULP GPIO Mode = 2 | ULP GPIO Mode = 3 | ULP GPIO Mode = 4 | ULP GPIO Mode = 5 | ULP GPIO Mode = 6 | ULP GPIO Mode = 7 |
|-------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| ULP_GPIO_0 | ULP_EGPIO[0] | ULP_SPI_CL_K | ULP_I2S_DI_N | ULP_UART_RTS | ULP_I2C_SD_A | UULP_VBAT_GPIO_1 | SOC_GPIO_0 | Analog Function |
| ULP_GPIO_1 | ULP_EGPIO[1] | ULP_SPI_D_OUT | ULP_I2S_D_OUT | ULP_UART_CTS | ULP_I2C_SC_L | Timer0 | SOC_GPIO_1 | Analog Function |
| ULP_GPIO_2 | ULP_EGPIO[2] | ULP_SPI_DI_N | ULP_I2S_W_S | ULP_UART_RX | ULP_SPI_CS_1 | COMP1_OU_T | SOC_GPIO_2 | Analog Function |
| ULP_GPIO_3 | ULP_EGPIO[3] | ULP_SPI_CS_0 | ULP_I2S_CL_K | ULP_UART_TX | ULP_SPI_DI_N | | SOC_GPIO_3 | Analog Function |
| ULP_GPIO_4 | ULP_EGPIO[4] | ULP_SPI_CS_1 | ULP_I2S_W_S | ULP_UART_RTS | ULP_I2C_SD_A | | SOC_GPIO_4 | Analog Function |
| ULP_GPIO_5 | ULP_EGPIO[5] | IR_PG_EN | ULP_I2S_D_OUT | ULP_UART_CTS | ULP_I2C_SC_L | | SOC_GPIO_5 | Analog Function |
| ULP_GPIO_6 | ULP_EGPIO[6] | ULP_SPI_CS_2 | ULP_I2S_DI_N | ULP_UART_RX | ULP_I2C_SD_A | UULP_VBAT_GPIO_1 | SOC_GPIO_6 | Analog Function |
| ULP_GPIO_7 | ULP_EGPIO[7] | IR_INPUT | ULP_I2S_CL_K | ULP_UART_TX | ULP_I2C_SC_L | Timer1 | SOC_GPIO_7 | Analog Function |
| ULP_GPIO_8 | ULP_EGPIO[8] | ULP_SPI_CL_K | ULP_I2S_CL_K | ULP_UART_CTS | ULP_I2C_SC_L | Timer0 | SOC_GPIO_8 | Analog Function |
| ULP_GPIO_9 | ULP_EGPIO[9] | ULP_SPI_DI_N | ULP_I2S_DI_N | ULP_UART_RX | ULP_I2C_SD_A | COMP1_OU_T | SOC_GPIO_9 | Analog Function |
| ULP_GPIO_10 | ULP_EGPIO[10] | ULP_SPI_CS_0 | ULP_I2S_W_S | ULP_UART_RTS | IR_INPUT | UULP_VBAT_GPIO_4 | SOC_GPIO_10 | Analog Function |

| ULP_GPIO | ULP GPIO Mode = 0 | ULP GPIO Mode = 1 | ULP GPIO Mode = 2 | ULP GPIO Mode = 3 | ULP GPIO Mode = 4 | ULP GPIO Mode = 5 | ULP GPIO Mode = 6 | ULP GPIO Mode = 7 |
|-------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-----------------------------|
| ULP_GPIO_11 | ULP_EGPIO[11] | OUT | ULP_SPI_D | ULP_I2S_D | ULP_UART_TX | ULP_I2C_SD_A | SOC_GPIO_11 | Analog Function |
| ULP_GPIO_12 | ULP_EGPIO[12] | 1 | ULP_SPI_CS | ULP_I2S_CL_K | | ULP_I2C_SD_A | UULP_VBAT_GPIO_1 | SOC_GPIO_12 Analog Function |
| ULP_GPIO_13 | ULP_EGPIO[13] | 2 | ULP_SPI_CS | ULP_I2S_DI_N | COMP2_OUT_T | ULP_I2C_SC_L | Timer1 | SOC_GPIO_13 Analog Function |
| ULP_GPIO_14 | ULP_EGPIO[14] | UULP_VBAT_GPIO_4 | ULP_I2S_W_S | Timer0 | IR_PG_EN | COMP1_OUT_T | SOC_GPIO_14 | Analog Function |
| ULP_GPIO_15 | ULP_EGPIO[15] | | Timer2 | ULP_I2S_D_OUT | UULP_VBAT_GPIO_4 | IR_INPUT | | SOC_GPIO_15 Analog Function |

ULP GPIO Pin Multiplexing

Note that each of the ULP_GPIO's 0 to 15 can be used for MCU HP Peripheral functions when ULP GPIO Mode = 6.

The MCU HP Peripheral function available on these ULP_GPIOS is further configured by GPIO Mode (4-bit programmable value) of GPIO_CONFIG_REG_n(n=64:79) as described in [Enhanced-GPIO](#) Section of MCU APB Peripherals.

| ULP_GPIO | GPIO Mode = 0 | GPIO Mode = 1 | GPIO Mode = 2 | GPIO Mode = 3 | GPIO Mode = 4 | GPIO Mode = 5 | GPIO Mode = 6 | GPIO Mode = 7 | GPIO Mode = 8 | GPIO Mode = 9 | GPIO Mode = 10 |
|--------------|---------------|-------------------|-----------------|----------------|------------------------|-----------------|-------------------|--------------------------|-----------------------|---------------------|----------------|
| ULP_GPIO_0_0 | GPIO_64 | SIO_0 | SCT_IN_0 | SCT_OUT_T_4 | SSI_MST_CS1 | GSPI_MS_T1_CS2 | UART2_RQEI_IDX_X | PWM_TM_R_EXT_T | PWM_FA_ULTA_RIG_1 | | |
| ULP_GPIO_0_1 | GPIO_65 | SIO_1 | SCT_IN_1 | SCT_OUT_T_5 | SSI_MST_CS2 | GSPI_MS_T1_CS3 | UART2_T_QEI_PHA_X | PWM_TM_R_EXT_T | PWM_FA_ULTB_RIG_2 | | |
| ULP_GPIO_0_2 | GPIO_66 | SIO_2 | SCT_IN_2 | SCT_OUT_T_6 | UART2_C_I2C2_SCL_TS | I2C2_SCL_PWM_1L | QEI_PHB_TS | PWM_TM_R_EXT_T | USB_DRV_VBUS_RIG_3 | | |
| ULP_GPIO_0_3 | GPIO_67 | SIO_3 | SCT_IN_3 | SCT_OUT_T_7 | UART2_R_I2C2_SD_TS | I2C2_SD_PWM_1H | QEI_DIR_A | PWM_TM_R_EXT_T | USB_XTA_L_ON_RIG_4 | | |
| ULP_GPIO_0_4 | GPIO_68 | UART2_R_USART1_X | SSI_MST_CTS | SSI_CLK_D | CAN1_RX_SIO_0 | SCT_OU_T_0 | I2C1_SC_L | PWM_1H_PWM_SL_ST_1 | PMU_TE_I2S_2CH_CLK | | |
| ULP_GPIO_0_5 | GPIO_69 | UART2_T_USART1_X | SSI_MST_RS485_D | SSI_CLK_E | CAN1_TX_SIO_1 | SCT_OU_T_1 | I2C1_SD_A | PWM_1L_PWM_SL_P_EVENT_WS | I2S_2CH_TRIGGER | | |
| ULP_GPIO_0_6 | GPIO_70 | UART2_C_USART1_TS | SSI_MST_RS485_E | SSI_CLK_N | PMU_TE_RS485_E | SIO_2_ST_2 | SCT_OU_T_2 | I2C2_SC_L | PWM_2L_PWM_TM_R_EXT_T | I2S_2CH_DIN_0_RIG_1 | |
| ULP_GPIO_0_7 | GPIO_71 | UART2_R_USART1_TS | SSI_MST_RS485_E | SSI_CLK_E | GSPI_MS_T1_CS1 | SIO_3 | SCT_OU_T_3 | I2C2_SD_A | PWM_2H_PWM_TM_R_EXT_T | I2S_2CH_DOUT_RIG_2 | DOUT_0 |
| ULP_GPIO_0_8 | GPIO_72 | UART2_R_USART1_X | SSI_MST_CLK | SSI_CLK_T1_CLK | GSPI_MS_I2S_2CH_T1_CLK | I2S_2CH_CLK | SCT_OU_T_4 | I2C1_SC_L | PWM_3L_PWM_3H | SSI_SLV_CLK | USART1_CTS |

| | | | | | | | | | | | |
|--------------|---------|-----------|----------------|-------------|----------------|------------|------------|-----------|----------------|-------------|---------|
| ULP_GPI_O_9 | GPIO_73 | UART2_T_X | USART1_CS0_RTS | SSI_MST_CS0 | GSPI_MS_T1_CS0 | I2S_2CH_WS | SCT_OU_T_5 | I2C1_SD_A | PWM_3H_CS | SSI_SLV_IN | XTAL_ON |
| ULP_GPI_O_10 | GPIO_74 | CAN1_RX_D | USART1_RX_RX | SSI_MST_CS0 | GSPI_MS_T1_CS0 | I2S_2CH_WS | SCT_IN_4 | I2C2_SC_L | PWM_4L_MOSI | SSI_SLV_T_6 | SCT_OU |
| ULP_GPI_O_11 | GPIO_75 | CAN1_TX_D | USART1_TX_TX | SSI_MST_CS0 | GSPI_MS_T1_CS0 | I2S_2CH_WS | SCT_IN_5 | I2C2_SD_A | PWM_4H_MISO | SSI_SLV_T_7 | SCT_OU |
| ULP_GPI_O_12 | GPIO_76 | SIO_4 | | SCT_IN_4 | SCT_OU_T_0 | | | PWM_2L | PWM_SL_P_EVENT | | |
| ULP_GPI_O_13 | GPIO_77 | SIO_5 | | SCT_IN_5 | SCT_OU_T_1 | | | PWM_2H | PWM_FAULTA | | |
| ULP_GPI_O_14 | GPIO_78 | SIO_6 | | SCT_IN_6 | SCT_OU_T_2 | | | | PWM_FAULTB | | |
| ULP_GPI_O_15 | GPIO_79 | SIO_7 | | SCT_IN_7 | SCT_OU_T_3 | | | | | | |

11.2.3 UULP VBAT GPIOs

The UULP VBAT GPIOs listed in the table below (UULP_VBAT_GPIO_0 to UULP_VBAT_GPIO_4) are available in the normal mode of operation (Power-states 4 and 3), in Ultra-low power mode of operation (Power-states 2 and 1) and also in the retention and deep sleep mode of operation (Retention and Power-state 0). The power-states are described in [Power Architecture](#).

Each of this UULP VBAT GPIO's Pin function can be configured by a 3-bit GPIO Mode of UULP_VBAT_GPIOOn_CONFIG_REG (n=0:4) as described in [Pad Configurations](#).

| UULP_VBAT GPIO | UULP_VBAT GPIO | UULP_VBAT GPIO | UULP_VBAT GPIO | UULP_VBAT GPIO | UULP_VBAT GPIO | UULP_VBAT GPIO | UULP_VBAT GPIO | UULP_VBAT GPIO | Default |
|-------------------|--------------------|----------------|--------------------------|----------------|----------------|----------------|----------------|----------------|---------------------------|
| Mode = 0 | Mode = 1 | Mode = 2 | Mode = 3 | Mode = 4 | Mode = 5 | Mode = 6 | Mode = 7 | | |
| UULP_VBA_T_GPIO_0 | UULP_VBA_T_GPIO[0] | EXT_PG_E_N | MCU_GPIO0_WAKEUP | | | | | | EXT_PG_E_N |
| UULP_VBA_T_GPIO_1 | UULP_VBA_T_GPIO[1] | XTAL_EN | MCU_GPIO32KHZ_XT1_WAKEUP | AL_CLK | | | | | XTAL_EN |
| UULP_VBA_T_GPIO_2 | UULP_VBA_T_GPIO[2] | | MCU_GPIO2_WAKEUP | | 32KHZ_XTAL_CLK | | | | VOLT_SEN MCU_GPIO2_WAKEUP |
| UULP_VBA_T_GPIO_3 | UULP_VBA_T_GPIO[3] | | MCU_GPIO3_WAKEUP | | | 32KHZ_XTAL_CLK | | | COMP_P UULP_VBA_T_GPIO[3] |
| UULP_VBA_T_GPIO_4 | UULP_VBA_T_GPIO[4] | | MCU_GPIO4_WAKEUP | | | | 32KHZ_XTAL_CLK | COMP_N | UULP_VBA_T_GPIO[4] |

UULP VBAT GPIO Pin Multiplexing

11.2.4 Description of Digital Peripheral Pins Multiplexed on GPIOs

| Pin Name | Direction | Description |
|--|-----------|-------------------|
| CAN (Controller Area Network) Interface | | |
| CAN1_RXD | Input | CAN Receive Data |
| CAN1_TXD | Output | CAN Transmit Data |
| CCI (Companion Chip Interface) | | |

| Pin Name | Direction | Description |
|--------------------------|------------------|--|
| CCI_CLK | Output/ Input | Output Clock from the CCI Controller when CCI is in Arbiter-Master-Slave(AMS) Mode Input Clock from the CCI Controller when CCI is in Master-Slave(MS) Mode |
| CCI_CS[1: 0] | Output/ Input | Active Low Chip Select. Output in AMS Mode and input in MS Mode. All the 3 bits are valid and CCI Controller can select three external slaves in AMS Mode. Only bit [0] is only valid in MS Mode. |
| CCI_DATA_0 to CCI_DATA_7 | Input | Bidirectional Data. All the bits are under the control of AMS or one of the three MS |
| CCI_DATA_VALID | Output | Active high Indicates that the data on CCI_DATA_0 to CCI_DATA_7 is valid |
| CCI_INTR[1:0] | Input/ Output | Active high interrupt. Input in AMS Mode and Output in MS Mode All the 3 bits are valid and CCI Controller can get interrupt from any of the three external slaved in AMS Mode. Only bit [0] is only valid in MS Mode. |
| CCI_RDY[1:0] | Input/ Output | Active Low Ready. Input in AMS Mode and Output in MS Mode. MS accepts data from AMS only when corresponding CCI_RDY is zero. All the 3 bits are valid and CCI Controller can get ready from any of the three external slaved in AMS Mode. Only bit [0] is only valid in MS Mode. |

GSPI (General SPI) Interface

| | | |
|--------------------------------|--------|---|
| GSPI_MST1_CLK | Output | Output Clock from the GSPI master to external slave |
| GSPI_MST1_CS0 to GSPI_MST1_CS3 | Output | Active Low CSN. GSPI Master can select a maximum of 4 slaves. |
| GSPI_MST1_MISO | Input | Input data to master from external slave |
| GSPI_MST1_MOSI | Output | Output data from master to external slave |

I2C (Inter-integrated Circuit) Interface

| | | |
|--------------------------|-------|-----------------------------|
| I2Cx_SCL, ULP_I2C_SCL | Inout | I2C Serial Clock x= 1, 2 |
| I2Cx_SDA, ULP_I2C_SDA | Inout | I2C Serial Data x= 1, 2 |

2 Channel I2S (Inter-IC Sound) Interface

| Pin Name | Direction | Description |
|---|-----------|---|
| I2S_2CH_CLK | Output/ | I2S Clock |
| | Input | Output in Master Mode and Input in Slave Mode |
| I2S_2CH_WS | Output/ | Active high I2S Word Select |
| | Input | Output in Master Mode and Input in Slave Mode |
| I2S_2CH_DIN_0 to I2S_2CH_DIN_1 | Input | I2S Input Data |
| I2S_2CH_DOUT_0 to I2S_2CH_DOUT_1 | Output | I2S Output Data |
| QSPI (Quad SPI) Interface | | |
| M4SS_QSPI_CLK | Output | Output clock to the external SPI slave. |
| M4SS_QSPI_CSN0 to M4SS_QSPI_CSN1 | Output | Active Low Chip Select to select a maximum of two slaves. |
| M4SS_QSPI_D0 to M4SS_QSPI_D7 | Inout | QSPI Data. Supports both QUAD and OCTA Data. In Quad Mode, only Bits M4SS_QSPI_D0 to M4SS_QSPI_D3 are valid. In Octa Mode, all the bits are valid |
| M4SS_QSPI_DQS | Input | Data Strobe signal |
| SMIH (SD/SDIO/MMC Host Controller) Interface | | |
| M4SS_SMIH_CLK | Output | Output Clock from the SMIH Controller |
| M4SS_SMIH_CMD | Output | Output Command from the SMIH Controller |
| M4SS_SMIH_D0 to M4SS_SMIH_D7 | Inout | Bidirectional 8-bit Data |
| M4SS_SMIH_CD_N | Input | Active Low Card Detect |
| M4SS_SMIH_WP | Input | Active Low Write Protect |
| M4 Trace Interface | | |
| M4SS_TRACE_CLK | Output | Trace Clock from Cortex M4 |
| M4SS_TRACE_CLKIN | Input | Trace Port Clock to Cortex M4 |
| M4SS_TRACE_D0 to M4SS_TRACE_D3 | Output | Trace Port Data bus from Cortex M4 |
| PWM (Pulse Width Modulation) Interface | | |
| PWM_xH | Output | PWM output signals. The output pins are grouped in pairs, to facilitate driving the low side and high side of a power half-bridge. |
| PWM_xL | Output | $x = 1,2,3,4$ |
| PWM_FAULTA | Input | External fault signal A |
| PWM_FAULTB | Input | External fault signal B |

| Pin Name | Direction | Description |
|---|-----------|--|
| PWM_SLP_EVENT_TRIG | Output | Special event trigger for synchronizing analog to digital conversions. |
| PWM_TMR_EXT_TRIG_1 to PWM_TMR_EXT_TRIG_4 | Input | External trigger for base timers to increment. Each Channel has separate trigger input. |
| QEI (Quadrature Encode Interface) | | |
| QEI_DIR | Output | Position counter direction. '1' means counter direction is positive. '0' means counter direction is negative. |
| QEI_IDX | Input | QE Index. Index pulse occurs once per mechanical revolution and is used as a reference to indicate an absolute position. |
| QEI_PHA | Input | QE Phase A input |
| QEI_PHB | Input | QE Phase B input |
| RMII (Reduced media-independent interface) | | |
| RMII_CRS_DV | Input | PHY Receive Data Valid |
| RMII_MDC | Output | Management Data Clock |
| RMII_MDO | Inout | Management Data |
| RMII_REF_CLK | Output/ | Output Clock when clock is provided from Internal PLL |
| | Input | The Clock is taken from external interface when internal PLL is not used. |
| RMII_RXD0 to RMII_RXD1 | Input | Receive Data |
| RMII_TXD0 to RMII_TXD1 | Output | Transmit Data |
| RMII_TXEN | Output | Active High Transmit Data Enable. When asserted indicates that Transmit Data is valid. |
| SCT (State Configurable Timer) Interface | | |
| SCT_IN_0 to SCT_IN_7 | Input | Timer input event |
| SCT_OUT_0 to SCT_OUT_7 | Output | Timer output event |
| SIO (Serial Input Output) Interface | | |
| SIO_0 to SIO_7 | Inout | Serial Input-Output Data |
| SSI (Synchronous Serial Interface) Master | | |
| SSI_MST_CLK | Output | Output clock from SSI Master |
| SSI_MST_CS0 to SSI_MST_CS3 | Output | Active Low Chip select |
| SSI_MST_DATA0 to SSI_MST_DATA3 | Inout | Bidirectional Data |
| SSI (Synchronous Serial Interface) Slave | | |
| SSI_SLV_CLK | Input | Input clock to SSI Slave |
| SSI_SLV_CS | Input | Active Low Chip select |

| Pin Name | Direction | Description |
|--------------------------------|-----------|--------------------------|
| SSI_SLV_MISO | Output | Slave Output Data |
| SSI_SLV_MOSI | Input | Slave Input Data |
| UART Interface | | |
| UART2_CTS, ULP_UART_CTS | Input | Clear to Send |
| UART2_RTS, ULP_UART_RTS | Output | Request to Send |
| UART2_RS485_DE | Output | Driver Enable |
| UART2_RS485_EN | Output | Active high RS485 Enable |
| UART2_RS485_RE | Output | Receiver Enable |
| UART2_RX, ULP_UART_RX | Input | Serial Input |
| UART2_TX, ULP_UART_TX | Output | Serial Output |
| USART Interface | | |
| USART1_CLK | | |
| USART1_CTS | Input | Clear to Send |
| USART1_RTS | Output | Request to Send |
| USART1_DCD | Input | Data Carrier Detect |
| USART1_DSR | Input | Data Set Ready |
| USART1_DTR | Output | Data Terminal Ready |
| USART1_IR_RX | Input | IrDA SIR Input |
| USART1_IR_TX | Output | IrDA SIR Ouput |
| USART1_RI | Input | Ring Indicator |
| USART1_RS485_DE | Output | Driver Enable |
| USART1_RS485_EN | Output | Active high RS485 Enable |
| USART1_RS485_RE | Output | Receiver Enable |
| USART1_RX | Input | Serial Input |
| USART1_TX | Output | Serial Output |
| Miscellaneous Interface | | |
| INTERFACE_PLL_CLOCK | Output | Clock from Interface PLL |
| I2S_PLL_CLOCK | Output | Clock from I2S PLL |
| USB_PLL_CLOCK | Output | USB Clock from Modem PLL |

| Pin Name | Direction | Description |
|--------------------------------|-----------|---|
| SOC_PLL_CLOCK | Output | Clock from SoC PLL |
| MEMS_REF_CLOCK | Output | Mems Ref Clock from Modem PLL |
| REF_CLK_OUT | Output | Reference Clock used by M4 SoC |
| RF_REF_CLK_OUT | Output | Clock from Internal RF |
| MCU_CLK_OUT | Output | All the Clocks that are used by M4 SoC are multiplexed and connected on this pin |
| PLL_TESTMODE_SIG | Output | Test Mode Signal from SoC PLL for Debug |
| PMU_TEST_1 to PMU_TEST_2 | Inout | Test Pins from IPMU for Debug |
| SDMEM_PRESENT | | |
| USB_CDC | | |
| USB_DRVVBUS | Output | Signal from USB Controller to be connected to be connected to off-chip charge pump circuit. |
| USB_XTAL_ON | Output | If the reference clock to the USB PLL is fed through GPIO_25, this signal is used to control the Crystal Oscillator which generates the reference clock |
| XTAL_ON_IN | Input | Crystal oscillator enable input when crystal is shared by another chip |
| UULP Vbat Pin Interface | | |
| BUCK_BOOST_EN | Output | Enable to an external Buck boost regulator or Power gate that controls the supply to all supply pins of the chip except the UULP Vbat GPIO supply |
| XTAL_EN | Output | External crystal oscillator enable |
| 32KHZ_XTAL_CLK | Input | Low Frequency clock input from an External 32KHz Crystal oscillator |
| MCU_GPIO0/1/2/3/4_WAKEUP | Input | GPIOs that can be used as Wakeup interrupt to MCU while in Retention or Deepsleep mode |
| NWP_GPIO0/2/3_WAKEUP | Input | GPIOs that can be used as Wakeup interrupt to Wireless Network processor (NWP) while in retention or deepsleep mode. |

11.2.5 Description of Analog Peripheral Pins Multiplexed on GPIOs

| Pin Name | Direction | Description |
|---------------------|-----------|---|
| ADC Inteface | | |
| ADCP0 - ADCP19 | Input | The 20 single ended input channels that are multiplexed onto the ADC P0 - P9 can be coupled with N0 - N9 for differential mode of operation of the ADC |
| ADCN0 - ADCN9 | Input | The N pins of the 10 possible differential channels multiplexed onto the ADC |
| DAC Inteface | | |

| Pin Name | Direction | Description |
|-------------------------------|-----------|---|
| DAC0, DAC1 | Output | Possible output pins from the internal DAC |
| OpAmp Interface | | |
| OPAMP"xyz" | Input | Multiplexed inputs of the three OpAmps. xyz denote the OpAmp number, the terminal and the multiplexing on that pin of the OpAmp x = OpAmp number (1, 2 or 3) y = P or N terminal of OpAmp z = 0, 1, 2, 3, 4, 5 (Multiplexing at OpAmp input pin). Note that OPAMP1P is available at 6 locations, OPAMP2P, 3P and 1N are available at 2 locations each and OPAMP2N and 3N pins are available at only one location |
| OPAMP1OUT0/1, OPAMP2/3OUT0 | Output | Outputs of the three OpAmps. Note that OPAMP1 output is available at two possible pin locations whereas OPAMP2 and 3 outputs are available at a fixed pin |
| Comparator Interface | | |
| CMP"xyz" | | Multiplexed inputs of the two Comparators. xyz denote the Comparator number, the terminal and the multiplexing on that pin of the Comparator x = Comparator number (A or B) y = P or N terminal of OpAmp z = 0, 1 (Multiplexing at Comparator Input pin). Note that each input pin of both comparators is available on two possible GPIO pins. |
| Touch Interface | | |
| TOUCH0/1/2/3/4/5/6/ 7 | Input | Capacitive Touch inputs |

12 SPI Flash Controller

12.1 General Description

The SPI Flash Controller is a 1/2/4/8-wired interface for serial access of data from Flash. It can be used in either Single, Dual, Quad or Octa modes with support for SDR and DDR to read the Processor's instructions and for data transfers to/from the Flash. The Controller supports inline decryption of encrypted instructions read from the Flash before they are passed on to the Processor's Instruction Cache. Instructions are read using the Direct Fetch mode while data transfers use the Indirect Access mode. The MCU has a DLL which is used for high-frequency DDR read/write operations along with the dedicated DDR I/O pads - the availability of these DDR I/Os is package dependent.

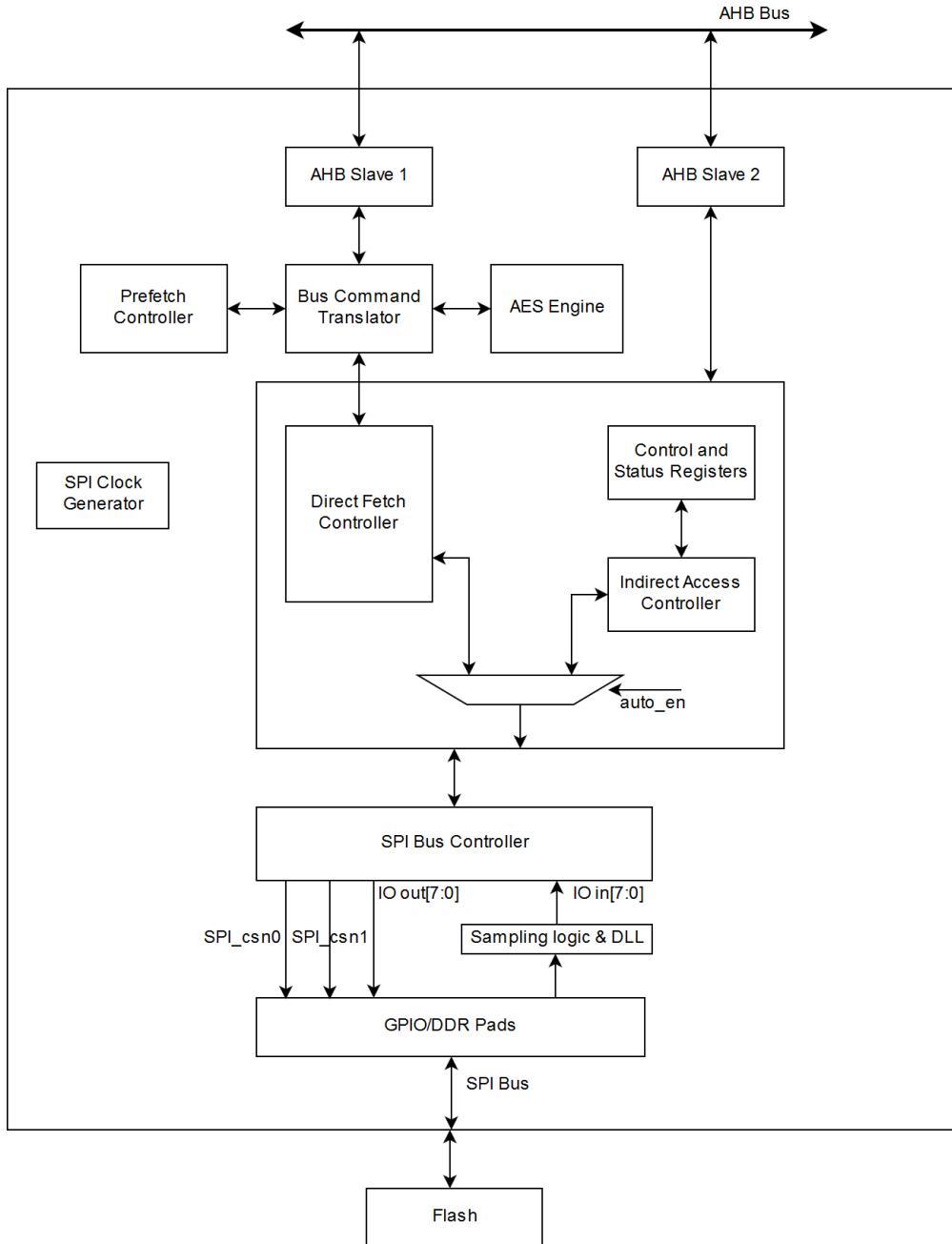
12.2 Features

- Supports Single/Dual/Quad/Octa (S/D/Q/O) modes for reading processor instructions and data transfers to/from Flash.
- Support for SPI Mode-0 and Mode-3
- Supports full duplex mode in single-bit SPI mode. Support for HOST SPI slave interface.
- Support for both SDR and DDR mode Flashes
- Supports both 8 and 16-bit Flash commands.
- Support both 24 and 32-bit addressing modes
- Supports inline decryption (AES) for while reading encrypted instructions from the Flash
- Supports up to two Flashes connected to CSN0 and CSN1
- Direct Fetch Mode:
 - Instructions are read from Flash using the Direct Fetch mode which does not need any processor involvement after the initial configuration of the Controller. The read command used for this mode is programmable depending on the Flash used.
 - Direct Fetch mode supports Wrap / Incremental / Single read operations.
 - Supports prefetch option - enabling this option makes the SPI Controller prefetch the next instruction before the request is posted on the internal AHB bus. If the address for the next instruction is different from the prefetch address, the instruction is scrapped.
 - Supports continuous fetch option to reduce instruction fetch delay from Flash - this option makes the SPI controller to post the Command and Address only once on the bus to read contiguous instructions by controlling only the CSN.
 - Supports programmable CSN high time.
- Indirect Access Mode:
 - Configuration of Flash and reading/writing data from/to the Flash uses the Indirect Access mode which requires the processor to program the SPI Flash controller for each access.
 - Supports reading of up to 32KB bytes of data from Flash in a single read operation.
 - In addition to 24 and 32-bit addressing, the SPI Controller supports 9, 10 and 16-bit addressing in this mode.
- Clock Configuration
 - Support for selection of source clock between AHB bus clock and PLL clock.
 - Support for even division factors up to 64 to generate the SPI clock from the source clock.
- Transmission of Extra-byte after the address phase is supported. The contents of this byte are programmable. There is also an option to only transmit the first nibble of the extra byte and maintain a Hi-z on the bus for the next nibble.
- Each phase of a Read operation (Command, Address, Dummy Byte, Extra Byte, Read Data) can be in any of the S/D/Q/O modes depending on the Flash requirements.
- The number of dummy bytes is programmable and can be programmed as per the instruction and the mode of operation.
- Supports DMA flow control and programmable FIFO thresholds
- Supports ultra high speed mode option with DLL and dedicated DDR I/O pads - the availability of these DDR I/Os is package dependent.

- Supports dual Flash mode - reading of data from two flashes simultaneously.
- Supports Flash Write Protect
- Supports interrupt generation based on different events

12.3 Functional Description

The SPI Flash Controller's block diagram is shown below:



SPI Flash Controller Block Diagram

The SPI Flash Controller in the MCU has been designed with programmable options for most of the single and multi-bit operations so that it can interface with Flash ICs from multiple vendors. The list of supported Flashes and vendors is given below:

| S.No. | Vendor | Part # | Flash Density (in Mbit) | Vcc | Bus Width |
|-------|------------|--------------|----------------------------|------------|-------------|
| 1 | GigaDevice | GD25LQ32D | 32 | 1.65V-2.0V | 1/2/4-bit |
| 2 | GigaDevice | GD25LQ80B | 8 | 1.65V-2.1V | 1/2/4-bit |
| 3 | Macronix | MX25R3235F | 32 | 1.65V-3.6V | 1/2/4-bit |
| 4 | Macronix | MX25U3235F | 32 | 1.65V-2.0V | 1/2/4-bit |
| 5 | Macronix | MX25R8035F | 8 | 1.65V-3.6V | 1/2/4-bit |
| 6 | Macronix | MX25U8033F | 8 | 1.65V-2.0V | 1/2/4-bit |
| 7 | Macronix | MX25UM51245G | 512 | 1.7V-2V | 1/2/4/8-bit |

262 . Table13-1 SPI Flash Variants

The Direct Fetch mode is used to read instructions and data directly from Flash without any processor intervention. It supports inline decryption using an AES engine for the instructions stored in Flash. The Indirect Access mode is used to read and write data/instructions from Flash. The two modes - Direct Fetch and Indirect Access - can be used to access the same Flash or two different Flashes (using CSN0 and CSN1) at a time by enabling hardware controlled mode. The SPI Flash Controller has independent AHB slaves for these modes of access.

12.3.1 Programming sequence

Writing and Reading methods

Please refer to the "Quad Serial Peripheral Interface (QSPI)" section of the MCU SAPI Manual for details on how to use the SPI Flash Controller to Read and Write to Quad SPI Flash.

13 GPDMA

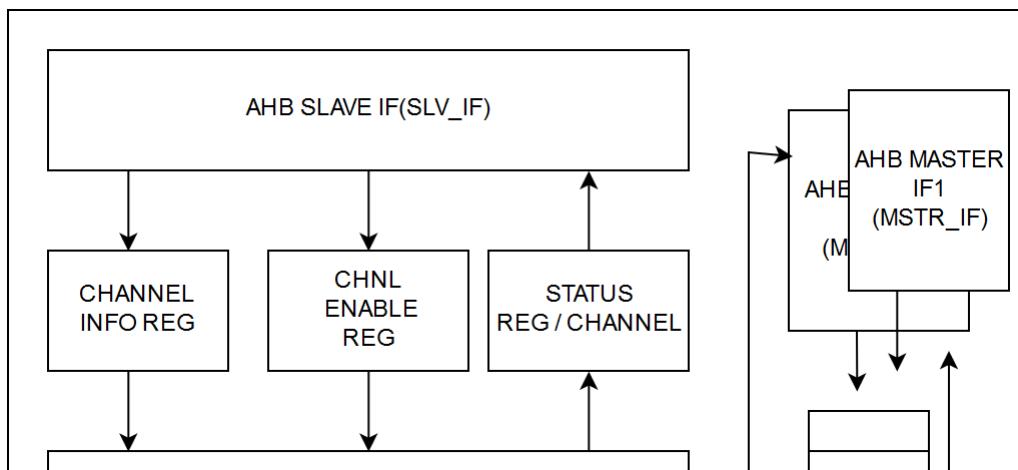
13.1 General Description

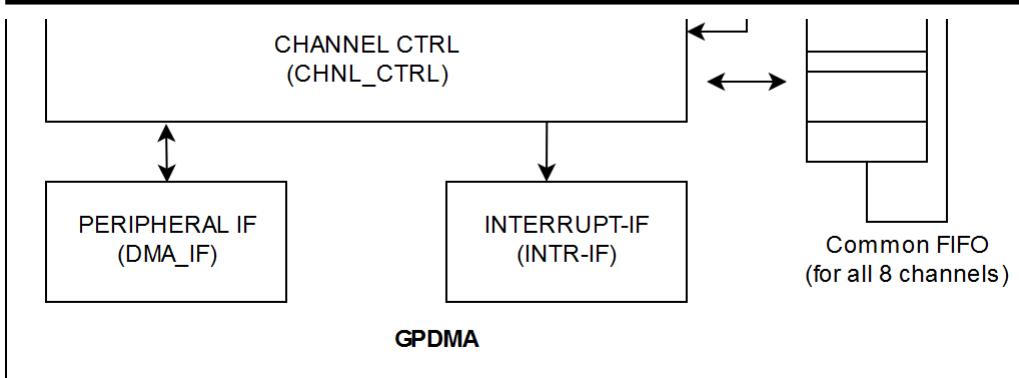
GPDMA is an AHB based general purpose DMA Controller core that transfers data from a source peripheral or memory to a destination peripheral or memory over one or more AHB buses. GPDMA has two masters interfaces. It can support 8 DMA channels. It can support at maximum of 64 peripherals.

13.2 Features

- Supports 8 channels (DMA links)
- Has two AHB masters for parallel data transfer.
- Dynamically size configurable 8 channel SRAM based 64 x 64bit FIFO.
- A maximum of 64 peripherals to be supported.
- Master selectable per channel and per source and destination.
- Selectable master for descriptor fetch.
- Supports programmable Source and destination burst sizes (beats): (1,2,3..63). Burst Size is the number of beats transferred on a peripheral request.
- Supports programmable AHB bursts in beats (1,4,8.. 32 beats).
- Source and Destination address alignment. Can support byte-aligned 16 and 32-bit transfers.
- Programmable Transfer Types
 - Memory to Memory
 - Memory to Peripheral(Peripheral or DMA controlled)
 - Peripheral to Memory (Peripheral or DMA controlled)
- Programmable transfer length in bytes per descriptor.
- Linked descriptors
 - Move to next descriptor on completion of transfer and when next descriptor address is not 0.
 - Move to next descriptor after last transfer indicated by Peripheral.
 - Supports address increment/no-increment
- Interrupt Generation Options
 - Generate at the end of the descriptor when indicated by a descriptor bit
 - When last transfer is indicated by the peripheral for peripheral controlled transfers
 - At the end of the overall transfer when next descriptor address is 0.
- Supports Programmable priority encoded arbiter
- DMA squash: DMA in progress will be having clean termination.(Any contents of the fifo will be lost).
- Support for memory Zero Fill and One Fill.
- Supports AHB slave interface for programming

13.3 Functional Description





GPDMA Block Diagram

GPDMA fetch data from the source and write same data into the destination according to the programmed source and destination locations. GPDMA has two AHB masters. So, fetching data from source location and writing same data into destination location can be done in parallel by using these two AHB masters. Once DMA transfer is done, status and interrupts are updated.

GPDMA supports the following three types transfers.

- Memory to Memory
- Memory to Peripheral (DMA controlled)
- Peripheral to Memory (Peripheral or DMA controlled)

MCU peripherals are assigned with following PERIPHERAL CODEs in GPDMA.

| PERIPHERAL Code | Peripheral |
|-----------------|----------------------------|
| 0 | SDIO MF(Source) |
| 1 | SDIO MF(Destination) |
| 2 | SSI Slave 1(Source) |
| 3 | SSI Slave 1(Destination) |
| 4 | Reserved |
| 5 | Reserved |
| 6 | Reserved |
| 7 | Reserved |
| 8 | UART 1(Source) |
| 9 | UART 1(Destination) |
| 10 | GSPI Master 1(Source) |
| 11 | GSPI Master 1(Destination) |
| 12 | Reserved |
| 13 | Reserved |
| 14 | I2C Slave1(Source) |
| 15 | I2C Slave 1(Destination) |
| 16 | QSPI 2(Source) |

| PERIPHERAL Code | Peripheral |
|------------------------|----------------------|
| 17 | QSPI 2(Destination) |
| 18-21 | Reserved |
| 22 | Reserved |
| 23 | Reserved |
| 24-25 | Reserved |
| 26 | SIO (Source) |
| 27 | SIO(Destination) |
| 28 | Reserved |
| 29 | Reserved |
| 30 | SDIO(Source) |
| 31 | SDIO(Destination) |
| 32 | Reserved |
| 33 | Reserved |
| 34 | USART 2(Source) |
| 35 | USART 2(Destination) |
| 36-45 | Reserved |
| 46 | CCI(Destination) |
| 47 | CRC(Destination) |
| 48-63 | Reserved |

263 . Table 14-1 Peripheral codes for GPDMA

13.4 Programming sequence

Before doing any thing, programmer must understand the dependability of transfer size on FIFO size. Based on FIFO size, programmer must select correct source/destination burst size and data width and AHB burst. Source/destination burst or ahb_burst must never exceed FIFO size. If programmer do not follow this rule, FIFO will over run or under run.

programming sequence is:

- Update channel SA/DA registers
- Update Link List Ptr Register
- Channel control register.
- Misc_channel_ctrl_register
- Channel and FIFO config register
- Config link list descriptors
- Write DMA channel enable register
- Depending on DMA transfer types, GPDMA perform DMA transfers.
- Once DMA transfer is done, status and interrupts are updated.

It supports two types of DMA transfers. They are

- Link listed mode

- non link listed mode

In case of link listed mode, MCU writes the link list fetch address in to LINK_LIST_PTR_REG. After the channel arbitration, Hardware fetches the descriptor from the address pointed by LINK_LIST_PTR_REG. Save the contents of the descriptor into appropriate control registers. Based on the control register settings, it perform DMA operation.

In case of non link listed operation, MCU writes the link list fetch address into LINK_LIST_PTR_REG as 0. MCU also writes all the control registers and SA/DA registers. After the channel arbitration, hard ware reads control registers. Based on the control register settings, it perform DMA operation.

13.4.1 Interrupt configurations of GPDMA

1. Mask Transfer done interrupt and Unmask Linked List descriptor Done Interrupt.

In this case, INTR_MASK_REG should be written with 0x00FF00FF and get LINK_LIST_FETCH_DONE interrupt.

2. Unmask Transfer done interrupt and Mask Linked List descriptor Done Interrupt.

In this case, INTR_MASK_REG should be written with 0x0000FFFF and get TFR_DONE interrupt

3. Mask Transfer done interrupt and Linked List descriptor Done Interrupts

In this case, INTR_MASK_REG should be written with 0x00FFFFFF. Any interrupts and their status in INTR_STAT_REG are not observed because of their masking.

4. Unmask Transfer done and Linked List descriptor Done Interrupts.

In this case, INTR_MASK_REG should be written with 0x000000FF. Here LINK_LIST_FETCH_DONE interrupt is observed after every link descriptor fetch done and TFR_DONE interrupt after completion of last link transfer.

5. Unmask Transfer done and Linked List descriptor Done Interrupts and Mask GPDMA INT ENABLE to NVIC

In this case, interrupt is masked from source GPDMA to processor by masking at M4SS_GPDMA_INTR_SEL register so that the status register bits should be polled for interrupt status.

Flow control error

When flow control error happens, rises an interrupt if interrupt is not masked, error bit for the channel is set and DMA will be terminated and channel enable is reset for this channel. FIFOs will be reset and a new arb grant is requested.

13.4.2 Transfer Size less than burst size error

For a non link listed dma transfers, it is expected transfer size is always equal to data_width x burst_size . If this condition is not met, this error will terminate DMA transfers and channel enable will be reset for this channel. Also rises an interrupt if interrupt is not masked. FIFOs will be reset and a new arb grant is requested.

13.4.3 FIFO Re-Configuration

Dynamic resize of the FIFO is supported in GPDMA. In order to do this, Programmer needs to know if all the channels are cleared. Alternatively, programmer can set DMA_SQUASH bit in channel control register. This ensures clean termination of the FIFO and hardware functionality. A done bit will be provided after completion of this operation.

13.5 Register Summary

Base Address: 0x0x2108_0000

| Register Name | Offset | Description |
|---------------------|--------|-----------------------------|
| INTERRUPT_REG | 0x1084 | Interrupt Register |
| INTERRUPT_MASK_REG | 0x1088 | Interrupt Mask Register |
| INTERRUPT_STAT_REG | 0x108C | Interrupt Status Register |
| DMA_CHNL_ENABLE_REG | 0x1090 | DMA Channel Enable Register |

| Register Name | Offset | Description |
|------------------------------|--------|--|
| DMA_CHNL_SQUASH_REG | 0x1094 | DMA Channel Squash Register |
| DMA_CHNL_LOCK_REG | 0x1098 | DMA Channel Lock Register |
| LINK_LIST_PTR_REG_CHNL_0 | 0x1004 | Pointer Register of Channel 0 |
| SRC_ADDR_REG_CHNL_0 | 0x1008 | Source Address Register of Channel 0 |
| DEST_ADDR_REG_CHNL_0 | 0x100C | Destination Address Register of Channel 0 |
| CHANNEL_CTRL_REG_CHNL_0 | 0x1010 | Channel Control Register for Channel 0 |
| MISC_CHANNEL_CTRL_REG_CHNL_0 | 0x1014 | Miscellaneous Control Register for Channel 0 |
| FIFO_CONFIG_REG_CHNL_0 | 0x1018 | FIFO Configuration Register for Channel 0 |
| PRIORITY_LEVEL_REG_CHNL_0 | 0x101C | Priority Level for Channel 0 |
| LINK_LIST_PTR_REG_CHNL_1 | 0x1104 | Pointer Register of Channel 1 |
| SRC_ADDR_REG_CHNL_1 | 0x1108 | Source Address Register of Channel 1 |
| DEST_ADDR_REG_CHNL_1 | 0x110C | Destination Address Register of Channel 1 |
| CHANNEL_CTRL_REG_CHNL_1 | 0x1110 | Channel Control Register for Channel 1 |
| MISC_CHANNEL_CTRL_REG_CHNL_1 | 0x1114 | Miscellaneous Control Register for Channel 1 |
| FIFO_CONFIG_REG_CHNL_1 | 0x1118 | FIFO Configuration Register for Channel 1 |
| PRIORITY_LEVEL_REG_CHNL_1 | 0x111C | Priority Level for Channel 1 |
| LINK_LIST_PTR_REG_CHNL_2 | 0x1204 | Pointer Register of Channel 2 |
| SRC_ADDR_REG_CHNL_2 | 0x1208 | Source Address Register of Channel 2 |
| DEST_ADDR_REG_CHNL_2 | 0x120C | Destination Address Register of Channel 2 |
| CHANNEL_CTRL_REG_CHNL_2 | 0x1210 | Channel Control Register for Channel 2 |
| MISC_CHANNEL_CTRL_REG_CHNL_2 | 0x1214 | Miscellaneous Control Register for Channel 2 |
| FIFO_CONFIG_REG_CHNL_2 | 0x1218 | FIFO Configuration Register for Channel 2 |
| PRIORITY_LEVEL_REG_CHNL_2 | 0x121C | Priority Level for Channel 2 |
| LINK_LIST_PTR_REG_CHNL_3 | 0x1304 | Pointer Register of Channel 3 |
| SRC_ADDR_REG_CHNL_3 | 0x1308 | Source Address Register of Channel 3 |
| DEST_ADDR_REG_CHNL_3 | 0x130C | Destination Address Register of Channel 3 |
| CHANNEL_CTRL_REG_CHNL_3 | 0x1310 | Channel Control Register for Channel 3 |
| MISC_CHANNEL_CTRL_REG_CHNL_3 | 0x1314 | Miscellaneous Control Register for Channel 3 |
| FIFO_CONFIG_REG_CHNL_3 | 0x1318 | FIFO Configuration Register for Channel 3 |
| PRIORITY_LEVEL_REG_CHNL_3 | 0x131C | Priority Level for Channel 3 |
| LINK_LIST_PTR_REG_CHNL_4 | 0x1404 | Pointer Register of Channel 4 |
| SRC_ADDR_REG_CHNL_4 | 0x1408 | Source Address Register of Channel 4 |

| Register Name | Offset | Description |
|------------------------------|--------|--|
| DEST_ADDR_REG_CHNL_4 | 0x140C | Destination Address Register of Channel 4 |
| CHANNEL_CTRL_REG_CHNL_4 | 0x1410 | Channel Control Register for Channel 4 |
| MISC_CHANNEL_CTRL_REG_CHNL_4 | 0x1414 | Miscellaneous Control Register for Channel 4 |
| FIFO_CONFIG_REG_CHNL_4 | 0x1418 | FIFO Configuration Register for Channel 4 |
| PRIORITY_LEVEL_REG_CHNL_4 | 0x141C | Priority Level for Channel 4 |
| LINK_LIST_PTR_REG_CHNL_5 | 0x1504 | Pointer Register of Channel 5 |
| SRC_ADDR_REG_CHNL_5 | 0x1508 | Source Address Register of Channel 5 |
| DEST_ADDR_REG_CHNL_5 | 0x150C | Destination Address Register of Channel 5 |
| CHANNEL_CTRL_REG_CHNL_5 | 0x1510 | Channel Control Register for Channel 5 |
| MISC_CHANNEL_CTRL_REG_CHNL_5 | 0x1514 | Miscellaneous Control Register for Channel 5 |
| FIFO_CONFIG_REG_CHNL_5 | 0x1518 | FIFO Configuration Register for Channel 5 |
| PRIORITY_LEVEL_REG_CHNL_5 | 0x151C | Priority Level for Channel 5 |
| LINK_LIST_PTR_REG_CHNL_6 | 0x1604 | Pointer Register of Channel 6 |
| SRC_ADDR_REG_CHNL_6 | 0x1608 | Source Address Register of Channel 6 |
| DEST_ADDR_REG_CHNL_6 | 0x160C | Destination Address Register of Channel 6 |
| CHANNEL_CTRL_REG_CHNL_6 | 0x1610 | Channel Control Register for Channel 6 |
| MISC_CHANNEL_CTRL_REG_CHNL_6 | 0x1614 | Miscellaneous Control Register for Channel 6 |
| FIFO_CONFIG_REG_CHNL_6 | 0x1618 | FIFO Configuration Register for Channel 6 |
| PRIORITY_LEVEL_REG_CHNL_6 | 0x161C | Priority Level for Channel 6 |
| LINK_LIST_PTR_REG_CHNL_7 | 0x1704 | Pointer Register of Channel 7 |
| SRC_ADDR_REG_CHNL_7 | 0x1708 | Source Address Register of Channel 7 |
| DEST_ADDR_REG_CHNL_7 | 0x170C | Destination Address Register of Channel 7 |
| CHANNEL_CTRL_REG_CHNL_7 | 0x1710 | Channel Control Register for Channel 7 |
| MISC_CHANNEL_CTRL_REG_CHNL_7 | 0x1714 | Miscellaneous Control Register for Channel 7 |
| FIFO_CONFIG_REG_CHNL_7 | 0x1718 | FIFO Configuration Register for Channel 7 |
| PRIORITY_LEVEL_REG_CHNL_7 | 0x171C | Priority Level for Channel 7 |

264 . Register Summary Table

13.6 Register Description

13.6.1 INTERRUPT_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|-------------|
| 31:8 | R | Reserved | - | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------|-------------|---|
| 7:0 | R/W | GPDMAC_INT_STAT | 0x00 | <p>This bit indicates the status of transfer done interrupt or linked link descriptor fetch interrupt.</p> <p>Transfer done interrupt is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled</p> <p>Linked list descriptor fetch interrupt is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) is enabled.</p> <p>Bit 0 = 1: channel 0- interrupt event</p> <p>.....</p> <p>Bit 7 = 1: channel 7, interrupt event has happened.</p> <p>To clear this interrupt, register write data from CPU is set to "1" and actual register value is set to '0'(inverted value of write data). That means, if this register is read back, '0' will be read upon writing "1" from CPU.</p> <p>Clearing up this interrupt also resets the values of INTERRUPT_STAT_REG bits for that corresponding channel.</p> |

265 . Interrupt Register Description

13.6.2 INTERRUPT_MASK_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|--|
| 31:24 | R/W | RSVD | 8'h0 | Reserved |
| 23:16 | R/W | TFR_DONE_MASK | 8'h0 | <p>Transfer done interrupt bit mask control. By default, transfer done interrupt is unmasked. 0 stands for unmask and 1 stands for mask.</p> <p>16-bit – Channel 0</p> <p>17-bit – Channel 1</p> <p>18-bit – Channel 2</p> <p>19-bit – Channel 3</p> <p>20-bit – Channel 4</p> <p>21-bit – Channel 5</p> <p>22-bit – Channel 6</p> <p>23-bit – Channel 7</p> |

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------|-------------|---|
| 15:8 | R/W | LINK_LIST_FETCH_MASK | 8'hff | <p>Linked list fetch done interrupt bit mask control. By default, descriptor fetch done interrupt is masked.</p> <p>Each bit is used to mask the interrupt per channel. Writing value “1” in to the bit MASKS that particular interrupt.</p> <ul style="list-style-type: none"> 8-bit – Channel 0 9-bit – Channel 1 10-bit – Channel 2 11-bit – Channel 3 12-bit – Channel 4 13-bit – Channel 5 14-bit – Channel 6 15-bit – Channel 7 |
| 7:0 | R/W | Reserved | 8'hff | Reserved |

266 . Interrupt Mask Register Description

13.6.3 INTERRUPT_STAT_REG

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------------|-------------|---|
| 31 | R | gpdmaC_ERR7 | 1'b0 | (1) transfer size/burst size /h size mismatch (2)flow_ctrl_err |
| 30 | R | TFR_DONE7 | 1'b0 | <p>This bit indicates the status of DMA transfer done interrupt for channel 7.</p> <p>This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled.</p> |
| 29 | R | LINK_LIST_FETCH_DONE7 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 7. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 28 | R | HRESP_ERR7 | 1'b0 | 1: channel 7 ,dma error |
| 27 | R | gpdmaC_ERR6 | 1'b0 | (1) transfer size/burst size /h size mismatch (2) flow_ctrl_err |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 26 | R | TFR_DONE6 | 1'b0 | This bit indicates the status of DMA transfer done interrupt for channel 6. This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled. |
| 25 | R | LINK_LIST_FETCH_DO_NE6 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 6. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 24 | R | HRESP_ERR6 | 1'b0 | 1: channel 6, dma error |
| 23 | R | gpdmaC_ERR5 | 1'b0 | (1) transfer size/burst size /h size mismatch (2) flow_ctrl_err |
| 22 | R | TFR_DONE5 | 1'b0 | This bit indicates the status of DMA transfer done interrupt for channel 5. This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled. |
| 21 | R | LINK_LIST_FETCH_DO_NE5 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 5. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 20 | R | HRESP_ERR5 | 1'b0 | 1: channel 5, dma error |
| 19 | R | gpdmaC_ERR4 | 1'b0 | (1) transfer size/burst size /h size mismatch (2) flow_ctrl_err |
| 18 | R | TFR_DONE4 | 1'b0 | This bit indicates the status of DMA transfer done interrupt for channel 4. This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled. |
| 17 | R | LINK_LIST_FETCH_DO_NE4 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 4. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 16 | R | HRESP_ERR4 | 1'b0 | 1: channel 4, dma error |
| 15 | R | gpdmaC_ERR3 | 1'b0 | (1) transfer size/burst size /h size mismatch (2) flow_ctrl_err |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 14 | R | TFR_DONE3 | 1'b0 | This bit indicates the status of DMA transfer done interrupt for channel 3. This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled. |
| 13 | R | LINK_LIST_FETCH_DO_NE3 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 3. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 12 | R | HRESP_ERR3 | 1'b0 | 1: channel 3, dma error |
| 11 | R | gpdmaC_ERR2 | 1'b0 | (1) transfer size/burst size /h size mismatch (2) flow_ctrl_err |
| 10 | R | TFR_DONE2 | 1'b0 | This bit indicates the status of DMA transfer done interrupt for channel 2. This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled. |
| 9 | R | LINK_LIST_FETCH_DO_NE2 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 2. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 8 | R | HRESP_ERR2 | 1'b0 | 1: channel 2, dma error |
| 7 | R | gpdmaC_ERR1 | 1'b0 | (1) transfer size/burst size /h size mismatch (2) flow_ctrl_err |
| 6 | R | TFR_DONE1 | 1'b0 | This bit indicates the status of DMA transfer done interrupt for channel 1. This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled. |
| 5 | R | LINK_LIST_FETCH_DO_NE1 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 1. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 4 | R | HRESP_ERR1 | 1'b0 | 1: channel 1, dma error |
| 3 | R | gpdmaC_ERR0 | 1'b0 | (1) transfer size/burst size /h size mismatch (2) flow_ctrl_err |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------------|-------------|---|
| 2 | R | TFR_DONE0 | 1'b0 | This bit indicates the status of DMA transfer done interrupt for channel 0. This bit is set only when corresponding TFR_DONE_MASK(controlled via INTERRUPT_MASK_REG register) is enabled. |
| 1 | R | LINK_LIST_FETCH_DO NE0 | 1'b0 | This bit indicates the status of linked list descriptor fetch done for channel 0. This bit is set only when corresponding LINK_LIST_FETCH_MASK(controlled via INTERRUPT_MASK_REG register) is enabled and LINK_INTERRUPT(controlled via CHANNEL_CTRL_REG_CHNL_n register) |
| 0 | R | HRESP_ERR0 | 1'b0 | 1: channel 0, dma error |

267 . Interrupt Status Register Description

13.6.4 DMA_CHNL_ENABLE_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:8 | R | Reserved | | Reserved |
| 7:0 | R/W | CH_ENB | 0x00 | When a bit is set to one, it indicates, corresponding channel is enabled for dma operation. "CPU Will be masked to write zeros, CPU is allowed write 1 only. Hardware will set these bits to zero when the DMA operation is done. These bits feed in to channel arbitration logic. Bit 0 = 1: channel 0 enabled Bit 7 = 1: channel 7 is enabled |

268 . DMA Channel Enable Register Description

13.6.5 DMA_CHNL_SQUASH_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:8 | R | Reserved | | Reserved |
| 7:0 | R/W | CH_DIS | 0x00 | CPU Will be masked to write zeros, CPU is allowed write 1 only. Hardware will set these bits to zero when the DMA Squash is done. Setting this bit ensures clean termination of fifo, fsm, etc. Bit 0 = 1: Channel 1 fifo will be cleared, FSM will be reset. Bit 7 = 1: Channel 1 fifo will be cleared, FSM will be reset. |

269 . DMA Channel Squash Register Description

13.6.6 DMA_CHNL_LOCK_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------|-------------|--|
| 31:8 | R | Reserved | | Reserved |
| 7:0 | R/W | CHNL_LOCK | 0x00 | When set entire DMA block transfer is done, before other DMA request is serviced. When set entire DMA transfer is done, before other DMA request is serviced. This bit is reset to zero up on completion of DMA. |

270 . DMA Channel Lock Register Description

13.6.7 LINK_LIST_PTR_REG_CHNL_n

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 31:0 | R/W | LINK_LIST_POINTER | 0x00 | This is the address of the memory location from which next descriptor is obtained |

271 . Link List Pointer Register Description

13.6.8 SRC_ADDR_REG_CHNL_n

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:0 | R/W | SRC_ADDR | 0x00 | This is the source address from which the data is fetched |

272 . Source Address Register Description

13.6.9 DEST_ADDR_REG_CHNL_n

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------|-------------|--|
| 31:0 | R/W | DEST_ADDR | 0x00 | This is the destination address from which the data is fetched |

273 . Destination Address Register Description

13.6.10 CHANNEL_CTRL_REG_CHNL_n

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------|-------------|--|
| 31 | R | Reserved | 0x0 | Reserved |
| 30 | R/W | DEST_FIFO_MODE | 0x0 | If set to 1; destination address will not be incremented(means fifo mode for destination) |
| 29 | R/W | SRC_FIFO_MODE | 0x0 | If set to 1; source address will not be incremented(means fifo mode for source) |
| 28 | R/W | LINK_INTERRUPT | 0x0 | This bit is set in link list descriptor. Hard ware will send an interrupt when the DMA transfer is done for the corresponding link list address. |
| 27 | R/W | RETRY_ON_ERROR | 0x0 | When this bit is set, if we recieve HRESPERR, We will retry the DMA for that channel. |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------|-------------|---|
| 26 | R/W | DEST_ADDR_CONTIG_UOUS | 0x0 | 1: Indicates Address is contiguous from previous |
| 25 | R/W | SRC_ADDR_CONTIGUOUS | 0x0 | 1: Indicates Address is contiguous from previous |
| 24 | R/W | LINK_LIST_MSTR_SEL | 0x0 | 0 : M0 will be used to fetch desc. 1: M1 will be used to fetch desc |
| 23 | R/W | LINK_LIST_ON | 0x0 | This mode is set, when we do link listed operation. |
| 22 | R/W | SRC_ALIGN | 0x0 | Reserved.Value set to 0 We do not do any singles. We just do burst, save first 3 bytes in to residue buffer in one cycle, In the next cycle send 4 bytes to fifo, save 3 bytes in to residue. This continues on. |
| 21:20 | R/W | SRC_DATA_WIDTH | 0x0 | Data transfer to destination. 00: 08 bits of data on the bus 01: 16 bits of data on the bus 10: 32 bits of data on the bus 11: reserved |
| 19:18 | R/W | DEST_DATA_WIDTH | 0x0 | Data transfer to destination. 00: 08 bits of data on the bus 01: 16 bits of data on the bus 10: 32 bits of data on the bus 11: reserved |
| 17 | R/W | MSTR_IF_SEND_SEL | 0x0 | This selects the MASTER IF from which data to be sent 0: MSTR-0 for send (to destination) 1: MSTR-1 for send (to destination) |
| 16 | R/W | MSTR_IF_FETCHSEL | 0x0 | This selects the MASTER IF from which data to be fetched . 0: MSTR-0 for fetch (from src) 1: MSTR-1 for fetch (from src) |
| 15:14 | R/W | DMA_FLOW_CTRL | 0x0 | 00: gpdmaC :can be set for any type of transfers 01: source peripheral : typically set for peripheral to memory 10: destination peripheral : typically set for memory to peripheral 11: src_and_dest peripheral : Typically set for peripheral to peripheral |
| 13:12 | R/W | TRNS_TYPE | 0x0 | DMA transfer type 00: memory to memory 01: memory to peripheral 10: peripheral to memory 11: peripheral to peripheral |

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------|-------------|---|
| 11:0 | R/W | DMA_BLK_SIZE | 0x000 | <p>This is data to be transmitted.</p> <p>User should program non-zero value.</p> <p>Loaded at the beginning of the DMA transfer and decremented at every dma transaction.</p> <p>Zero size length is not supported. If zero is programmed, DMA will get stuck.</p> |

274 . Channel Control Register Description

13.6.11 MISC_CHANNEL_CTRL_REG_CHNL_n

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|--|
| 31 | R/W | Mem_one_fill | 0x0 | Select for memory filling with either 1's or 0's. 0 – Memory fill with 0's. 1 - Memory fill with 1's. |
| 30 | R/W | Mem_fill_enable | 0x0 | Enable for memory filling with either 1's or 0's. 0 – Disabled 1 – Enabled the memory filling When memory fill is enabled, transfer size has to be multiples of hsize (destination data-width). This is a mandate requirement. |
| 29:27 | R/W | DMA_PROT | 0x0 | Protection level to go with the data. It will be concatenated with 1'b1 as there will be no opcode fetching and directly assign to hprot in AHB interface. |
| 26:21 | R/W | SRC_CHNL_ID | 0x0 | This is the source channel Id, from which the data is fetched. must be set up prior to DMA_CHANNEL_ENABLE |
| 20:15 | R/W | DEST_CHNL_ID | 0x0 | This is the destination channel Id to which the data is sent. Must be set up prior to DMA_CHANNEL_ENABLE |
| 14:9 | R/W | SRC_DATA_BURST | 0x0 | Burst writes in beats from source 000000: 64 beats 000001: 1 beat 001111: 15 beats 111111: 63 beats |
| 8:3 | R/W | DEST_DATA_BURST | 0x0 | Burst writes in beats to destination. 000000: 64 beats .000001: 1 beat 001111: 15 beats 111111: 63 beats |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 2:0 | R/W | AHB_BURST_SIZE | 0x0 | 000 : 1 beat 001 : 4 beat 010 : 8 011 : 16 100 : 20 101: 24 110: 28 111 : 32 |

275 .Miscellaneous Channel Control Register Description

13.6.12 FIFO_CONFIG_REG_CHNL_n

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 31:12 | R | Reserved | 0x0 | Reserved |
| 11:6 | R/W | FIFO_SIZE | 0x00 | Channel size. Configure FIFO size before starting DMA operation |
| 5:0 | R/W | FIFO_STRT_ADDR | 0x00 | Starting row address of channel |

276 .FIFO Configuration Register Description

13.6.13 PRIORITY_LEVEL_REG_CHNL_n

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| 31:2 | R | Reserved | 0x0 | Reserved |
| 1:0 | R/W | PRIORITY_CH_n | 0x00 | Indicates the level of priority Four levels of priority is supported 00 – first level priority 01 – second level priority 10 – third level priority 11 – four level priority |

277 .Priority Level Channel Register Description

14 Micro DMA (uDMA)

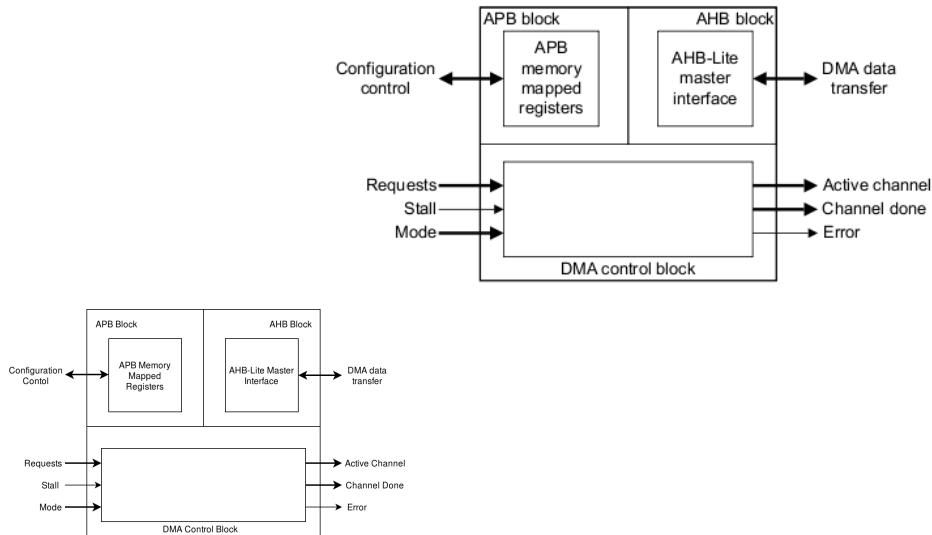
14.1 General Description

The μDMA is an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-Chip (SoC) peripheral. It is a very low gate count DMA controller that is compatible with the AMBA AHB-Lite protocol.

14.2 Features

- It is compatible with AHB-Lite for the DMA transfers.
- It is compatible with APB for programming the registers.
- It has a single AHB-Lite master for transferring data using a 32-bit address bus and 32-bit data bus.
- It has 32 channels. Out of which last 24 are dedicated channels for particular peripherals. First 8 channels can support 32 different peripherals.
- Each DMA channel has dedicated handshake signals.
- Each DMA channel has a programmable priority level.
- Each priority level arbitrates using a fixed priority that is determined by the DMA channel number.
- It supports multiple transfer types: memory-to-memory ,memory-to-peripheral ,peripheral-to-memory.
- It supports multiple DMA cycle types.
- It supports multiple DMA transfer data widths.
- Each DMA channel can access a primary, and alternate, channel control data structure.
- All the channel control data is stored in system memory in little-endian format.
- It performs all DMA transfers using the SINGLE AHB-Lite burst type.
- The destination data width is equal to the source data width.
- The number of transfers in a single DMA cycle can be programmed from 1 to 1024.
- The transfer address increment can be greater than the data width.
- It has a single output to indicate when an ERROR condition occurs on the AHB bus.

14.3 Functional Description



7 . UDMA Block Diagram

uDMA has DMA control block, uDMA control and status register block and AHB-Lite master as shown in Figure1. The uDMA control and status register block contains the registers that enable to configure the controller by using the

APB slave interface. The AHB-Lite master transfer data from a source AHB slave to a destination AHB slave using a 32-bit data bus. DMA control block do the following functions:

- arbitrates the incoming requests
- indicates which channel is active
- indicates when a channel is complete
- indicates when an ERROR has occurred on the AHB-Lite interface
- enables slow peripherals to stall the completion of a DMA cycle
- waits for a request to clear before completing a DMA cycle
- performs multiple or single DMA transfers for each request

All MCU peripherals are assigned with following PERIPHERAL CODEs in uDMA.

| PERIPHERAL CODE | Acting as Source(Channel Number) | Acting as Destination(Channel Number) |
|-----------------|----------------------------------|---------------------------------------|
| CT0 | 0(CT0_0 - destination) | 1(CT0_1 - destination) |
| I2C 2 | 2 | 3 |
| CCI/CRC | 4(CCI - destination) | 5(CRC - destination) |
| SOC PLL SPI | 6 | 7 |
| CT1 | 8(CT1_0 - destination) | 9(CT1_1 - destination) |
| GSPI Master 1 | 10 | 11 |
| SIO | 12 | 13 |
| I2S Channel 0 | 14 | 15 |
| I2S Channel 1 | 16 | 17 |
| SDIO | 18 | 19 |
| QSPI | 20 | 21 |
| SSI Slave | 22 | 23 |
| USART 1 | 24 | 25 |
| UART 2 | 26 | 27 |
| SSI Master | 28 | 29 |
| I2C | 30 | 31 |

14.4 Programming Sequence

The following are the steps needed to program uDMA to do transfers.

- 1) First program the source end address.
- 2) Program the destination end address.
- 3) Program the configuration information (the type of transfer and no of transfers.).
- 4) Enable the particular channel for programming by writing to '1' at particular bit in enable register.
- 5) Enable the controller by writing '1' at 0th position of dma_config register.
- 6) Now give the request to transfer. Either write the '1' at particular position of channel software request register or through dma_req or dma_sreq.

14.5 Register Summary

Base Address: 0x4403_0000

| Register Name | Offset | Description |
|----------------------|--------|---|
| dma_status | 0x000 | DMA Status Register |
| dma_cfg | 0x004 | DMA Configuration Register |
| ctrl_base_ptr | 0x008 | Channel Control Data Base Pointer |
| alt_ctrl_base_ptr | 0x00C | Channel Alternate Control Data Base Pointer |
| dma_waitonreq_status | 0x010 | Channel Wait on request status |
| chnl_sw_request | 0x014 | Channel Software Request |
| chnl_useburst_set | 0x018 | UDMA Channel useburst set |
| chnl_useburst_clr | 0x01C | UDMA Channel useburst clear |
| chnl_req_mask_set | 0x020 | UDMA Channel request mask set |
| chnl_req_mask_clr | 0x024 | UDMA Channel request mask clear |
| chnl_enable_set | 0x028 | UDMA Channel enable register |
| chnl_enable_clr | 0x02C | UDMA Channel enable clear register |
| chnl_pri_alt_set | 0x030 | UDMA Channel primary -alternate set |
| chnl_pri_alt_clr | 0x034 | UDMA Channel primary -alternate clear |
| chnl_priority_set | 0x038 | UDMA Channel Priority Set |
| chnl_priority_clr | 0x03C | UDMA Channel Priority Clear |
| err_clr | 0x04C | UDMA Bus Error Clear Register |
| skip_desc_fetch | 0x050 | UDMA Skip Descriptor Register |
| UDMA_done_status | 0x800 | UDMA Done Status Register |
| Channel_Status | 0x804 | Channel Status Register |
| UDMA_config_ctrl | 0x828 | UDMA Configuration Control Register |

278 . Register Summary Table

14.6 Register Description

14.6.1 DMA_STATUS

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|---|
| 31:28 | R | test_status | 0x0 | To reduce the gate count you can configure the controller, to exclude the integration test logic. Read as: 0x0 - controller does not include the integration test logic 0x1 - controller includes the integration test logic 0x2 - 0xF - undefined. |
| 27:21 | R | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|---|
| 20:16 | R | chnls_minus1 | | Number of available DMA channels minus one. For example: b00000 - controller configured to use 1 DMA channel b00001 - controller configured to use 2 DMA channels b00010 - controller configured to use 3 DMA channels . . . b11111 - controller configured to use 32 DMA channels. |
| 15:8 | R | Reserved | 0x0 | Reserved |
| 7:4 | R | state | 0x0 | Current state of the control state machine. State can be one of the following: b0000 - idle b0001 - reading channel controller data b0010 - reading source data end pointer b0011 - reading destination data end pointer b0100 - reading source data b0101 - writing destination data b0110 - waiting for DMA request to clear b0111 - writing channel controller data b1000 - stalled b1001 - done b1010 - peripheral scatter-gather transition b1011-b1111 - undefined. |
| 3:1 | R | Reserved | 0x0 | Reserved |
| 0 | R | master_enable | 0x0 | Enable status of the controller: 0 - controller is disabled 1 - controller is enabled. |

279 . DMA Status Register Description

14.6.2 DMA_CFG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------|-------------|--|
| 31:8 | W | Reserved | 0x0 | Undefined. Write as zero. |
| 7:5 | W | chnl_prot_ctrl | | Sets the AHB-Lite protection by controlling the HPROT[3:1] signal levels as follows: Bit [7] -Controls HPROT[3] to indicate if a cacheable access is occurring. Bit [6] -Controls HPROT[2] to indicate if a bufferable access is occurring. Bit [5] -Controls HPROT[1] to indicate if a privileged access is occurring. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------|-------------|--|
| 4:1 | W | Reserved | 0x0 | Undefined. Write as zero. |
| 0 | W | master_enable | 0x0 | Enable for the controller: 0 - disables the controller 1 - enables the controller. |

280 . DMA Configuration Register Description

14.6.3 CTRL_BASE_PTR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|--|
| 31:10 | R/W | ctrl_base_ptr | 0x0 | Pointer to the base address of the primary data structure. |
| 4:0 | W | Reserved | 0x0 | Undefined. Write as zero. |

281 . Channel Control Data Base Pointer Register Description

14.6.4 ALT_CTRL_BASE_PTR

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:0 | R | alt_ctrl_base_ptr | 0x0 | Base address of the alternate data structure |

282 . Channel Alternate Control Data Base Pointer Register Description

14.6.5 DMA_WAITONREQUEST_STATUS

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------|-------------|---|
| 31:0 | R | dma_waitonreq_stat | 0x0 | Per Channel wait on request status(where C specifies channel number). Read as: Bit [C] - 0 dma_waitonreq[C] is LOW. Bit [C] - 1 dma_waitonreq[C] is HIGH. |

283 . Channel Wait On Request Status Register Description

14.6.6 CHNL_SW_REQUEST

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|--|
| 31:0 | W | chnl_sw_request | 0x0 | Set the appropriate bit to generate a software DMA request on the corresponding DMA channel(C specifies channel number). Write as: Bit [C] - 0 Does not create a DMA request for channel C. Bit [C] - 1 Creates a DMA request for channel C. Writing to a bit where a DMA channel is not implemented does not create a DMA request for that channel. |

284 . Channel Software Request Register Description

14.6.7 CHNL_USEBURST_SET

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 31:0 | R/W | Chnl_useburst_set | 0x0 | <p>Returns the useburst status, or disables dma_sreq[C] from generating DMA requests.</p> <p>Read as:</p> <p>Bit [C] - 0 DMA channel C responds to requests that it receives on dma_req[C] or dma_sreq[C]. The controller performs 2R, or single, bus transfers.</p> <p>Bit [C] - 1 DMA channel C does not respond to requests that it receives on dma_sreq[C]. The controller only responds to dma_req[C] requests and performs 2R transfers.</p> <p>Write as:</p> <p>Bit [C] - 0 No effect. Use the chnl_useburst_clr Register to set bit [C] to 0.</p> <p>Bit [C] - 1 Disables dma_sreq[C] from generating DMA requests. The controller performs 2R transfers.</p> <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> |

285 . UDMA Channel Useburst Set Register Description

14.6.8 CHNL_USEBURST_CLR

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:0 | W | chnl_useburst_clr | 0x0 | <p>Set the appropriate bit to enable dma_sreq[] to generate requests.</p> <p>Write as:</p> <p>Bit [C] - 0 No effect. Use the chnl_useburst_set Register to disable dma_sreq[] from generating requests.</p> <p>Bit [C] - 1 Enables dma_sreq[C] to generate DMA requests.</p> <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> |

286 . UDMA Channel Useburst Clear Register Description

14.6.9 CHNL_REQ_MASK_SET

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 31:0 | R/W | chnl_req_mask_set | 0x0 | <p>Returns the request mask status of dma_req[] and dma_sreq[], or disables the corresponding channel from generating DMA requests.</p> <p>Read as:</p> <p>Bit [C] - 0 External requests are enabled for channel C.</p> <p>Bit [C] - 1 External requests are disabled for channel C.</p> <p>Write as:</p> <p>Bit [C] - 0 No effect. Use the chnl_req_mask_clr Register to enable DMA requests.</p> <p>Bit [C] - 1 Disables dma_req[C] and dma_sreq[C] from generating DMA requests.</p> <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> |

287 . UDMA Channel Request Mask Set Register Description

14.6.10 CHNL_REQ_MASK_CLR

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 31:0 | W | chnl_req_mask_clr | 0x0 | <p>Set the appropriate bit to enable DMA requests for the channel corresponding to dma_req[] and dma_sreq[].</p> <p>Write as:</p> <p>Bit [C] - 0 No effect. Use the chnl_req_mask_set Register to disable dma_req[] and dma_sreq[] from generating requests.</p> <p>Bit [C] - 1 Enables dma_req[C] or dma_sreq[C] to generate DMA requests.</p> <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> |

288 . UDMA Channel Request Mask Clear Register Description

14.6.11 CHNL_ENABLE_SET

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|---|
| 31:0 | R/W | chnl_enable_set | 0x0 | <p>Returns the enable status of the channels, or enables the corresponding channels.</p> <p>Read as:</p> <ul style="list-style-type: none"> Bit [C] - 0 Channel C is disabled. Bit [C] - 1 Channel C is enabled. <p>Write as:</p> <ul style="list-style-type: none"> Bit [C] - 0 No effect. Use the chnl_enable_clr Register to disable a channel. Bit [C] - 1 Enables channel C. <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> |

289 . UDMA Channel Enable Set Register Description

14.6.12 CHNL_ENABLE_CLR

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|--|
| 31:0 | W | chnl_enable_clr | 0x0 | <p>Set the appropriate bit to disable the corresponding DMA channel.</p> <p>Write as:</p> <ul style="list-style-type: none"> Bit [C] - 0 No effect. Use the chnl_enable_set Register to enable DMA channels. Bit [C] - 1 Disables channel C. <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> <p>Note</p> <p>The controller disables a channel, by setting the appropriate bit, when either:</p> <ul style="list-style-type: none"> • it completes the DMA cycle • it reads a channel_cfg memory location which has cycle_ctrl - b000 • an ERROR occurs on the AHB-Lite bus. |

290 . UDMA Channel Enable Clear Register Description

14.6.13 CHNL_PRI_ALT_SET

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------|-------------|--|
| 31:0 | R/W | chnl_pri_alt_set | 0x0 | <p>Returns the channel control data structure status, or selects the alternate data structure for the corresponding DMA channel.</p> <p>Read as:</p> <ul style="list-style-type: none"> Bit [C] - 0 DMA channel C is using the primary data structure. Bit [C] - 1 DMA channel C is using the alternate data structure. <p>Write as:</p> <ul style="list-style-type: none"> Bit [C] - 0 No effect. Use the chnl_pri_alt_clr Register to set bit [C] to 0. Bit [C] - 1 Selects the alternate data structure for channel C. <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> <p>Note</p> <p>The controller toggles the value of the chnl_pri_alt_set [C] bit after it completes:</p> <ul style="list-style-type: none"> the four transfers that the primary data structure specifies for a memory scatter-gather, or peripheral scatter-gather, DMA cycle all the transfers that the primary data structure specifies for a ping-pong DMA cycle all the transfers that the alternate data structure specifies for the following DMA cycle types: <ul style="list-style-type: none"> — ping-pong — memory scatter-gather — peripheral scatter-gather. |

291 . UDMA Channel Primary –Alternate Set Register Description

14.6.14 CHNL_PRI_ALT_CLR

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------|-------------|---|
| 31:0 | W | chnl_pri_alt_clr | 0x0 | <p>Set the appropriate bit to select the primary data structure for the corresponding DMA channel.</p> <p>Write as:</p> <p>Bit [C] - 0 No effect. Use the chnl_pri_alt_set Register to select the alternate data structure.</p> <p>Bit [C] - 1 Selects the primary data structure for channel C.</p> <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> <p>Note</p> <p>The controller toggles the value of the chnl_pri_alt_clr [C] bit after it completes:</p> <ul style="list-style-type: none"> • the four transfers that the primary data structure specifies for a memory scatter-gather, or peripheral scatter-gather, DMA cycle • all the transfers that the primary data structure specifies for a ping-pong DMA cycle • all the transfers that the alternate data structure specifies for the following DMA cycle types: <ul style="list-style-type: none"> — ping-pong — memory scatter-gather — peripheral scatter-gather. |

292 . UDMA Channel Primary –Alternate Clear Register Description

14.6.15 CHNL_PRIORITY_SET

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:0 | R/W | chnl_priority_set | 0x0 | <p>Set the appropriate bit to select the primary data structure for the corresponding DMA channel.</p> <p>Write as:</p> <ul style="list-style-type: none"> Bit [C] - 0 No effect. Use the chnl_pri_alt_set Register to select the alternate data structure. Bit [C] - 1 Selects the primary data structure for channel C. <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> <p>Note</p> <p>The controller toggles the value of the chnl_pri_alt_clr [C] bit after it completes:</p> <ul style="list-style-type: none"> the four transfers that the primary data structure specifies for a memory scatter-gather, or peripheral scatter-gather, DMA cycle all the transfers that the primary data structure specifies for a ping-pong DMA cycle all the transfers that the alternate data structure specifies for the following DMA cycle types: <ul style="list-style-type: none"> ping-pong memory scatter-gather peripheral scatter-gather. |

293 . UDMA Channel Priority Set Register Description

14.6.16 CHNL_PRIORITY_CLR

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:0 | W | chnl_priority_clr | 0x0 | <p>Set the appropriate bit to select the default priority level for the specified DMA channel.</p> <p>Write as:</p> <ul style="list-style-type: none"> Bit [C] - 0 No effect. Use the chnl_priority_set Register to set channel C to the high priority level. Bit [C] - 1 Channel C uses the default priority level. <p>Writing to a bit where a DMA channel is not implemented has no effect.</p> |

294 . UDMA Channel Priority Clear Register Description

14.6.17 UDMA_BUS_ERR_CLR_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|-------------|
| 31:1 | R/W | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|---|
| 0 | R/W | Err_clr | 0x0 | <p>Returns the status of dma_err, or sets the signal LOW. Read as: 0 - dma_err is LOW 1 - dma_err is HIGH. Write as: 0 - No effect, status of dma_err is unchanged. 1 - Sets dma_err LOW. Note:- If you deassert dma_err at the same time as an ERROR occurs on the AHB-Lite bus, then the ERROR condition takes precedence and dma_err remains asserted.</p> |

295 . UDMA Bus Error Clear Register Description

14.6.18 UDMA_SKIP_DESC_FETCH_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|--|
| 31:0 | R/W | skip_desc_fetch | 0x0 | <p>Bit[C] - 1, enables the skipping of descriptor for each transfer for channel C</p> <p>UDMA by default fetches the source and destination addresses for each transfer, even during a burst. Setting above bit will avoid these repeated fetches within burst. We will buffer them and use within burst. This will help improving the performance of transfer and saves bus cycles. This features has to be enabled always.</p> |

296 . UDMA Skip Descriptor Register Description

14.6.19 UDMA_DONE_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 31 | R/W | Done_status_channel_31 | 0x0 | <p>Reading '1' indicates the transfer is completed for channel 31st .</p> <p>Writing '1' will clear the bit. Writing 0 will have no effect.</p> |
| 30 | R/W | Done_status_channel_30 | 0x0 | <p>Reading '1' indicates the transfer is completed for channel 30th .</p> <p>Writing '1' will clear the bit. Writing 0 will have no effect.</p> |
| 29 | R/W | Done_status_channel_29 | 0x0 | <p>Reading '1' indicates the transfer is completed for channel 29th .</p> <p>Writing '1' will clear the bit. Writing 0 will have no effect.</p> |
| 28 | R/W | Done_status_channel_28 | 0x0 | <p>Reading '1' indicates the transfer is completed for channel 28th.</p> <p>Writing '1' will clear the bit. Writing 0 will have no effect.</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 27 | R/W | Done_status_channel_27 | 0x0 | Reading '1' indicates the transfer is completed for channel 27th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 26 | R/W | Done_status_channel_26 | 0x0 | Reading '1' indicates the transfer is completed for channel 26th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 25 | R/W | Done_status_channel_25 | 0x0 | Reading '1' indicates the transfer is completed for channel 25th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 24 | R/W | Done_status_channel_24 | 0x0 | Reading '1' indicates the transfer is completed for channel 24th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 23 | R/W | Done_status_channel_23 | 0x0 | Reading '1' indicates the transfer is completed for channel 23rd . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 22 | R/W | Done_status_channel_22 | 0x0 | Reading '1' indicates the transfer is completed for channel 22nd . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 21 | R/W | Done_status_channel_21 | 0x0 | Reading '1' indicates the transfer is completed for channel 21st. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 20 | R/W | Done_status_channel_20 | 0x0 | Reading '1' indicates the transfer is completed for channel 20th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 19 | R/W | Done_status_channel_19 | 0x0 | Reading '1' indicates the transfer is completed for channel 19th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 18 | R/W | Done_status_channel_18 | 0x0 | Reading '1' indicates the transfer is completed for channel 18th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 17 | R/W | Done_status_channel_17 | 0x0 | Reading '1' indicates the transfer is completed for channel 17th . Writing '1' will clear the bit. Writing 0 will have no effect. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 16 | R/W | Done_status_channel_16 | 0x0 | Reading '1' indicates the transfer is completed for channel 16th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 15 | R/W | Done_status_channel_15 | 0x0 | Reading '1' indicates the transfer is completed for channel 15th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 14 | R/W | Done_status_channel_14 | 0x0 | Reading '1' indicates the transfer is completed for channel 14th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 13 | R/W | Done_status_channel_13 | 0x0 | Reading '1' indicates the transfer is completed for channel 13th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 12 | R/W | Done_status_channel_12 | 0x0 | Reading '1' indicates the transfer is completed for channel 12th. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 11 | R/W | Done_status_channel_11 | 0x0 | Reading '1' indicates the transfer is completed for channel 11th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 10 | R/W | Done_status_channel_10 | 0x0 | Reading '1' indicates the transfer is completed for channel 10th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 9 | R/W | Done_status_channel_9 | 0x0 | Reading '1' indicates the transfer is completed for channel 9th. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 8 | R/W | Done_status_channel_8 | 0x0 | Reading '1' indicates the transfer is completed for channel 8th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 7 | R/W | Done_status_channel_7 | 0x0 | Reading '1' indicates the transfer is completed for channel 7th . Writing '1' will clear the bit. Writing 0 will have no effect. |
| 6 | R/W | Done_status_channel_6 | 0x0 | Reading '1' indicates the transfer is completed for channel 6th . Writing '1' will clear the bit. Writing 0 will have no effect. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------------|--------------------|--|
| 5 | R/W | Done_status_channel_5 | 0x0 | Reading '1' indicates the transfer is completed for channel 5th. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 4 | R/W | Done_status_channel_4 | 0x0 | Reading '1' indicates the transfer is completed for channel 4th. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 3 | R/W | Done_status_channel_3 | 0x0 | Reading '1' indicates the transfer is completed for channel 3rd. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 2 | R/W | Done_status_channel_2 | 0x0 | Reading '1' indicates the transfer is completed for channel 2nd. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 1 | R/W | Done_status_channel_1 | 0x0 | Reading '1' indicates the transfer is completed for channel 1st. Writing '1' will clear the bit. Writing 0 will have no effect. |
| 0 | R/W | Done_status_channel_0 | 0x0 | Reading '1' indicates the transfer is completed for channel 0th. Writing '1' will clear the bit. Writing 0 will have no effect. |

297 . UDMA Done Status Register Description

14.6.20 UDMA_CHANNEL_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|---------------------------------|--------------------|--|
| 31 | R | Busy or ideal status_channel_31 | 0x0 | Reading '1' indicates that the channel 31 is busy. |
| 30 | R | Busy or ideal status_channel_30 | 0x0 | Reading '1' indicates that the channel 30 is busy. |
| 29 | R | Busy or ideal status_channel_29 | 0x0 | Reading '1' indicates that the channel 29 is busy. |
| 28 | R | Busy or ideal status_channel_28 | 0x0 | Reading '1' indicates that the channel 28 is busy. |
| 27 | R | Busy or ideal status_channel_27 | 0x0 | Reading '1' indicates that the channel 27 is busy. |
| 26 | R | Busy or ideal status_channel_26 | 0x0 | Reading '1' indicates that the channel 26 is busy. |
| 25 | R | Busy or ideal status_channel_25 | 0x0 | Reading '1' indicates that the channel 25 is busy. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|---------------------------------|--------------------|--|
| 24 | R | Busy or ideal status_channel_24 | 0x0 | Reading '1' indicates that the channel 24 is busy. |
| 23 | R | Busy or ideal status_channel_23 | 0x0 | Reading '1' indicates that the channel 23 is busy. |
| 22 | R | Busy or ideal status_channel_22 | 0x0 | Reading '1' indicates that the channel 22 is busy. |
| 21 | R | Busy or ideal status_channel_21 | 0x0 | Reading '1' indicates that the channel 21 is busy. |
| 20 | R | Busy or ideal status_channel_20 | 0x0 | Reading '1' indicates that the channel 20 is busy. |
| 19 | R | Busy or ideal status_channel_19 | 0x0 | Reading '1' indicates that the channel 19 is busy. |
| 18 | R | Busy or ideal status_channel_18 | 0x0 | Reading '1' indicates that the channel 18 is busy. |
| 17 | R | Busy or ideal status_channel_17 | 0x0 | Reading '1' indicates that the channel 17 is busy. |
| 16 | R | Busy or ideal status_channel_16 | 0x0 | Reading '1' indicates that the channel 16 is busy. |
| 15 | R | Busy or ideal status_channel_15 | 0x0 | Reading '1' indicates that the channel 15 is busy. |
| 14 | R | Busy or ideal status_channel_14 | 0x0 | Reading '1' indicates that the channel 14 is busy. |
| 13 | R | Busy or ideal status_channel_13 | 0x0 | Reading '1' indicates that the channel 13 is busy. |
| 12 | R | Busy or ideal status_channel_12 | 0x0 | Reading '1' indicates that the channel 12 is busy. |
| 11 | R | Busy or ideal status_channel_11 | 0x0 | Reading '1' indicates that the channel 11 is busy. |
| 10 | R | Busy or ideal status_channel_10 | 0x0 | Reading '1' indicates that the channel 10 is busy. |
| 9 | R | Busy or ideal status_channel_9 | 0x0 | Reading '1' indicates that the channel 9 is busy. |
| 8 | R | Busy or ideal status_channel_8 | 0x0 | Reading '1' indicates that the channel 8 is busy. |
| 7 | R | Busy or ideal status_channel_7 | 0x0 | Reading '1' indicates that the channel 7 is busy. |
| 6 | R | Busy or ideal status_channel_6 | 0x0 | Reading '1' indicates that the channel 6 is busy. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------------------|-------------|---|
| 5 | R | Busy or ideal status_channel_5 | 0x0 | Reading '1' indicates that the channel 5 is busy. |
| 4 | R | Busy or ideal status_channel_4 | 0x0 | Reading '1' indicates that the channel 4 is busy. |
| 3 | R | Busy or ideal status_channel_3 | 0x0 | Reading '1' indicates that the channel 3 is busy. |
| 2 | R | Busy or ideal status_channel_2 | 0x0 | Reading '1' indicates that the channel 2 is busy. |
| 1 | R | Busy or ideal status_channel_1 | 0x0 | Reading '1' indicates that the channel 1 is busy. |
| 0 | R | Busy or ideal status_channel_0 | 0x0 | Reading '1' indicates that the channel 0 is busy. |

298 . UDMA Channel Status Register Description

14.6.21 UDMA_CONFIG_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------|-------------|---|
| 31:1 | R/W | Reserved | 0x0 | Reserved |
| 0 | R/W | Single_request_enable | 0x0 | <p>Enabled signal for single request 0 - Single request will be disabled 1 - Single request will be enabled</p> <p>We are connecting dma_waitonreq port to zero. DMA is considering single requests(empty, full signals from peripherals) only when dma_waitonreq is high. When single_request_enable is set, we have added logic to ignore dma_waitonreq and consider single requests.</p> <p>If you don't enable this, following case will not work We want to read 10 bytes from UART and beat size/fifo threshold being configured as 4 bytes. Only 8 bytes will be read and DMA will not read remaining two bytes as dma_req will not come</p> |

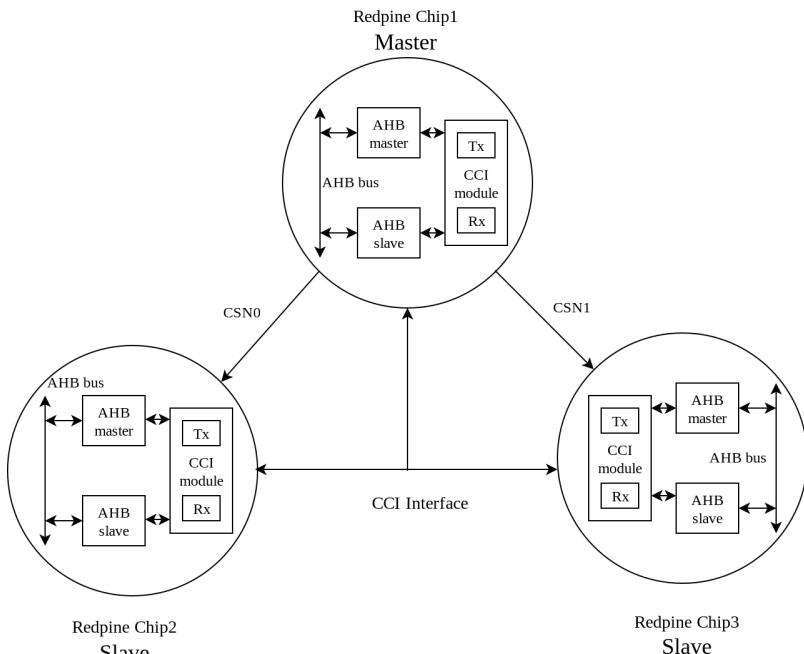
299 . UDMA Configuration Control Register Description

15 MCU AHB High Speed Peripherals

15.1 CCI Controller

15.1.1 General Description

Companion Chip Interface (CCI) controller is used to connect Multiple Redpine chipsets together for accessing memories and peripherals. This also enables connecting our smallest size chip with MCU to other chips where MCU may not be present. They will be connected through this interface. This will enable create faster Multi-chip Modules quickly.



CCI system level diagram

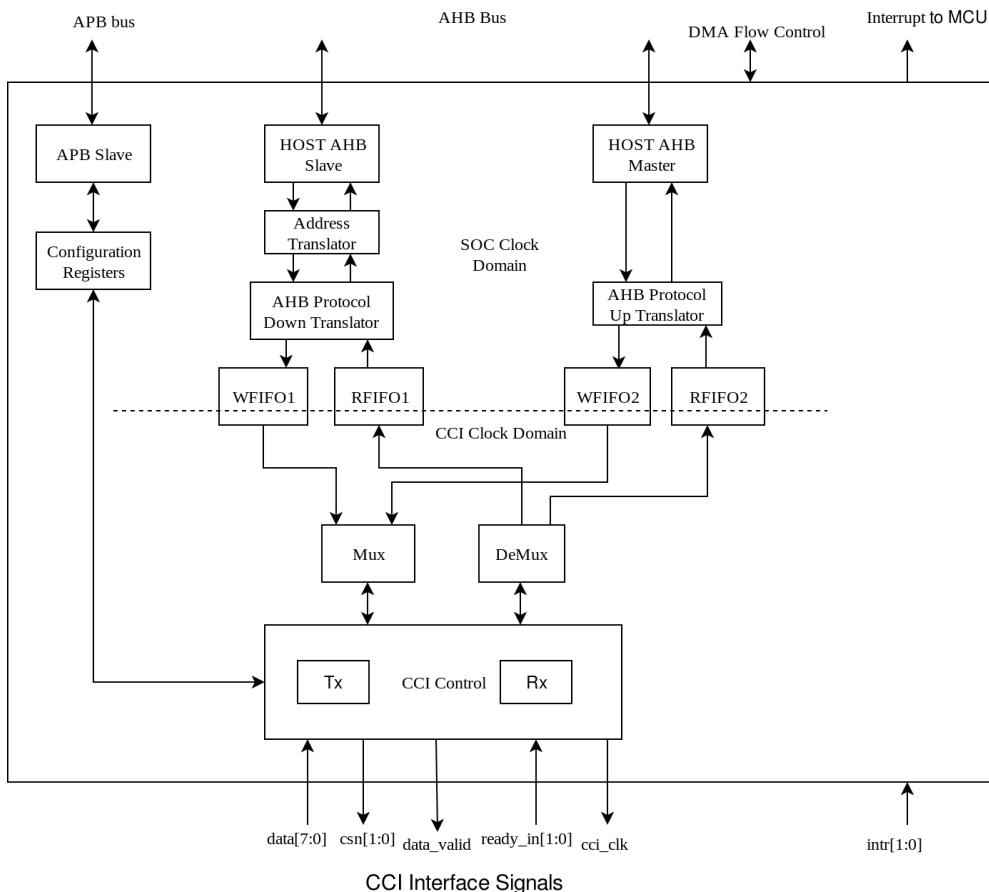
The CCI controller communicates with the other chips through the CCI interface. On the other side, it communicates within the chip through the AHB and APB interface. APB is used for configuration and AHB is used for the data transfer. Through the CCI a maximum of 3 Redpine chips can be connected to each other as shown in Figure 1. One of the three devices is configured as CCI Master, which drives the chip selects csn0, and csn1 for the two CCI slaves. CCI Master can communicate with any of the CCI Slaves and any CCI Slave can communicate with the CCI Master only. In this way the functionality that is present in all the chips can be achieved in the super chip (by connecting the dies together).

15.1.2 Features

- Used as Chip to chip AHB extension to accessing memories and peripherals
- Supports 4 bit and 8-bit bidirectional data bus at CCI
- A maximum of 3 devices can be connected using CCI
- Supports messaging protocol
- Supports both SDR and DDR modes. SDR is the default mode.
- Supports Address translation in CCI Module
- Peer to Peer communication is possible without processor intervention
- A Redpine device with CCI will be able to connect other Redpine chipsets, new peripherals, analog peripherals (analog companion chip – adc,dac, PLLs, PMU etc.)
- Supports flow control during data transfer between two chips
- Minimizes latency between AHB side to external I/F and vice versa by using DDR Mode at CCI

- A device can access any AHB address of other chip when the master is connected to one slave. If the master is connected to multiple(max 2) slaves, the address space has to be shared
- When the master is idle/after certain time, it gives chance to any of the slaves for transferring data in round robin fashion.
- A device with CCI acts as a master or slave based on chip initialization
- Provides wait cycles at CCI when there is no data in AHB FIFO to transfer
- Uses 2x clock to read the data from CCI DDR interface and write into FIFO
- APB is used for internal register access and AHB for data transfer
- Supports AMBA AHB 2.0 protocol and has both AHB master and slave
- Supports DMA flow control
- Programmable FIFO thresholds
- CCI clock is synchronous to AHB(SoC) clock in Master mode
- Supports transaction request from Slave when there is data to transfer from Slave to Master. Supports priority when more than one slave is transaction request is asserted at a time.
- Supports interrupt from CCI Slave
- Supports prefetch mode during read operation

15.1.3 Functional Description



CCI block diagram

CCI controller consists of the following sub modules – APB Slave, AHB master, AHB Slave, Address translator, AHB protocol down translator, AHB protocol up translator, CCI TX and RX Control, Read and Write FIFOs. Protocol Down Translator, translates the AHB Transactions into 40 bit format to be written into FIFO. During the address phase of the AHB, it will append the control information along with the address and then post into the FIFO. During data phase of AHB transactions, data is posted in WFIFO for write operation. Protocol Up Translator reads the data from

the FIFO and generates the AHB address, control signals and data, which are going to be posted on AHB bus. Similarly, it generates the AHB Master read transactions when the request is posted by the slave. As Protocol down translator does not have any bus size related information it will control only ready generation. So transactions posted in the RFIFO should take care of unnecessary read transactions.

CCI controller module controls the CCI Tx controller, CCI Rx controller, master and slave bus access. Tx controller comprises of transaction decoder and mux, to convert the 40 bit transaction into 8 bit transaction. Rx controller mainly comprises of command decoder and transaction monitor and 8 to 40 bit FIFO converter. It takes care of scheduling the interrupt to MCU, based on the current transfer and the interrupts asserted. It takes care of the priority of the interrupts when multiple interrupts are asserted.

Programming Details

In the CCI write, WFIFO1 of Tx device and RFIFO2 of Rx device will be active. The address, data and control information from the AHB slave will be posted into the WFIFO1 through the protocol down translator and transmitted using CCI. The data received by the Rx device will be written into RFIFO2 and transferred to AHB master through the protocol up translator.

In the CCI read, WFIFO1 of Tx device and RFIFO2 of Rx device will be active during the address phase. The address and control information from the AHB slave will be posted into the WFIFO1 through the protocol down translator and transmitted using CCI. This information received by the Rx device will be written into RFIFO2 and transferred to AHB master through the protocol up translator. In the data phase, the response data from the AHB master in Rx device is written into WFIFO2 using protocol up translator. This information is transferred through the CCI to the Tx device. This information is written into RFIFO1 and passed on to AHB slave using the protocol down translator. When the master wants to initiate a transaction, it drives the corresponding CSN low. Slave requests the master by using the request/interrupt signal. It supports SDR and DDR Modes.

Following is the programming sequence for CCI controller:

- Configure chip as master or slave using MCU misc config registers (MISC_CFG_MISC_CTRL_M4SS). Enable the CCI controller.
- Choose the SDR or DDR mode, Address translation enable, address width config and number of slaves by programming the CCI_CONTROL register.
- Allocate the lower and higher address range for each slave by programming in the following registers

| Register name | Slave number |
|---------------|----------------------|
| CCI_LSB_A_S1 | slave1 lower address |
| CCI_LSB_A_S2 | slave2 lower address |
| CCI_MSB_A_S1 | slave1 Upper address |
| CCI_MSB_A_S2 | slave2 Upper address |

- Load the translation address if address translation feature is enabled by programming the cci_translaton_address
- Configure the interrupt_mode in CCI_MODE_INTR_STATUS register .
- config the CCI FIFO threshold by using CCI_FIFO_THRESHOLD register
- If CCI prefetch is required, enable CCI_prefetch mode in CCI_PREFETCH_CTRL register
- Set cci_2x_clk_enable bit in CCI_PREFETCH_CTRL register for DDR mode.
- Configure time out value for slave response in CCI_CONTROL register
- If interrupt mode is enabled, it rises interrupt after receiving interrupt from Slave. After serving this interrupt, set ack bit in CCI_MODE_INTR_STATUS register.
- After configuring CCI with above steps, use the CCI interface for transferring data from one chip to another chip.
- Above configuration to be done for both chips (master and Slave)

Prefetch mode

There is an option for prefetch the data corresponding to the next read address during read operation. Prefetch is enabled by using 0th bit (prefetch_en) in CCI_PREFETCH_CTRL. This is valid only for AHB read operations. In this mode, after completion any read operation, another read operation with consecutive address is initiated by CCI controller. Once read data is available from CCI interface, it is stored in internal FIFO, which is readily available for next consecutive AHB read address. If next AHB read address matches with this prefetched address, then it provides readily available prefetched data without any delay. It will not initiate read operation on CCI interface. If next AHB read address is not matches with this prefetched address, then it drops the prefetched data from FIFO and initiate the read operation on CCI interface.

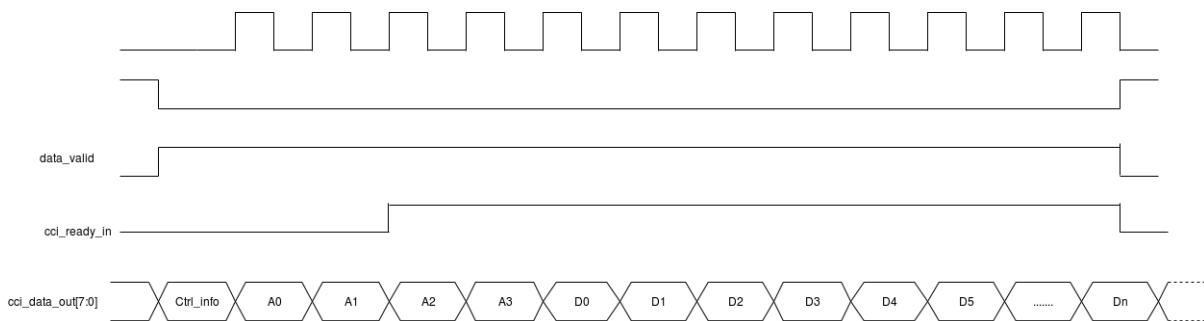
Supported Transfers

Following transactions are supported between CCI controllers:

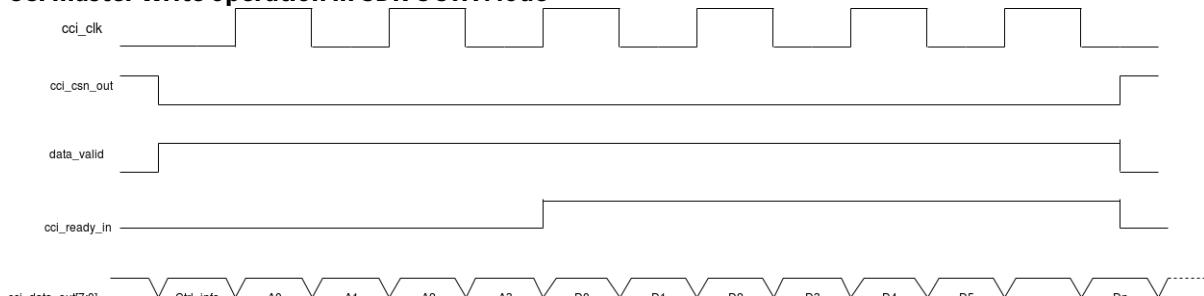
CCI Master Write Operation

The following transactions take place when the bus is idle during CCI master requesting for a write into one of the CCI Slave's peripherals/memory:

- CSN will be made low by master and the control, address and data bytes are sent in the command.
- In case of any delays from Slave, ready will be made low in between.
- When master wants to introduce delay in the transaction, it will gate the clock.
- After completion of this write operation, CSN will be made high by CCI Master.
- If this address is not valid on slave side, slave drops this packet.
- CCI master transfers this data either in SDR or DDR mode according to the initial programming as shown in Figure 3 and 4.



CCI master Write operation in SDR OCTA Mode

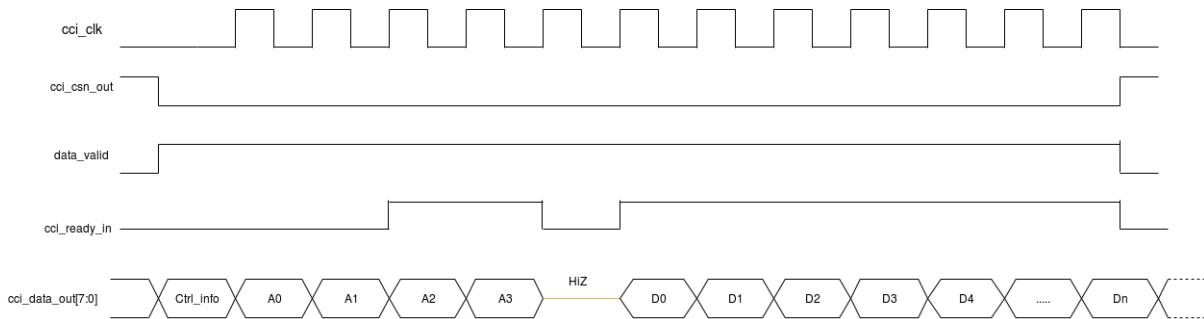


CCI master Write operation in DDR Mode

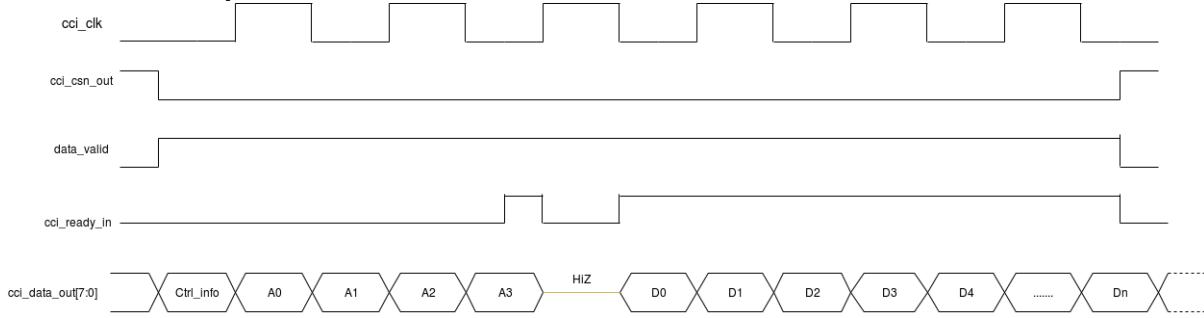
CCI Master Read Operation

The following transactions take place when the bus is idle during CCI master requesting for a read from one of the CCI Slave's peripherals/memory:

- CSN will be made low by master and sends the control and address information.
- After that bus will be in high impedance mode for a clock cycle.
- Then Slave starts driving the data when address is valid.
- In case of any delays from slave, ready will be made low in between.
- When master wants to introduces any delay during the transaction, it will gate the clock.
- After completion of this write operation, CSN will be made high by CCI Master.
- CCI master read this data either in SDR or DDR mode according to the initial programming as shown in Figure 5 and 6.



CCI master Read Operation in SDR Mode



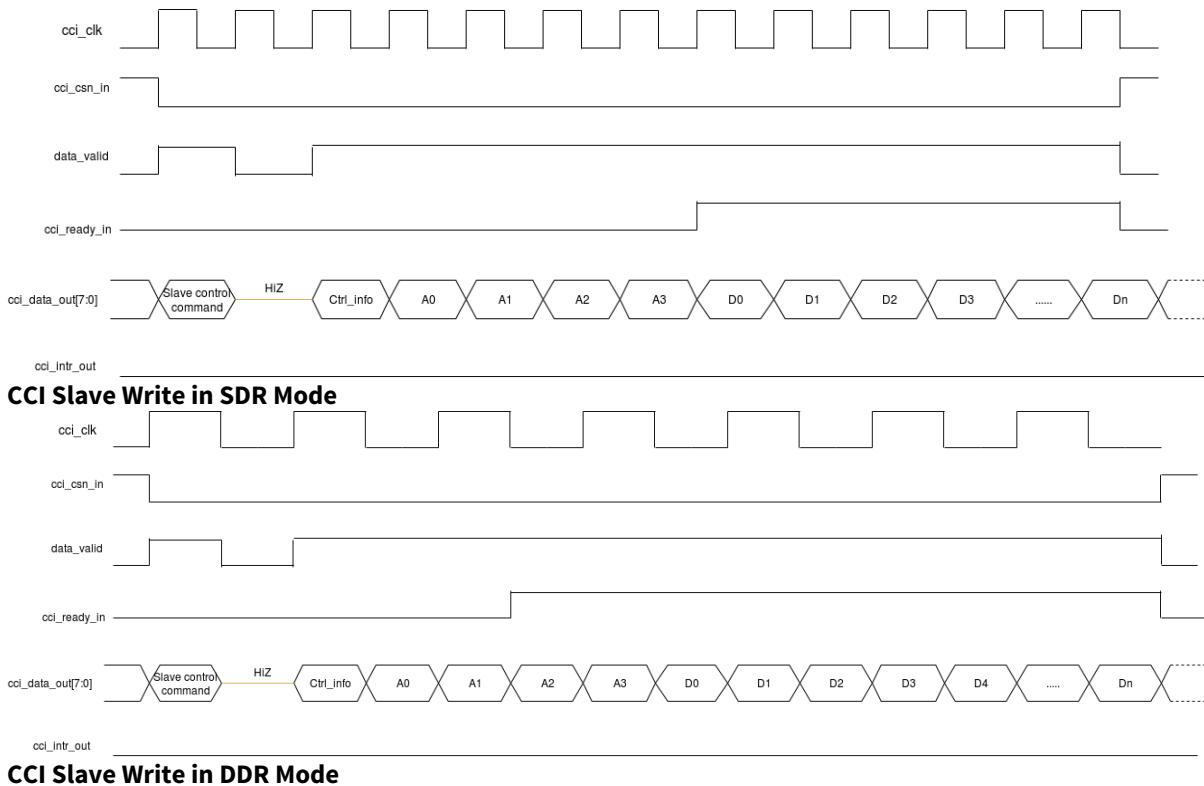
CCI master Read Operation in DDR Mode

CCI Slave Write Operation

The following transactions take place when the bus is idle during CCI Slave requesting for a write into CCI master's peripherals/memory:

- Slave asserts ready for requesting the transaction.
- CSN will be made low by Master and the slave control command is sent in the command and the data bus goes into high impedance state.
- When slave starts driving it, it will make ready high(indicating data_vld) and sends the data.
- In case of any delays from slave, ready will be made low in between.
- When master wants to introduce delay in the transaction, it will gate the clock.
- If master wants to terminate the transaction, it will de-assert CSN as soon as the complete data is transferred for the current transaction.

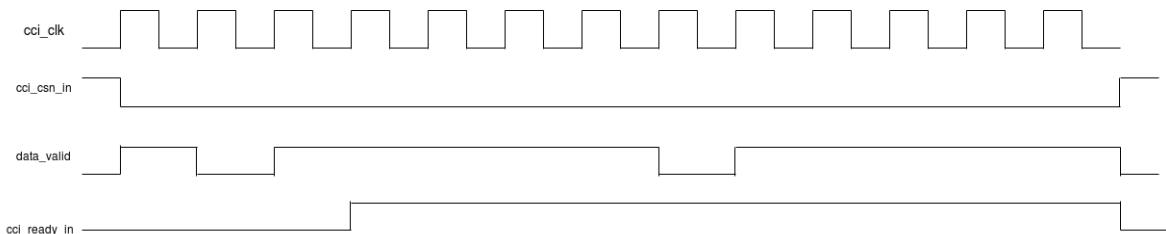
- CCI slave transfers this data either in SDR or DDR mode according to the initial programming as shown in Figure 7 and 8.

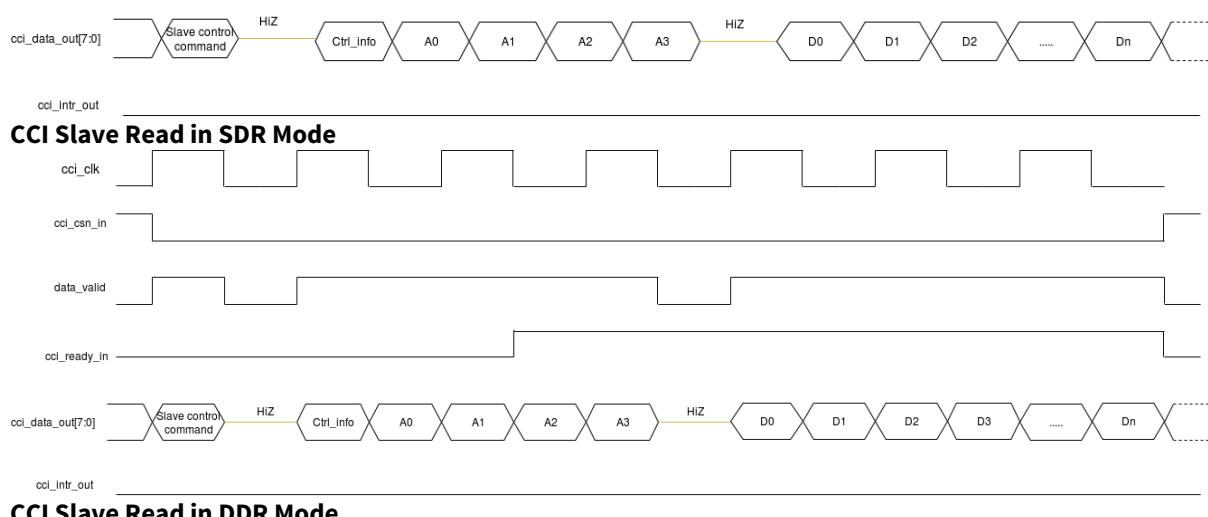


CCI Slave Read Operation

The following transactions take place when the bus is idle during CCI Slave requesting for a read from CCI master's peripherals/memory

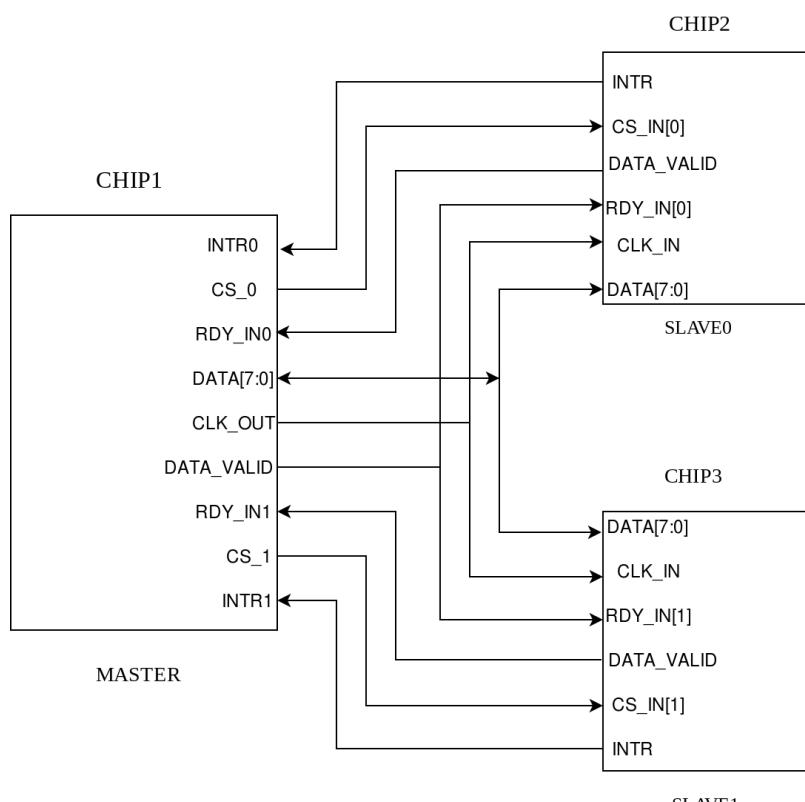
- Slave asserts ready for requesting the transaction.
- CSN will be made low by master and the slave control command is sent in the command and the data bus goes into high impedance state.
- When slave starts driving it, it will make ready high(indicating `data_vld`) and sends the control and address information. After that bus will be in high impedance state for a clock cycle.
- Then master starts driving the data when address is valid.
- In case of any delays from slave, ready will be made low in between.
- When master wants to introduce any delays during the transaction, it will gate the clock.
- If master wants to terminate this transaction, it will make CSN high.
- CCI slave reads this data either in SDR or DDR mode according to the initial programming as shown in Figure 9 and 10.





CCI connections between chips

Following Figure 11 shows the CCI interface connection between CCI master chip and CCI slave chips.



Connections between Chips using CCI interface

15.1.4 Register Summary

APB Register's Base Address: 0x4617_0000

CCI controller AHB slave base address for accessing any data from another chip is **0x6000_0000**

| Register Name | Offset | Description |
|----------------------|--------|--|
| CCI_CONTROL | 0x000 | CCI control register |
| CCI_LSB_A_S1 | 0x004 | CCI slave1 lower part address register |
| CCI_LSB_A_S2 | 0x008 | CCI slave2 lower part address register |
| Reserved | 0x00C | Reserved |
| CCI_MSB_A_S1 | 0x010 | CCI slave1 upper part address register |
| CCI_MSB_A_S2 | 0x014 | CCI slave2 upper part address register |
| Reserved | 0x018 | Reserved |
| Reserved | 0x01C | Reserved |
| Reserved | 0x020 | Reserved |
| Reserved | 0x024 | Reserved |
| CCI_MODE_INTR_STATUS | 0x028 | CCI interrupt status register |
| Reserved | 0x034 | Reserved |
| Reserved | 0x038 | Reserved |
| Reserved | 0x03C | Reserved |
| Reserved | 0x040 | Reserved |
| CCI_FIFO_THRESHOLD | 0x200 | CCI FIFO threshold register |
| CCI_TRANS_ADDRESS | 0x204 | CCI address translation register |
| CCI_PREFETCH_CTRL | 0x208 | CCI pre-fetch control register |

300 . Register Summary Table

15.1.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, - = Reserved

CCI_CONTROL

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------------------------------|-----------|--|
| 31 | R/W | cci_ctrl_enable | 0 | enable for CCI controller. set this bit after gpio pads initialization |
| 30 | R/W | select_time_out_intr_or_msg_intr | 0 | selection between time out and message interrupts |
| 29 | R/W | disable_time_out_for_data_access | 0 | disables the time out for data access |
| [28:26] | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|---------|--------|-------------------|-----------|--|
| [25:16] | R/W | time_out_prg | 1 | configurable time out value for slave response |
| [15:13] | R/W | slave_priority | 0 | This bits will represents priority of the slaves : 001 : slave 0 has highest priority 010 : slave – 1 has highest priority others : Reserved |
| 12 | R/W | Reserved | 0 | Reserved |
| [11:9] | R/W | mode | 0 | [11] : This bit represents mode of the interface : 1'b1 : ddr mode 1'b0 : sdr mode [10:9] : This two bits represents width of the interface. 00 : quad mode 01 : octa mode 10 : word mode |
| 8 | R/W | translate_enable | 0 | translation enable |
| [7:6] | R/W | addr_width_config | 2'b11 | address width configuration of AHB slave during address phase 11 -> 40 bit width (32 bit address and 8 bit command) 10 -> 32 bit width (24 bit address and 8 bit command) 01 -> 24 bit width (16 bit address and 8 bit command) 00 -> 16 bit width (8 bit address and 8 bit command) |
| 5 | R/W | ebt_s | 0 | Support for Early Burst Termination |
| [4:2] | R/W | enabled_slaves | 0 | Indicates Slaves enable. 000,001: slave0 is enabled. 010: slave1 is enabled. 011: slave0 and slave 1 are enabled 100: reserved 101: reserved 110: reserved 111: reserved. According to these slave enables, corresponding slave csn is selected during data transfer. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------|-----------|--------------------------------|
| [1:0] | R/W | num_slaves | 0 | Indicates the number of slaves |

301 .CCI_CONTROL Description

CCI_LSB_A_S1

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------|-----------|---|
| [31:0] | R/W | Lower address | 0 | Lower Address of slave 0 supported. Make sure that slave0 is enabled. |

302 .CCI LSB A S1 Description

CCI_LSB_A_S2

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------|-----------|---|
| [31:0] | R/W | Lower address | 0 | Lower Address of slave 1 supported. Make sure that slave1 is enabled. |

303 .CCI LSB A S2 Description

CCI_MSB_A_S1

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------------|-----------|--|
| [31:0] | R/W | Higher Address | 0 | Higher Address of slave 0 supported. Make sure that slave0 is enabled. |

304 .CCI MSB A S1 Description

CCI_MSB_A_S2

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------------|-----------|--|
| [31:0] | R/W | Higher Address | 0 | Higher Address of slave 1 supported. Make sure that slave1 is enabled. |

305 .CCI MSB A S2 Description

CCI_MODE_INTR_STATUS

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------|-----------|-------------|
| [15:13] | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|---------|--------|-------------|-----------|--|
| [12:11] | R | intr_status | 0 | <p>These bits will represents the status of the interrupt in read mode.</p> <p>Whenever CCI slave asserts interrupt on pin, CCI master rises interrupt to SOC processor. There are two interrupt status bits in CCI master interrupt register. They represent two CCI slaves (slave1 and slave0) respectively.</p> |
| [10:5] | R/W | Reserved | 0 | Reserved |
| [4:3] | R/W | intr_clear | 0 | <p>By setting this bits will clear the interrupt status [1:0] interrupts from the peer chips</p> |
| [2:0] | R/W | Reserved | 0 | Reserved |

306 .CCI_MODE_INTR_STATUS Description

CCI_FIFO_THRESHOLD

| Bit | Access | Function | POR Value | Description |
|---------|--------|------------------------|-----------|------------------------|
| [31:10] | R/W | Reserved | 0 | Reserved |
| [9:5] | R/W | FIFO_AEMPTY_Threshold0 | | Almost Empty Threshold |
| [4:0] | R/W | FIFO_AFull_Threshold | 0XA | Almost Full Threshold |

307 .CCI_FIFO_THRESHOLD Description

CCI_TRANS_ADDRESS

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------------------|-----------|---|
| [31:1] | R/W | Translation Address | 0 | <p>Address offset for translation address. Address translation should be enabled and translation address value is a000_0000 for accessing other chip address range 0000_0000 to 0x5FFF_FBFF. If we want access beyond this range, we have to change translation address value according to new address range of other chip.</p> |
| [0] | R/W | Translation Address Valid | 0 | <p>Translation is enabled or not</p> <p>1 : Translation is enabled</p> <p>0 : Translation is not enabled</p> |

308 .CCI_TRANS_ADDRESS Description

CCI_PREFETCH_CTRL

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------------------------|-----------|---|
| [31:2] | R/W | Reserved | 0x00 | Reserved |
| [1] | R/W | cci_2x_clk_enable_for_ddr_mode | 0x0 | It is an enable for CCI 2x clock in DDR mode. |
| [0] | R/W | cci_prefetch_en | 0x0 | cci pre-fetch enable: 1: prefetch is enable on AHB read operation. In this mode, next AHB read transaction is perfected and kept in async FIFO, which is readily available for next consecutive AHB read address. 0 : prefetch operation is disabled |

309 .CCI_PREFETCH_CTRL Description

15.2 Ethernet Controller

15.2.1 General Description

The Ethernet Controller enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2002 standard. It supports Reduced Media Independent Interface (RMII) interface to connect external PHY. The RMII specification reduces the pin count between Ethernet PHY and Controller.

It is compliant to the following standards:

- IEEE 802.3-2002 for Ethernet MAC
- AMBA 2.0 for AHB Master/Slave ports
- RMII specification from RMII consortium
- The Ethernet transfers data to system memory through the AHB master interface. It is present in parallel to SDIO/SPI/USB host interfaces.

It has internal DMA exchanges data between the MAC Transaction Layer (MTL) and host memory. A set of registers (DMA CSR) to control DMA operation is accessible by the host.

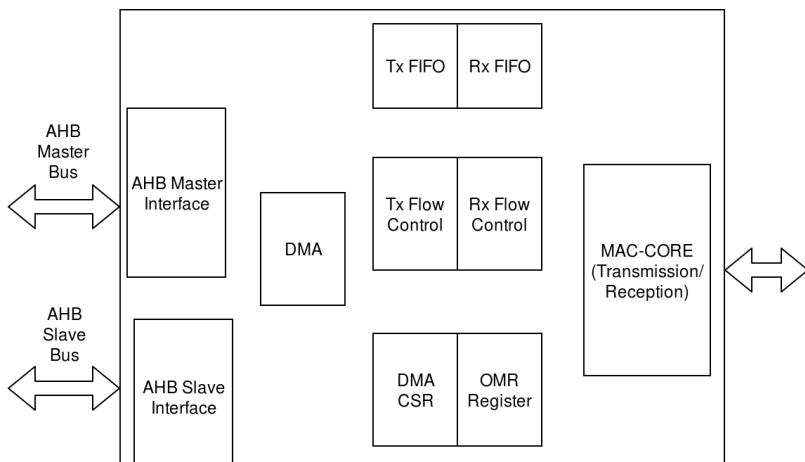
15.2.2 Features

- Supports 10/100-Mbps data transfer rates with the IEEE 802.3-compliant RMII interface to communicate with an external Fast Ethernet PHY.
- Supports both full-duplex and half-duplex operation
 - Supports CSMA/CD Protocol for half-duplex operation
 - Supports packet bursting and frame extension in 1000 Mbps half-duplex operation
 - Supports IEEE 802.3x flow control for full-duplex operation
 - forwarding of received pause control frames to the user application in full-duplex operation
 - Back-pressure support for half-duplex operation
 - Automatic transmission of zero-quanta pause frame on de-assertion of flow control input in full duplex operation
- Preamble and SFD insertion in Transmit, and deletion in Receive paths
- Automatic CRC and pad generation controllable on a per-frame basis
- Options for Automatic Pad/CRC Stripping on receive frames.
- Programmable frame length to support Standard or Jumbo Ethernet frames with sizes up to 16 KB

- Programmable Inter Frame Gap (40-96 bit times in steps of 8)
- Supports a variety of flexible address filtering modes:
 - Up to 31 additional 48-bit perfect (DA) address filters with masks for each byte
 - Up to 31 48-bit SA address comparison check with masks for each byte
 - 64-bit Hash filter for multicast and uni-cast (DA) addresses
 - Option to pass all multicast addressed frames
 - Promiscuous mode support to pass all frames without any filtering for network monitoring
 - Passes all incoming packets (as per filter) with a status report
- Separate 32-bit status returned for transmission and reception packets
- Supports IEEE 802.1Q VLAN tag detection for reception frames
- Separate transmission, reception, and control interfaces to the Application
- Little endian support for transmission and reception data paths
- Supports 32-bit data transfer interface on the system-side
- MDIO Master interface for PHY device configuration and management
- Optional module for detection of LAN wake-up frames and AMD Magic Packet frames
- Optional Receive module for checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame
- Optional Enhanced Receive module for checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams.
- Supports Internal DMA
 - 32-bit data transfers
 - Single-channel Transmit and Receive engines
 - Optimization for packet-oriented DMA transfers with frame delimiters
 - Byte-aligned addressing for data buffer support
 - Dual-buffer (ring) or linked-list (chained) descriptor chaining
 - Descriptor architecture, allowing large blocks of data transfer with minimum CPU intervention; each descriptor can transfer up to 8 KB of data
 - Comprehensive status reporting for normal operation and transfers with errors
 - Individual programmable burst size for Transmit and Receive DMA Engines for optimal host bus utilization
 - Programmable interrupt options for different operational conditions
 - Per-frame Transmit/Receive complete interrupt control
 - Round-robin or fixed-priority arbitration between Receive and Transmit engines
 - Start/Stop modes

15.2.3 Functional Description

A block diagram of the Ethernet Controller is shown in Figure 1. Major blocks of Ethernet Controller are DMA, AHB interface, MTL, MAC transmitter and receiver and PHY interface.



Ethernet Block Diagram

In this, the DMA Controller interfaces with the SOC through the AMBA AHB Interface. The AHB Master Interface controls data transfers while the AHB Slave interface accesses CSR space. The DMA can be used in 32-bit embedded applications where DMA is required to optimize data transfer between the MTL and system memory. The AHB Master interface converts the internal DMA request cycles into AHB cycles.

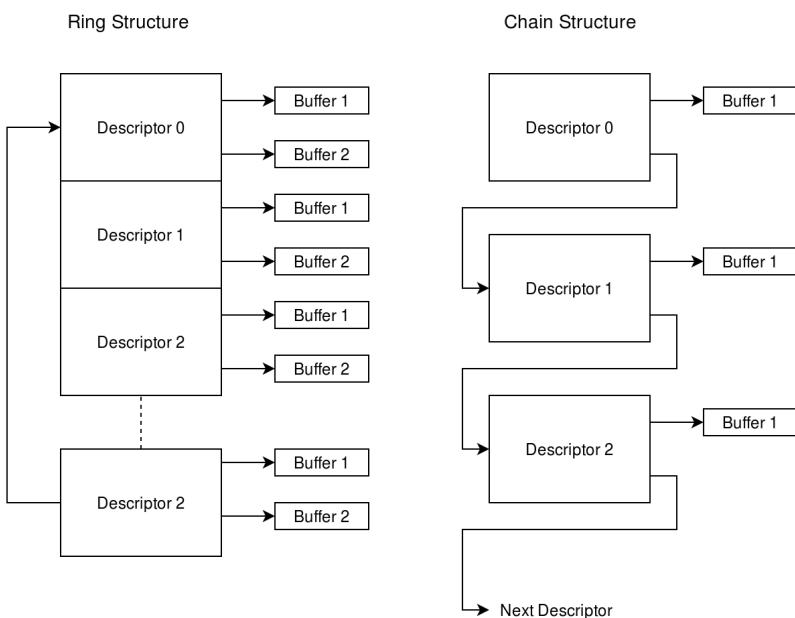
DMA

The DMA has independent Transmit and Receive engines, and a CSR space. The Transmit Engine transfers data from system memory to the device port (MTL), while the Receive Engine transfers data from the device port to system memory. The controller utilizes descriptors to efficiently move data from source to destination with minimal MCU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the MCU for situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The DMA and the Host driver communicate through two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers data frames received by the core to the Receive Buffer in the Host memory, and Transmit data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers. There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into DMA Receive Descriptor List Address Register and Transmit Descriptor List Address Register, respectively. A descriptor list is forward linked. The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both Receive and Transmit descriptors (RDES1[24] and TDES1[24]). The descriptor lists resides in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory. A data buffer resides in the Host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA will skip to the next frame buffer when end-of-frame is detected. Data chaining can be enabled or disabled. The descriptor ring and chain structure is shown in Figure 2.



DMA Descriptor Ring and Chain Structure

Initialization

Initialization for the MAC-Core is as follows.

1. Write to DMA Bus Mode Register to set Host bus access parameters.
2. Write to DMA Interrupt Enable Register to mask unnecessary interrupt causes.
3. The software driver creates the Transmit and Receive descriptor lists. Then it writes to both DMA Receive Descriptor List Address Register and DMA Transmit Descriptor List Address Register, providing the DMA with the starting address of each list.
4. Write to MAC-Core MAC Frame Register, Hash Table High Register and Hash Table Low Register for desired filtering options.
5. Write to MAC-Core MAC Configuration Register to configure and enable the Transmit and Receive operating modes. The PS and DM bits are set based on the auto-negotiation result (read from the PHY).
6. Write to DMA Operation Mode Register to set bits 13 and 1 to start transmission and reception.
7. The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

DMA Arbiter

The arbiter inside the DMA module performs the arbitration between the Transmit and Receive channel accesses to the AHB Master interface. Two types of arbitration are possible: round-robin, and fixed-priority. When round-robin arbitration is selected (DA bit of DMA Bus Mode Register

is reset), the arbiter allocates the data bus in the ratio set by the PR bits of DMA Bus Mode Register, when both Transmit and Receive DMAs are requesting for access simultaneously. When the DA bit is set, the Receive DMA always gets priority over the Transmit DMA for data access.

Transmission

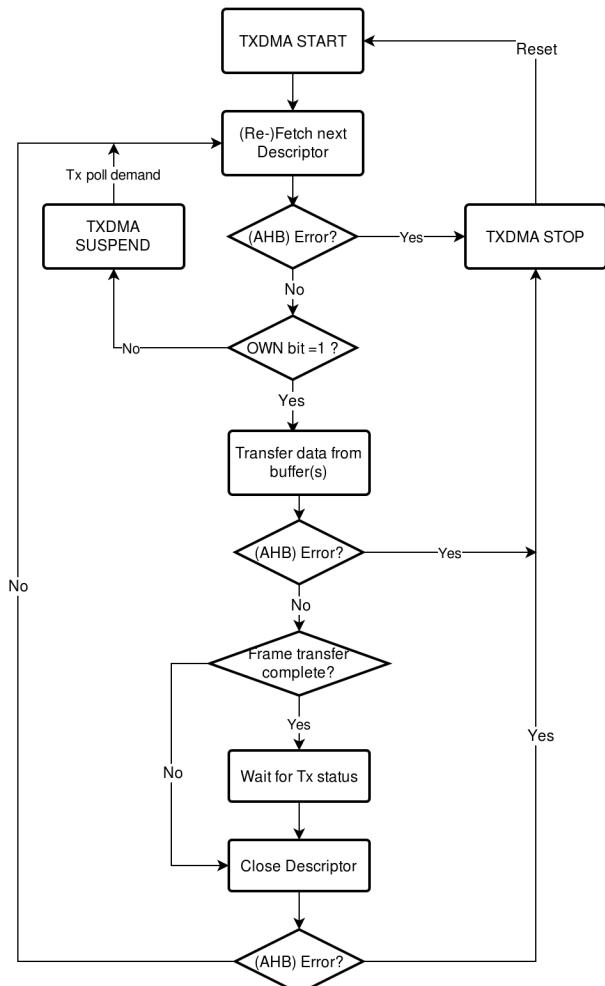
Normal mode

The Transmit DMA engine in default mode proceeds in the following sequence:

1. The Host sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Frame data.
2. Once the ST bit (DMA Operation Mode Register[13]) is set, the DMA enters the Run state.
3. While in the Run state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the Host, or if an error condition occurs, transmission is suspended and both the Transmit Buffer Unavailable (DMA Status Register[2]) and Normal Interrupt Summary (DMA Status Register[16]) bits are set. The Transmit Engine proceeds to Step 9.
4. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1'b1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the Transmit data from the Host memory and transfers the data to the MTL for transmission.
6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3, 4, and 5 are repeated until the end-of-Ethernet frame data is transferred to the MTL.
7. When frame transmission is complete, status information is written into Transmit Descriptor 0 (TDES0) which has the end-of frame buffer.
8. Transmit Interrupt (DMA Status Register[0]) is set after completing transmission of a frame that has Interrupt on Completion (TDES1[31]) set in its Last Descriptor. The DMA engine then returns to Step 3.

9.In Suspend state, the DMA tries to re-acquire the descriptor (jump to Step 3) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared.

The default mode transmission flow is charted in Figure 3.



Normal TxDMA Operation

OSF Mode

While in the Run state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first (if the OSF bit is set in DMA Operation Mode Register[2]). As the transmit process finishes transferring the first frame, it immediately polls the Transmit Descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the status information of the first frame.

In OSF mode, the sequence of Transmit DMA operation in the Run state is as follows:

- 1.The DMA operates as described in steps 1–6 of the Transmission process.
- 2.Without closing the previous frame's descriptor, the DMA fetches the next descriptor.
- 3.If the acquired descriptor is owned by DMA, the DMA decodes the Transmit Buffer address in this descriptor. If the descriptor is not owned by DMA, the sequence DMA goes into SUSPEND mode and the sequence skips to Step 9.
- 4.The DMA fetches the Transmit frame from the Host memory and transfers the frame to the MTL until the End-of-Frame data is transferred.
- 5.The DMA waits for the previous frame's frame transmission status and writes the status to its corresponding TDES0 when it receives it.

6.If enabled, the Transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to Step 3 (when Status is normal). If the transmission status shows errors such as Underflow, the DMA goes into Suspend mode (Step 9).

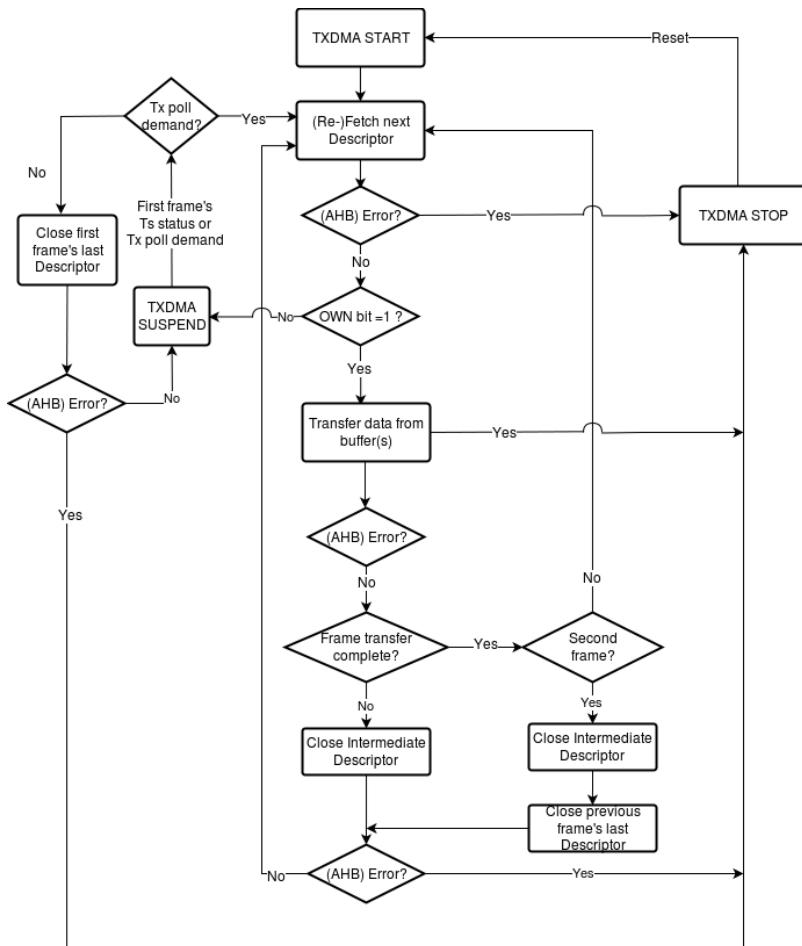
7.The DMA waits for the current frame's frame transmission status and, when it received it, writes the status to the corresponding TDES0.

8.If enabled, the Transmit interrupt is set and the DMA goes into Suspend mode.

9.In Suspend mode, if any pending status is received from MTL, that status is written to the corresponding TDES0, relevant interrupts are set, and the DMA returns to Suspend mode.

10.The DMA can exit Suspend mode and enter the Run state (go to Step 1 or Step 2 depending on pending status) only after receiving a Transmit Poll demand (DMA Transmit Poll Demand Register).

The basic flow is charted in Figure 4.



TxDMA Operation in OSF Mode

Reception

The general reception sequence is depicted in Figure5 and proceeds as follows:

- 1.The host sets up Receive descriptors (RDES0-RDES3) and sets the Own bit (RDES0[31]).
- 2.Once the SR (DMA Operation Mode Register[1]) bit is set, the DMA enters the Run state. While in the Run state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors.
- 3.The DMA decodes the receive data buffer address from the acquired descriptors.
- 4.Incoming frames are processed and placed in the acquired descriptor's data buffers.

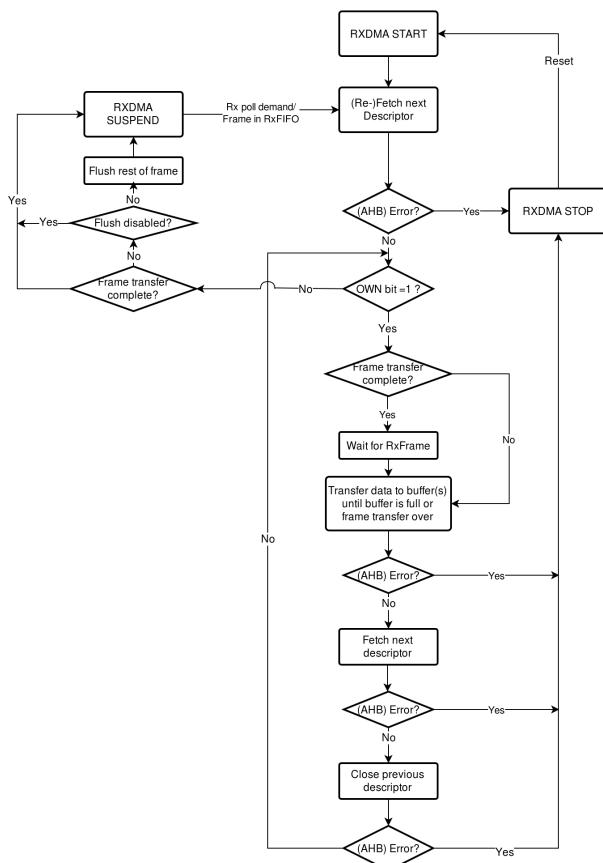
5. When the buffer is full or the frame transfer is complete, the Receive engine fetches the next descriptor.

6.The status information is written to RDES0 of the previous Receive descriptor with the frame's Own bit reset to 1'b0. If the frame transfer is not complete, the Descriptor Error bit is set and the DMA does not own the next descriptor.

7. The Receive engine checks the latest descriptor's Own bit. When the host owns a descriptor, the Own bit is 1'b0, the Receive Buffer Unavailable bit (Status Register[7]) is set and the Receive Engine enters the Suspended state. If the DMA owns the descriptor, the engine jumps to Step 4 and awaits the next frame.

8. Before the Receive engine enters the Suspend state, partial frames are flushed from the Receive FIFO (user-controllable).

9. The Receive DMA exits the Suspend state when a Receive Poll demand is given or the start of next frame is available from the MTL's Receive FIFO. The engine proceeds to Step 2 and fetches the next descriptor.



Receive DMA Operation

Interrupts

Interrupts can be generated as a result of various events. DMA Status Register contains all the bits that might cause an interrupt. DMA Interrupt Enable Register contains an Enable bit for each of the events that can cause an interrupt. There are two groups of interrupts, Normal and Abnormal, as described in DMA Status Register. Interrupts are cleared by writing a 1'b1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal `sbd_intr_o` is deasserted. If the MAC-Core is the cause for assertion of the interrupt, then any of the GLI, GMI, or GPI bits of DMA Status Register will be set high. Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt (DMA Status Register[6]) indicates that one or more frames was transferred to the Host buffer. The driver must scan all descriptors, from the last recorded position to the first one owned by the DMA. An interrupt is generated only once for simultaneous, multiple events. The driver must scan DMA Status Register for the interrupt cause. The interrupt is not generated again, unless a new interrupting event occurs after the driver has cleared the appropriate DMA Status Register bit. For example, the controller generates a Receive Interrupt (DMA Status Register[6]) and the

driver begins reading DMA Status Register. Next, Receive Buffer Unavailable (DMA Status Register[7]) occurs. The driver clears the Receive Interrupt. `sbd_intr_o` is deasserted for at least one cycle and then asserted again for the Receive Buffer Unavailable Interrupt.

MAC core

MAC core contains transmitter and receiver logic. Transmission is initiated when the MTL Application pushes in data with the SOF signal asserted. When the SOF signal is detected, the MAC core accepts the data and begins transmitting to the RMII. The time required to transmit the frame data to the RMII after the Application initiates transmission is variable, depending on delay factors like IFG delay, time to transmit preamble/SFD, and any back-off delays for Half-Duplex mode. Until then, the MAC core does not accept the data received from MTL. After the EOF is transferred to the MAC Core, the core complete normal transmission and then gives the Status of Transmission back to the MTL. If a normal collision (in Half-duplex mode) occurs during transmission, the MAC core makes valid the Transmit Status to the MTL. It will then accept and drop all further data until the next SOF is received. The MTL block should retransmit the same frame from SOF on observing a Retry request (in the Status) from the MAC core. The MAC core issues an underflow status if the MTL is not able to provide the data continuously during the transmission. During the normal transfer of a frame from MTL, if the MAC core receives a SOF without getting an EOF for the previous frame, then it (the SOF) is ignored and the new frame is considered as continuation of the previous frame.

A receive operation is initiated when the MAC core detects an SFD on the RMII. The core strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The received frame is stored in a shallow buffer until the address filtering is performed. The frame is dropped in the core if it fails the address filter.

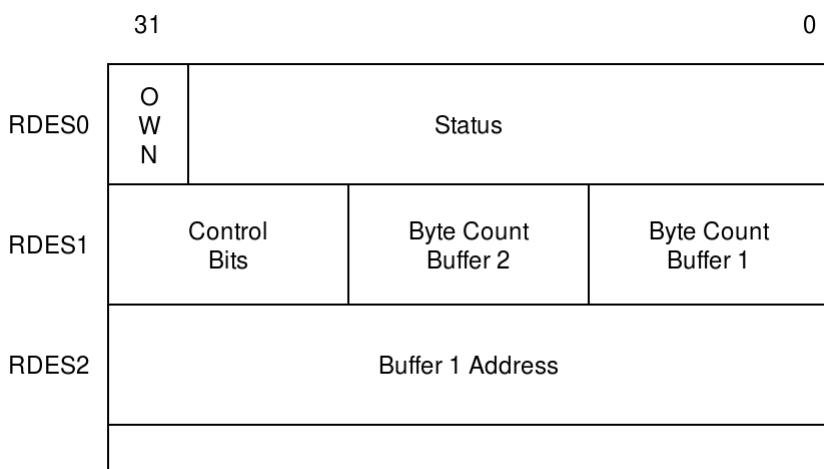
The MAC Transaction Layer provides FIFO memory to buffer and regulate the frames between the application system memory and the MAC core. It also enables the data to be transferred between the application clock domain and the MAC clock domains. The MTL layer has 2 data paths, namely the Transmit path and the Receive Path. The data path for both directions is 32-bit wide and operates with a simple FIFO protocol.

Descriptor Formats

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The default descriptor formats (both Receive and Transmit Descriptors) are explained in below sections

Receive Descriptor

The MAC Subsystem requires at least two descriptors when receiving a frame. The Receive state machine of the DMA always attempts to acquire an extra descriptor in anticipation of an incoming frame. (The size of the incoming frame is unknown). Before the RxDMA closes a descriptor, it will attempt to acquire the next descriptor even if no frames are received. In a single descriptor (receive) system, the subsystem will generate a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Thus, the Host is forced to increase either its descriptor pool or the buffer size. Otherwise, the subsystem starts dropping all incoming frames. The Rx descriptor format is shown in Figure 6.



| | |
|-------|--|
| RDES3 | Buffer 2 Address / Next Descriptor Address |
|-------|--|

Receive Descriptor Format

Receive Descriptor 0 (RDES0)

RDES0 contains the received frame status, the frame length, and the descriptor ownership information. It is described in below Table16-1.

| Bit | Description |
|-------|---|
| 31 | <p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA of the MAC-Core Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the Host.</p> <p>The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.</p> |
| 30 | <p>AFM: Destination Address Filter Fail</p> <p>When set, this bit indicates a frame that failed in the DA Filter in the MAC-Core.</p> |
| 29:16 | <p>FL: Frame Length</p> <p>These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid only when Last Descriptor (RDES0[8]) is set and Descriptor Error (RDES0[14]) is reset.</p> <p>The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame.</p> <p>This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.</p> |
| 15 | <p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • RDES0[1]: CRC Error • RDES0[3]: Receive Error • RDES0[4]: Watchdog Timeout • RDES0[6]: Late Collision • RDES0[7]: Giant Frame (This is not applicable when RDES0[7] indicates an IPV4 header Checksum error.) • RDES0[11]: Overflow Error • RDES0[14]: Descriptor Error <p>This field is valid only when the Last Descriptor (RDES0[8]) is set.</p> |

| Bit | Description |
|-----|---|
| 14 | <p>DE: Descriptor Error</p> <p>When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated.</p> <p>This field is valid only when the Last Descriptor (RDES0[8]) is set.</p> |
| 13 | <p>SAF: Source Address Filter Fail</p> <p>When set, this bit indicates that the SA field of frame failed the SA Filter in the MAC-Core.</p> |
| 12 | <p>LE: Length Error</p> <p>When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset.</p> |
| 11 | <p>OE: Overflow Error</p> <p>When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.</p> |
| 10 | <p>VLAN: VLAN Tag</p> <p>When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the MAC-Core.</p> |
| 9 | <p>FS: First Descriptor</p> <p>When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame.</p> <p>If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.</p> |
| 8 | <p>LS: Last Descriptor</p> <p>When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame</p> |
| 7 | <p>IPC Checksum Error/Giant Frame</p> <p>When set, this bit indicates that the 16-bit IPv4 Header checksum calculated by the core did not match the received checksum bytes.</p> <p>If this bit is set when Full Checksum Offload Engine (Type 2) is enabled, it indicates an error in the IPv4 or IPv6 header.</p> <p>This error can be due to inconsistent Ethernet Type field and IP header Version field values, a header checksum mismatch in IPv4, or an Ethernet frame lacking the expected number of IP header bytes.</p> <p>If IP Checksum Module is not selected during core configuration , this bit, when set, indicates that the received frame was a Giant Frame.</p> <p>Giant frames are larger-than-1,518-byte (or 1,522-byte for VLAN) normal frames and larger-than-9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled.</p> |
| 6 | <p>LC: Late Collision</p> <p>When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.</p> |

| Bit | Description |
|-----|---|
| 5 | FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes. |
| 4 | RWT: Receive Watchdog Timeout When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout. |
| 3 | RE: Receive Error When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error (rxd != 0f) during extension. |
| 2 | DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in RMII Mode. |
| 1 | CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set. |
| 0 | Rx MAC Address/Payload Checksum Error When set, this bit indicates that the Rx MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register0 value matched the DA field. If Full Checksum Offload Engine is enabled, this bit, when set, indicates the TCP, UDP, or ICMP checksum the core calculated does not match the received encapsulated TCP, UDP, or ICMP segment's Checksum field. This bit is also set when the received number of payload bytes does not match the value indicated in the Length field of the encapsulated IPv4 or IPv6 datagram in the received Ethernet frame. |

310 . Table 16-1 Receive Descriptor 0

Receive Descriptor 1 (RDES1)

RDES1 contains the buffer sizes and other bits that control the descriptor chain/ring. It is described in below Table 16-2.

| Bit | Description |
|-----|--|
| 31 | Disable Interrupt on Completion When set, this bit will prevent the setting of the RI (CSR5[6]) bit of the Status Register for the received frame that ends in the buffer pointed to by this descriptor. This, in turn, will disable the assertion of the interrupt to Host due to RI for that frame. |

| Bit | Description |
|-------|--|
| 30:26 | Reserved. |
| 25 | RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring. |
| 24 | RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24] is set, RBS2 (RDES1[21-11]) is a “don’t care” value. RDES1[25] takes precedence over RDES1[24]. |
| 23:22 | Reserved |
| 21:11 | RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size in bytes. The buffer size must be a multiple of 4/8/16 depending upon the bus widths (32/64/128), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. This field is not valid if RDES1[24] is set. |
| 10:0 | RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4/8/16 depending upon the bus widths (32/64/128), even if the value of RDES2 (buffer1 address pointer) is not aligned. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (bit 24). |

311 . Table 16-2 Receive Descriptor 1

Receive Descriptor 2 (RDES2)

RDES2 contains the address pointer to the first data buffer in the descriptor. It is described in below Table 16-3.

| Bit | Description |
|------|--|
| 31:0 | Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2:1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2:1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored. |

312 . Table 16-3 Receive Descriptor 2

Receive Descriptor 3 (RDES3)

RDES3 contains the address pointer either to the second data buffer in the descriptor or the next descriptor. It is described in below Table 16-4.

| Bit | Description |
|------|---|
| 31:0 | <p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>These bits indicate the physical address of Buffer 2 when descriptor chaining is used. If the Second Address Chained (RDES1[24]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned. However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition:</p> <p>The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[3, 2, or 1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored.</p> |

313 . Table 16-4 Receive Descriptor 3

Transmit Descriptor

The descriptor addresses must be aligned to the bus width used. Figure 7 shows the transmit descriptor format in Little-Endian mode with a 32-bit data bus.

Each descriptor is provided with two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory-management schemes.

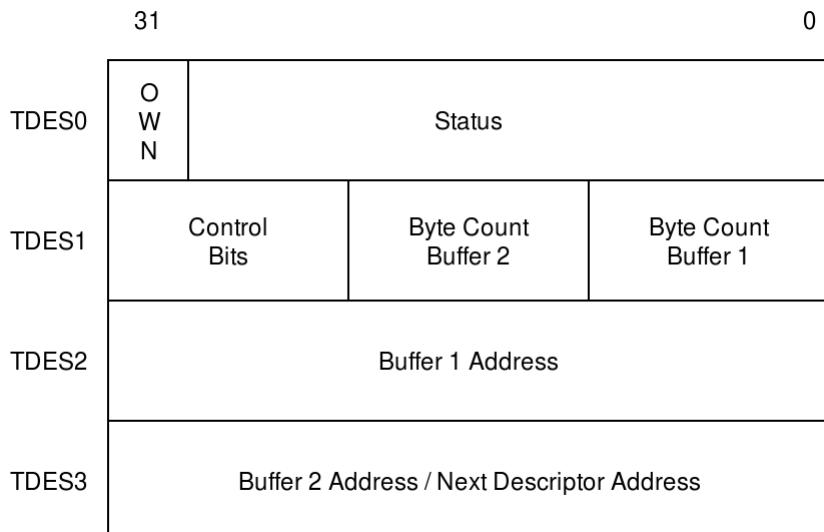


Figure 76: Transmit Descriptor Format

Transmit Descriptor 0 (TDES0)

TDES0 contains the transmitted frame status and the descriptor ownership information. It is described in below Table 16-5.

| Bit | Description |
|-------|--|
| 31 | <p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, this bit indicates that the descriptor is owned by the Host.</p> <p>The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty.</p> <p>The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set.</p> <p>This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.</p> |
| 30:17 | Reserved |
| 16 | <p>IHE: IP Header Error</p> <p>When set, this bit indicates that the Checksum Offload engine detected an IP header error and consequently did not modify the transmitted frame for any checksum insertion.</p> <p>This bit is valid only when Full Checksum Offload is enabled; otherwise, it is reserved.</p> |
| 15 | <p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error |
| 14 | <p>JT: Jabber Timeout</p> <p>When set, this bit indicates the MAC-Core transmitter has experienced a jabber time-out. This bit is only set when the MAC-Core configuration register's JD bit is not set.</p> |
| 13 | <p>FF: Frame Flushed</p> <p>When set, this bit indicates that the DMA/MTL flushed the frame due to a SW flush command given by the CPU.</p> |

| Bit | Description |
|-----|--|
| 12 | <p>PCE: Payload Checksum Error</p> <p>This bit, when set, indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload.</p> <p>This failure can be either due to insufficient bytes, as indicated by the IP Header's Payload Length field, or the MTL starting to forward the frame to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet frame being transmitted:</p> <p>to avoid deadlock, the MTL starts forwarding the frame when the FIFO is full, even in Store-and-Forward mode.</p> <p>When the Full Checksum Offload engine is not enabled during configuration, this bit is reserved.</p> |
| 11 | <p>LC: Loss of Carrier</p> <p>When set, this bit indicates that Loss of Carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission).</p> <p>This is valid only for the frames transmitted without collision and when the MAC-Core operates in Half-Duplex Mode.</p> |
| 10 | <p>NC: No Carrier</p> <p>When set, this bit indicates that the carrier sense signal from the PHY was not asserted during transmission.</p> |
| 9 | <p>LC: Late Collision</p> <p>When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in RMII Mode).</p> <p>Not valid if Underflow Error is set.</p> |
| 8 | <p>EC: Excessive Collision</p> <p>When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame.</p> <p>If the DR (Disable Retry) bit in the MAC-Core Configuration Register is set, this bit is set after the first collision and the transmission of the frame is aborted.</p> |
| 7 | <p>VF: VLAN Frame</p> <p>When set, this bit indicates that the transmitted frame was a VLAN type frame.</p> |
| 6:3 | <p>CC: Collision Count</p> <p>This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.</p> |
| 2 | <p>ED: Excessive Deferral</p> <p>When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000-Mbps mode, or in Jumbo Frame enabled mode)</p> <p>if the Deferral Check (DC) bit is set high in the MAC-Core Control Register.</p> |

| Bit | Description |
|-----|--|
| 1 | <p>UF: Underflow Error</p> <p>When set, this bit indicates that the MAC-Core aborted the frame because data arrived late from the Host memory.</p> <p>Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame.</p> <p>The transmission process enters the suspended state and sets both Transmit Underflow (Status Register[5]) and Transmit Interrupt (Status Register[0]).</p> |
| 0 | <p>DB: Deferred Bit</p> <p>When set, this bit indicates that the MAC-Core defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.</p> |

314 . Transmit Descriptor 0

Transmit Descriptor 1 (TDES1)

TDES1 contains the buffer sizes and other bits which control the descriptor chain/ring and the frame being transferred. It is described in below Table 16-6.

| Bit | Description |
|-----|---|
| 31 | <p>IC : Interrupt on Completion</p> <p>When set, this bit sets Transmit Interrupt (Status Register[0]) after the present frame has been transmitted.</p> |
| 30 | <p>LS: Last Segment</p> <p>When set, this bit indicates that the buffer contains the last segment of the frame.</p> |
| 29 | <p>FS: First Segment</p> <p>When set, this bit indicates that the buffer contains the first segment of a frame.</p> |

| Bit | Description |
|-------|--|
| 28:27 | <p>CIC: Checksum Insertion Control</p> <p>These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6 as described below.</p> <ul style="list-style-type: none"> • 2'b00: Do nothing. Checksum Engine is bypassed • 2'b01: Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. • 2'b10: Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame's Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4. • 2'b11: Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. In other words, the TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4. <p>The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly.</p> |
| 26 | <p>DC: Disable CRC</p> <p>When set, the MAC-Core does not append the Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES1[29]).</p> |
| 25 | <p>TER: Transmit End of Ring</p> <p>When set, this bit indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.</p> |
| 24 | <p>TCH: Second Address Chained</p> <p>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES1[24] is set, TBS2 (TDES1[21-11]) are “don’t care” values. TDES1[25] takes precedence over TDES1[24].</p> |
| 23 | <p>DP: Disable Padding</p> <p>When set, the MAC-Core does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes and the CRC field is added despite the state of the DC (TDES1[26]) bit. This is valid only when the first segment (TDES1[29]) is set.</p> |
| 22 | Reserved |
| 21:11 | <p>TBS2: Transmit Buffer 2 Size</p> <p>These bits indicate the Second Data Buffer in bytes. This field is not valid if TDES1[24] is set.</p> |
| 10:0 | <p>TBS1: Transmit Buffer 1 Size</p> <p>These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of TCH (bit 24).</p> |

315 . Transmit Descriptor 1

Transmit Descriptor 2 (TDES2)

TDES2 contains the address pointer to the first buffer of the descriptor. It is described in below Table 16-7.

| Bit | Description |
|------|--|
| 31:0 | <p>Buffer 1 Address Pointer</p> <p>These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment.</p> |

316 . Transmit Descriptor 2

Transmit Descriptor 3 (TDES3)

TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor. It is described in below Table 16-8.

| Bit | Description |
|------|---|
| 31:0 | <p>Buffer 2 Address Pointer (Next Descriptor Address)</p> <p>Indicates the physical address of Buffer 2 when the descriptor chaining is used. If the Second Address Chained (TDES1[24]) bit is set, then this address contains the pointer to the physical memory where the Next Descriptor is present.</p> <p>The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)</p> |

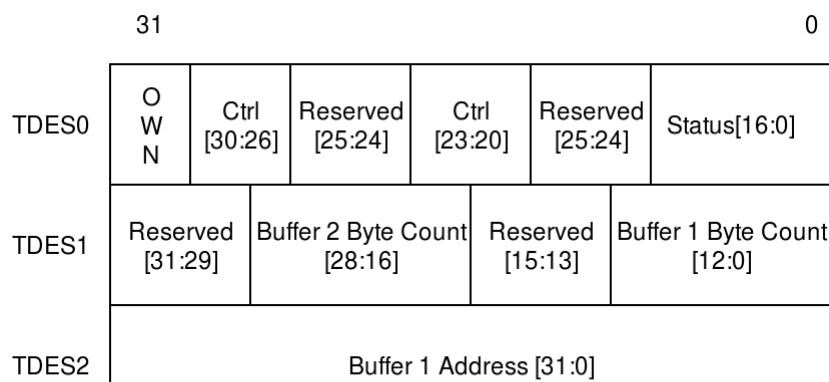
317 . Transmit Descriptor 3

Alternate (Enhanced) Descriptor Structure

The default descriptor structure allows data buffers of up to 2,048 bytes. An alternative descriptor structure has been implemented to support buffers of up to 8 KB (useful for Jumbo frames). This descriptor structure also consists of four 32-bit words, as in the default Receive and

Transmit Descriptors. These are shown in the two figures 8 and 9. The main difference from the default is the re-assignment of the Control and Status bits in TDES0, TDES1 and RDES1. Hence, only the description or bit-mapping of TDES0, TDES1, and RDES1 in the alternate

descriptor structure is given below.



| | |
|-------|--|
| TDES3 | Buffer 2 Address [31:0] / Next Descriptor Address [31:0] |
|-------|--|

Enhanced Transmit Descriptor

| | | | | | | |
|--------|---|------------------|-----------------------------|--------------|-------|----------------------------|
| | 31 | 0 | | | | |
| RDES 0 | O W N | Status[30:0] | | | | |
| RDES 1 | C T R L | Reserved [30:29] | Buffer 2 Byte Count [28:16] | Ctrl [15:14] | Resvd | Buffer 1 Byte Count [12:0] |
| RDES 2 | Buffer 1 Address[31:0] | | | | | |
| RDES 3 | Buffer 2 Address [31:0] or Next Descriptor Address [31:0] | | | | | |

Enhanced Receive Descriptor

TDES0 in Enhanced Descriptor Structure

The application software must program control bits 31:20 during descriptor initialization. When the DMA updates the descriptor (or writes it back), it resets all the control bits (including the Own bit) and reports only the status bits.

| Bit | Description |
|-----|--|
| 31 | <p>OWN: Own Bit</p> <p>When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host.</p> <p>The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are read completely.</p> <p>The ownership bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set.</p> <p>This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.</p> |
| 30 | <p>IC: Interrupt on Completion</p> <p>When set, this bit sets the Transmit Interrupt (Status Register[0]) after the present frame has been transmitted.</p> |
| 29 | <p>LS: Last Segment</p> <p>When set, this bit indicates that the buffer contains the last segment of the frame.</p> |

| Bit | Description |
|-------|---|
| 28 | <p>FS: First Segment</p> <p>When set, this bit indicates that the buffer contains the first segment of a frame.</p> |
| 27 | <p>DC: Disable CRC</p> <p>When this bit is set, the MAC-Core does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set.</p> |
| 26 | <p>DP: Disable Pad</p> <p>When set, the MAC-Core does not automatically add padding to a frame shorter than 64 bytes.</p> <p>When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit.</p> <p>This is valid only when the first segment (TDES0[28]) is set.</p> |
| 25:24 | Reserved |
| 23:22 | <p>CIC: Checksum Insertion Control</p> <p>These bits control the checksum calculation and insertion. Bit encodings are as shown below.</p> <ul style="list-style-type: none"> • 2'b00: Checksum Insertion Disabled. • 2'b01: Only IP header checksum calculation and insertion are enabled. • 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. • 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. <p>This field is reserved when the IPC_FULL_OFFLOAD configuration parameter is not selected.</p> |
| 21 | <p>TER: Transmit End of Ring</p> <p>When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.</p> |
| 20 | <p>TCH: Second Address Chained</p> <p>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a “don’t care” value.</p> <p>TDES0[21] takes precedence over TDES0[20].</p> |
| 19:17 | Reserved |
| 16 | <p>IHE: IP Header Error</p> <p>When set, this bit indicates that the MAC-Core transmitter detected an error in the IP datagram header. The transmitter checks the header length in the IPv4 packet against the number of header bytes received from the application and indicates an error status if there is a mismatch.</p> <p>For IPv6 frames, a header error is reported if the main header length is not 40 bytes. Furthermore, the Ethernet Length/Type field value for an IPv4 or IPv6 frame must match the IP header version received with the packet.</p> <p>For IPv4 frames, an error status is also indicated if the Header Length field has a value less than 0x5.</p> |

| Bit | Description |
|-----|--|
| 15 | <p>ES: Error Summary</p> <p>Indicates the logical OR of the following bits:</p> <ul style="list-style-type: none"> • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error • TDES0[16]: IP Header Error • TDES0[12]: IP Payload Error |
| 14 | <p>JT: Jabber Timeout</p> <p>When set, this bit indicates the MAC-Core transmitter has experienced a jabber time-out. This bit is only set when the MAC-Core configuration register's JD bit is not set.</p> |
| 13 | <p>FF: FrameFlushed</p> <p>When set, this bit indicates that the DMA/MTL flushed the frame due to a software Flush command given by the CPU.</p> |
| 12 | <p>IPE: IP Payload Error</p> <p>When set, this bit indicates that MAC-Core transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload.</p> <p>The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch.</p> |
| 11 | <p>LC: Loss of Carrier</p> <p>When set, this bit indicates that a loss of carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission).</p> <p>This is valid only for the frames transmitted without collision when the MAC-Core operates in Half-Duplex mode.</p> |
| 10 | <p>NC: No Carrier</p> <p>When set, this bit indicates that the Carrier Sense signal form the PHY was not asserted during transmission.</p> |
| 9 | <p>LC: Late Collision</p> <p>When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte-times, including preamble, in RMII mode).</p> <p>This bit is not valid if the Underflow Error bit is set.</p> |

| Bit | Description |
|-----|--|
| 8 | <p>EC: Excessive Collision</p> <p>When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame.</p> <p>If the DR (Disable Retry) bit in the MAC-Core Configuration register is set, this bit is set after the first collision and the transmission of the frame is aborted.</p> |
| 7 | <p>VF: VLAN Frame</p> <p>When set, this bit indicates that the transmitted frame was a VLAN-type frame.</p> |
| 6:3 | <p>CC: Collision Count</p> <p>This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted.</p> <p>The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.</p> |
| 2 | <p>ED: Excessive Deferral</p> <p>When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1,000-Mbps mode or if Jumbo Frame is enabled)</p> <p>if the Deferral Check (DC) bit in the MAC-Core Control register is set high.</p> |
| 1 | <p>UF: Underflow Error</p> <p>When set, this bit indicates that the MAC-Core aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the Suspended state and sets both Transmit Underflow (Status Register[5]) and Transmit Interrupt (Status Register[0]).</p> |
| 0 | <p>DB: Deferred Bit</p> <p>When set, this bit indicates that the MAC-Core defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.</p> |

318 . Table 16-9 Transmit Descriptor Word 0 (TDES0)

TDES1 in Enhanced Descriptor Structure

| Bit | Description |
|-------|---|
| 31:29 | Reserved |
| 28:16 | <p>TBS2: Transmit Buffer 2 Size</p> <p>These bits indicate the second data buffer size in bytes. This field is not valid if TDES0[20] is set.</p> |
| 15:13 | Reserved |
| 12:0 | <p>TBS1: Transmit Buffer 1 Size</p> <p>These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]).</p> |

319 . Transmit Descriptor Word 1

RDES1 in Enhanced Descriptor Structure

| Bit | Description |
|-------|---|
| 31 | DIC: Disable Interrupt on Completion When set, this bit prevents setting the Status Register's RI bit (CSR5[6]) for the received frame ending in the buffer indicated by this descriptor. This, in turn, disables the assertion of the interrupt to Host due to RI for that frame. |
| 30:29 | Reserved |
| 28:16 | RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size, in bytes. The buffer size must be a multiple of 4, 8, or 16, depending on the bus widths (32, 64, or 128, respectively), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. If the buffer size is not an appropriate multiple of 4, 8, or 16, the resulting behavior is undefined. This field is not valid if RDES1[14] is set. |
| 15 | RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring. |
| 14 | RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a "don't care" value. RDES1[15] takes precedence over RDES1[14]. |
| 13 | Reserved |
| 12:0 | RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, 8, or 16, depending upon the bus widths (32, 64, or 128), even if the value of RDES2 (buffer1 address pointer) is not aligned. When the buffer size is not a multiple of 4, 8, or 16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (bit 14). |

320 . Receive Descriptor Word 1 (RDES1)

15.2.4 Register Summary

Base Address: 0x2024_0000

Ethernet DMA(E-DMA or ED) Register Map

| Register Name | Offset | Description |
|---|--------|--|
| ED Bus Mode Register | 0x1000 | Controls the Host Interface Mode. |
| ED Transmit Poll Demand Register | 0x1004 | Used by the host to instruct the DMA to poll the Transmit Descriptor List. |
| ED Receive Poll Demand Register | 0x1008 | Used by the Host to instruct the DMA to poll the Receive Descriptor list. |
| ED Receive Descriptor List Address Register | 0x100C | Points the DMA to the start of the Receive Descriptor list. |

| Register Name | Offset | Description |
|--|---------------|--|
| ED Transmit Descriptor List Address Register | 0x1010 | Points the DMA to the start of the Transmit Descriptor List. |
| ED Status Register | 0x1014 | The Software driver (application) reads this register during interrupt service routine or polling to determine the status of the DMA |
| ED Status Register | 0x1014 | The Software driver (application) reads this register during interrupt service routine or polling to determine the status of the DMA. |
| ED Operation Mode Register | 0x1018 | Establishes the Receive and Transmit operating modes and command |
| ED Interrupt Enable Register | 0x101C | Enables the interrupts reported by the Status Register. |
| ED Missed Frame and Buffer Overflow Counter Register | 0x1020 | Contains the counters for discarded frames because no host Receive Descriptor was available and discarded frames because of Receive FIFO Overflow. |
| Reserved | 0x1024–0x1044 | Reserved |
| ED Current Host Transmit Descriptor Register | 0x1048 | Points to the start of current Transmit Descriptor read by the DMA. |
| ED Current Host Receive Descriptor Register | 0x104C | Points to the start of current Receive Descriptor read by the DMA. |
| ED Current Host Transmit Buffer Address Register | 0x1050 | Points to the current Transmit Buffer address read by the DMA. |
| ED Current Host Receive Buffer Address Register | 0x1054 | Points to the current Receive Buffer address read by the DMA. |

321 .DMA Register Summary Table

MAC Register Map

| Register Name | Offset | Description |
|------------------------------|--------|---|
| MAC Configuration Register | 0x0000 | This is the operation mode register for the MAC. |
| MAC Frame Filter | 0x0004 | Contains the frame filtering controls. |
| MAC Hash Table High Register | 0x0008 | Contains the higher 32 bits of the Multicast Hash table. |
| MAC Hash Table Low Register | 0x000C | Contains the lower 32 bits of the Multicast Hash table. |
| MAC GMII Address Register | 0x0010 | Controls the management cycles to an external PHY through management interface. |
| MAC GMII Data Register | 0x0014 | Contains the data to be written to or read from the PHY register located at the address specified in GMII Address Register. |
| MAC Flow Control Register | 0x0018 | Controls the generation of control frames. |
| MAC VLAN Tag Register | 0x001C | Identifies IEEE 802.1Q VLAN type frames. |
| MAC Version Register | 0x0020 | Identifies the version of the Core |
| Reserved | 0x0024 | Reserved |

| Register Name | Offset | Description |
|---------------------------------|---------------|---|
| MAC Remote Wake-Up Frame Filter | 0x0028 | <p>This is the address through which the remote Wake-up Frame Filter registers</p> <p>(wkupfmfilter_reg) are written/read by the Application.</p> <p>wkupfmfilter_reg is actually a pointer to eight (not transparent) such wkupfmfilter_reg registers. Eight sequential Writes to this address (028) will write all wkupfmfilter_reg</p> <p>Eight sequential Reads from this address (028) will read all wkupfmfilter_reg registers. This register contains the higher 16 bits of the 7th MAC address.</p> |
| MAC PMT Control and Status | 0x002C | Programs the request wake-up events and monitor the wake-up events. |
| Reserved | 0x0030–0x0034 | Reserved |
| MAC Interrupt Register | 0x0038 | Contains the interrupt status. |
| MAC Interrupt Mask Register | 0x003C | Contains the masks for generating the interrupts. |
| MAC Address0 High Register | 0x0040 | Contains the higher 16 bits of the first MAC address. |
| MAC Address0 Low Register | 0x0044 | Contains the lower 32 bits of the first MAC address. |
| MAC Address1 High Register | 0x0048 | Contains the higher 16 bits of the second MAC address. |
| MAC Address1 Low Register | 0x004C | Contains the lower 32 bits of the second MAC address. |
| MAC Address2 High Register | 0x0050 | Contains the higher 32 bits of the third MAC address. |
| MAC Address2 Low Register | 0x0054 | Contains the lower 32 bits of the third MAC address. |
| MAC Address3 High Register | 0x0058 | Contains the higher 16 bits of the fourth MAC address. |
| MAC Address3 Low Register | 0x005C | Contains the lower 32 bits of the fourth MAC address. |
| MAC Address4 High Register | 0x0060 | Contains the higher 16 bits of the fourth MAC address. |
| MAC Address4 Low Register | 0x0064 | Contains the lower 32 bits of the fourth MAC address. |
| Reserved | 0x0068–0x00D8 | Reserved |
| Reserved | 0x00DC–0x00FC | Reserved |
| MMC Register Map | 0x0100–0x02FC | MMC Register Map |
| Reserved | 0x0800–0x0FFC | Reserved |

322 . MAC Register Summary Table

MMC Register Map

| Register Name | Offset | Description |
|---------------|--------|---|
| mmc_cntrl | 0x0100 | MMC Control establishes the operating mode of MMC. |
| mmc_intr_rx | 0x0104 | MMC Receive Interrupt maintains the interrupt generated from all of the receive statistic counters. |
| mmc_intr_tx | 0x0108 | MMC Transmit Interrupt maintains the interrupt generated from all of the transmit statistic counters. |

| Register Name | Offset | Description |
|------------------|---------------|---|
| mmc_intr_mask_rx | 0x010C | MMC Receive Interrupt mask maintains the mask for the interrupt generated from all of the receive statistic counters. |
| mmc_intr_mask_tx | 0x0110 | MMC Transmit Interrupt Mask maintains the mask for the interrupt generated from all of the transmit statistic counters. |
| Reserved | 0x0114–0x0144 | |
| txunderflowerror | 0x0148 | Number of frames aborted due to frame underflow error. |
| txsinglecol_g | 0x014C | Number of successfully transmitted frames after a single collision in Half-duplex mode. |
| Reserved | 0x0150 | |
| txdeferred | 0x0154 | Number of successfully transmitted frames after a deferral in Half-duplex mode. |
| Reserved | 0x0158–0x0164 | |
| txframecount_g | 0x0168 | Number of good frames transmitted. |
| Reserved | 0x016C–0x017C | |
| rxframecount_gb | 0x0180 | Number of good and bad frames received. |
| Reserved | 0x0184–0x0188 | |
| rxcrcerror | 0x0194 | Number of frames received with CRC error. |
| Reserved | 0x0198–0x01D0 | |
| rxfifooverflow | 0x01D4 | Number of missed received frames due to FIFO overflow. This counter is not present in the MAC-CORE configuration. |
| Reserved | 0x01D8–0x02FC | |

323 . MMC Register Summary Table

Legend:

R = Read-only, W = Write-only, R/W = Read/Write

15.2.5 DMA Register Description

E-DMA Bus Mode Register

The Bus Mode register establishes the bus operating modes for the DMA.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:26 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------|-----------|--|
| 25 | R/W | AAL | 0 | <p>Address-Aligned Beats</p> <p>When this bit is set high and the FB bit equals 1, the AHB interface generates all bursts aligned to the start address LS bits. If the FB bit equals 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address.</p> |
| 24 | R/W | 4xPBL Mode | 0 | <p>When set high, this bit multiplies the PBL value programmed (bits [22:17] and bits [13:8]) four times. Thus the DMA will transfer data in to a maximum of 4, 8, 16, 32, 64 and 128 beats depending on the PBL value.</p> |
| 23 | R/W | USP | 0 | <p>Use Separate PBL</p> <p>When set high, it configures the RxDMA to use the value configured in bits [22:17] as PBL while the PBL value in bits [13:8] is applicable to TxDMA operations only. When reset to low, the PBL value in bits [13:8] is applicable for both DMA engines.</p> |
| 22:17 | R/W | RPBL | 0x1 | <p>RxDMA PBL</p> <p>These bits indicate the maximum number of beats to be transferred in one RxDMA transaction. This will be the maximum value that is used in a single block Read/Write. The RxDMA will always attempt to burst as specified in RPBL each time it starts a Burst transfer on the host bus. RPBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value will result in undefined behavior.</p> <p>These bits are valid and applicable only when USP is set high.</p> |
| 16 | R/W | FB | 0 | <p>Fixed Burst</p> <p>This bit controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When reset, the AHB will use SINGLE and INCR burst transfer operations.</p> |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 15:14 | R/W | PR | 0 | <p>Rx:Tx priority ratio. RxDMA requests given priority over TxDMA requests in the following ratio. This is valid only when the DA bit is reset.</p> <ul style="list-style-type: none">• 00: 1:1• 01: 2:1• 10: 3:1• 11: 4:1 |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------|----------------|-----------------|-----------|--|----------------|------------|-----------------|----|-----------|------------|--|-----------|------------|--|-----------|------------|--|----------------|-----|----|-----------|-----------|--|-----------|------------|--|-----------|------------|--|----------------|-----|-----|-----------|-----------|--|-----------|-----------|--|-----------|------------|--|------|------------|
| 13:8 | R/W | PBL | 0x1 | <p>Programmable Burst Length</p> <p>These bits indicate the maximum number of beats to be transferred in one DMA transaction. This will be the maximum value that is used in a single block Read/</p> <p>Write. The DMA will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value will result in undefined behavior. When USP is set high, this PBL value is applicable for TxDMA transactions only.</p> <p>The PBL values have the following limitations.</p> <p>The maximum number of beats (PBL) possible is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO. For different data bus widths and FIFO sizes, the valid PBL range (including x4 mode) is provided in the following table. If the PBL is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered. Do not program out-of-range PBL values, because the system may not behave properly.</p> <table border="1"> <thead> <tr> <th>Data Bus Width</th><th>FIFO Depth</th><th>Valid PBL Range</th></tr> </thead> <tbody> <tr> <td>32</td><td>128 bytes</td><td>16 or less</td></tr> <tr> <td></td><td>256 bytes</td><td>32 or less</td></tr> <tr> <td></td><td>512 bytes</td><td>64 or less</td></tr> <tr> <td></td><td>1 KB and above</td><td>All</td></tr> <tr> <td>64</td><td>128 bytes</td><td>8 or less</td></tr> <tr> <td></td><td>256 bytes</td><td>16 or less</td></tr> <tr> <td></td><td>512 bytes</td><td>32 or less</td></tr> <tr> <td></td><td>1 KB and above</td><td>All</td></tr> <tr> <td>128</td><td>128 bytes</td><td>4 or less</td></tr> <tr> <td></td><td>256 bytes</td><td>8 or less</td></tr> <tr> <td></td><td>512 bytes</td><td>16 or less</td></tr> <tr> <td></td><td>1 KB</td><td>32 or less</td></tr> </tbody> </table> | Data Bus Width | FIFO Depth | Valid PBL Range | 32 | 128 bytes | 16 or less | | 256 bytes | 32 or less | | 512 bytes | 64 or less | | 1 KB and above | All | 64 | 128 bytes | 8 or less | | 256 bytes | 16 or less | | 512 bytes | 32 or less | | 1 KB and above | All | 128 | 128 bytes | 4 or less | | 256 bytes | 8 or less | | 512 bytes | 16 or less | | 1 KB | 32 or less |
| Data Bus Width | FIFO Depth | Valid PBL Range | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 32 | 128 bytes | 16 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 256 bytes | 32 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 512 bytes | 64 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1 KB and above | All | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 64 | 128 bytes | 8 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 256 bytes | 16 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 512 bytes | 32 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1 KB and above | All | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 128 | 128 bytes | 4 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 256 bytes | 8 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 512 bytes | 16 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1 KB | 32 or less | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description | | |
|-----|--------|----------|-----------|--|------------|-----------------|
| | | | | Data Bus Width | FIFO Depth | Valid PBL Range |
| | | | | 2 KB | 64 or less | All |
| 7 | R | Reserved | 0 | Reserved | | |
| 6:2 | R/W | DSL | 0 | Descriptor Skip Length This bit specifies the number of Word/Dword/Lword (depending on 32/64/128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When DSL value equals zero, then the descriptor table is taken as contiguous by the DMA, in Ring mode. | | |
| 1 | R/W | DA | 0 | DMA Arbitration scheme <ul style="list-style-type: none"> • 0: Round-robin with Rx:Tx priority given in bits [15:14] • 1: Rx has priority over Tx | | |
| 0 | R/W* | SWR | 0 | Software Reset When this bit is set, the MAC DMA Controller resets all MAC-Core Subsystem internal registers and logic. It is cleared automatically after the reset operation has completed in all of the core clock domains. Read a 0 value in this bit before re-programming any register of the core. | | |

*- Self Clear bit

324 . Bus Mode Register Description

E-DMA Transmit Poll Demand Register

The Transmit Poll Demand register enables the Transmit DMA to check whether or not the current descriptor is owned by DMA. The Transmit Poll Demand command is given to wake up the TxDMA if it is in Suspend mode.

The TxDMA can go into Suspend mode due to an Underflow error in a transmitted frame or due to the unavailability of descriptors owned by Transmit DMA.

This command can be given anytime and the TxDMA will reset this command once it starts re-fetching the current descriptor from host memory.

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 0 | R/W* | TPD | 0 | <p>Transmit Poll Demand</p> <p>When these bits are written with any value, the DMA reads the current descriptor pointed to by Current Host Transmit Descriptor Register.</p> <p>If that descriptor is not available (owned by Host), transmission returns to the Suspend state and DMA Status Register[2] is asserted. If the descriptor is available, transmission resumes.</p> |

*-Register field can be read by the application, and when a write operation is performed with any data value, an event is triggered

325 . Transmit Poll Demand Register Description

E-DMA Receive Poll Demand Register

The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is given to wake up the RxDMA from SUSPEND state. The RxDMA can go into SUSPEND state only due to the unavailability of descriptors owned by it.

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W* | RPD | 0 | <p>Receive Poll Demand</p> <p>When these bits are written with any value, the DMA reads the current descriptor pointed to by Current Host Receive Descriptor Register. If that descriptor is not available (owned by Host), reception returns to the Suspended state and Status Register[7] is not asserted. If the descriptor is available, the Receive DMA returns to active state.</p> |

*-Register field can be read by the application, and when a write operation is performed with any data value, an event is triggered

326 . Receive Poll Demand Register Description

E-DMA Receive Descriptor List Address Register

The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word/Dword/Lword-aligned (for 32/64/128-bit data bus).

The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this Register is permitted only when reception is stopped. When stopped, this Register must be written to before the receive Start command is given.

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------|-----------|---|
| 31:0 | R/W | Start of Receive List | 0 | This field contains the base address of the First Descriptor in the Receive Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus width) will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are Read Only. |

327 . Receive Descriptor List Address Register Description

E-DMA Transmit Descriptor List Address Register

The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word/Dword/Lword-aligned (for 32/64/128-bit data bus).

The DMA internally converts it to bus width aligned address by making the corresponding LSB to low. Writing to this Register is permitted only when transmission has stopped. When stopped, this Register can be written before the transmission Start command is given.

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------|-----------|--|
| 31:0 | R/W | Start of Transmit List | 0 | This field contains the base address of the First Descriptor in the Transmit Descriptor list. The LSB bits [1/2/3:0] for 32/64/128-bit bus width) will be ignored and taken as all-zero by the DMA internally. Hence these LSB bits are Read Only. |

328 . Transmit Descriptor List Address Register Description

E-DMA Status Register

The Status register contains all the status bits that the DMA reports to the host. This register is usually read by the Software driver during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted.

This register bits are not cleared when read. Writing 1'b1 to (unreserved) bits in this register[16:0] clears them and writing 1'b0 has no effect. Each field (bits[16:0]) can be masked by masking the appropriate bit in Interrupt Enable Register.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:29 | R | Reserved | 0 | Reserved |
| 28 | R | GPI | 0 | MAC-Core PMT Interrupt This bit indicates an interrupt event in the MAC-Core's PMT module. The software must read the corresponding registers in the MAC-Core to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. The interrupt signal from the MAC-Core subsystem (sbd_intr_o) is high when this bit is high. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 27 | R | GMI | 0 | <p>MAC-Core MMC Interrupt</p> <p>This bit reflects an interrupt event in the MMC module of the MAC-Core. The software must read the corresponding registers in the MAC-Core to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the MAC-Core subsystem (sbd_intr_o) is high when this bit is high.</p> |
| 26 | R | Reserved | 0 | Reserved |
| 25:23 | R | EB | 0 | <p>Error Bits</p> <p>These bits indicate the type of error that caused a Bus Error (error response on the AHB interface). Valid only with Fatal Bus Error bit (Status Register[13]) set.</p> <p>This field does not generate an interrupt.</p> <ul style="list-style-type: none"> Bit 23 1'b1 Error during data transfer by TxDMA 1'b0 Error during data transfer by RxDMA Bit 24 1'b1 Error during read transfer 1'b0 Error during write transfer Bit 25 1'b1 Error during descriptor access 1'b0 Error during data buffer access |
| 22:20 | R | TS | 0 | <p>Transmit Process State</p> <p>These bits indicate the Transmit DMA FSM state. This field does not generate an interrupt.</p> <ul style="list-style-type: none"> • 3'b000: Stopped; Reset or Stop Transmit Command issued. • 3'b001: Running; Fetching Transmit Transfer Descriptor. • 3'b010: Running; Waiting for status. • 3'b011: Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO). • 3'b100, 3'b101: Reserved for future use. • 3'b110: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow. • 3'b111: Running; Closing Transmit Descriptor. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 19:17 | R | RS | 0 | <p>Receive Process State</p> <p>These bits indicate the Receive DMA FSM state. This field does not generate an interrupt.</p> <ul style="list-style-type: none"> • 3'b000: Stopped: Reset or Stop Receive Command issued. • 3'b001: Running: Fetching Receive Transfer Descriptor. • 3'b010: Reserved for future use. • 3'b011: Running: Waiting for receive packet. • 3'b100: Suspended: Receive Descriptor Unavailable. • 3'b101: Running: Closing Receive Descriptor. • 3'b110: Reserved for future use. • 3'b111: Running: Transferring the receive packet data from receive buffer to host memory. |
| 16 | R/W* | NIS | 0 | <p>Normal Interrupt Summary</p> <p>Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Interrupt Enable Register:</p> <ul style="list-style-type: none"> • Status Register[0]: Transmit Interrupt • Status Register[2]: Transmit Buffer Unavailable • Status Register[6]: Receive Interrupt • Status Register[14]: Early Receive Interrupt <p>Only unmasked bits affect the Normal Interrupt Summary bit.</p> <p>This is a sticky bit and must be cleared (by writing a 1 to this bit) each time a corresponding bit that causes NIS to be set is cleared.</p> |

| Bit | Access | Function | POR Value | Description |
|-------|------------------|----------|-----------|--|
| 15 | R/W* | AIS | 0 | <p>Abnormal Interrupt Summary</p> <p>Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Interrupt Enable Register:</p> <ul style="list-style-type: none"> • Status Register[1]:Transmit Process Stopped • Status Register[3]:Transmit Jabber Timeout • Status Register[4]: Receive FIFO Overflow • Status Register[5]: Transmit Underflow • Status Register[7]: Receive Buffer Unavailable • Status Register[8]: Receive Process Stopped • Status Register[9]: Receive Watchdog Timeout • Status Register[10]: Early Transmit Interrupt • Status Register[13]: Fatal Bus Error <p>Only unmasked bits affect the Abnormal Interrupt Summary bit.</p> <p>This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared.</p> |
| 14 | R/W ¹ | ERI | 0 | <p>Early Receive Interrupt</p> <p>This bit indicates that the DMA had filled the first data buffer of the packet. Receive Interrupt Status Register[6] automatically clears this bit.</p> |
| 13 | R/W* | FBI | 0 | <p>Fatal Bus Error Interrupt</p> <p>This bit indicates that a Bus Error occurred (Status Register[25:23]). When this bit is set, the corresponding DMA engine disables all its bus accesses.</p> |
| 12:11 | R | Reserved | 0 | Reserved |
| 10 | R/W* | ETI | 0 | <p>Early Transmit Interrupt</p> <p>This bit indicates that the frame to be transmitted was fully transferred to the MTL Transmit FIFO.</p> |
| 9 | R/W* | RWT | 0 | <p>Receive Watchdog Timeout</p> <p>This bit is asserted when a frame with a length greater than 2,048 bytes is received (10,240 when Jumbo Frame mode is enabled).</p> |
| 8 | R/W* | RPS | 0 | <p>Receive Process Stopped</p> <p>This bit is asserted when the Receive Process enters the Stopped state.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 7 | R/W* | RU | 0 | <p>Receive Buffer Unavailable</p> <p>This bit indicates that the Next Descriptor in the Receive List is owned by the host and cannot be acquired by the DMA. Receive Process is suspended.</p> <p>To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, Receive Process resumes when the next recognized incoming frame is received. Status Register[7] is set only when the previous Receive Descriptor was owned by the DMA.</p> |
| 6 | R/W* | RI | 0 | <p>Receive Interrupt</p> <p>This bit indicates the completion of frame reception. Specific frame status information has been posted in the descriptor. Reception remains in the Running state.</p> |
| 5 | R/W* | UNF | 0 | <p>Transmit Underflow</p> <p>This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.</p> |
| 4 | R/W* | OVF | 0 | <p>Receive Overflow</p> <p>This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to application, the overflow status is set in RDES0[11].</p> |
| 3 | R/W* | TJT | 0 | <p>Transmit Jabber Timeout</p> <p>This bit indicates that the Transmit Jabber Timer expired, meaning that the transmitter had been excessively active. The transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.</p> |
| 2 | R/W* | TU | 0 | <p>Transmit Buffer Unavailable</p> <p>This bit indicates that the Next Descriptor in the Transmit List is owned by the host and cannot be acquired by the DMA. Transmission is suspended.</p> <p>Bits[22:20] explain the Transmit Process state transitions. To resume processing transmit descriptors, the host should change the ownership of the bit of the descriptor and then issue a Transmit Poll Demand command.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 1 | R/W* | TPS | 0 | Transmit Process Stopped This bit is set when the transmission is stopped. |
| 0 | R/W* | TI | 0 | Transmit Interrupt This bit indicates that frame transmission is finished and TDES1[31] is set in the First Descriptor. |

*- Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and can be cleared to 1'b0 by the application with a register write of 1'b1 (Write Clear). A register write of 1'b0 has no effect on this field.

¹ - Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and can be cleared to 1'b0 either by the core itself (Self Clear) or by the application with a register write of 1'b0 (Write Clear). A register write of 1'b1 to this bit has to no effect on this field.

329 . Status Register Description

E-DMA Operation Mode Register

The Operation Mode register establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of DMA initialization.

This register is also present in the MAC-Core-MTL configuration with Bits 13, 2, and 1 unused and reserved.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:27 | R | Reserved | 0 | Reserved |
| 26 | R/W | DT | 0 | Disable Dropping of TCP/IP Checksum Error Frames When this bit is set, the core does not drop frames that only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors in the encapsulated payload only. When this bit is reset, all error frames are dropped if the FEF bit is reset. If the Full Checksum Offload engine (Type 2) is disabled, this bit is reserved (Read Only with value 1'b0). |
| 25 | R/W | RSF | 0 | Receive Store and Forward When this bit is set, the MTL only reads a frame from the Rx FIFO after the complete frame has been written to it, ignoring RTC bits. When this bit is reset, the Rx FIFO operates in Cut-Through mode, subject to the threshold specified by the RTC bits. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 24 | R/W | DFF | 0 | Disable Flushing of Received Frames When this bit is set, the RxDMA does not flush any frames due to the unavailability of receive descriptors/buffers as it does normally when this bit is reset. |
| 23:22 | R | Reserved | 0 | Reserved |
| 21 | R/W | TSF | 0 | Transmit Store and Forward When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Operation Mode Register[16:14] are ignored. This bit should be changed only when transmission is stopped. |
| 20 | R/W | FTF | 0 | Flush Transmit FIFO When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost/flushed. This bit is cleared internally when the flushing operation is completed fully. The Operation Mode register should not be written to until this bit is cleared. |
| 19:17 | R | Reserved | 0 | Reserved |
| 16:14 | R/W | TTC | 0 | Transmit Threshold Control These three bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when the TSF bit (Bit 21) is reset. <ul style="list-style-type: none"> • 000: 64 • 001: 128 • 010: 192 • 011: 256 • 100: 40 • 101: 32 • 110: 24 • 111: 16 |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 13 | R/W | ST | 0 | <p>Start/Stop Transmission Command</p> <p>When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted.</p> <p>Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Transmit Descriptor List Address Register, or</p> <p>from the position retained when transmission was stopped previously. If the current descriptor is not owned by the DMA, transmission enters the Suspended state and</p> <p>Transmit Buffer Unavailable (Status Register[2]) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting DMA Register Transmit Descriptor List Address Register, then the DMA behavior is unpredictable.</p> <p>When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and becomes the current position when transmission is restarted. The stop transmission command is effective only the transmission of the current frame is complete or when the transmission is in the Suspended state.</p> |
| 12:11 | R/W | RFD | 0 | <p>Threshold for deactivating flow control (in both HD and FD)</p> <p>These bits control the threshold (Fill-level of Rx FIFO) at which the flow-control is deasserted after activation.</p> <ul style="list-style-type: none"> • 00: Full – 1 KB • 01: Full – 2 KB • 10: Full – 3 KB • 11: Full – 4 KB <p>Note that the de-assertion is effective only after flow control is asserted. If the Rx FIFO is 8 KB or more, an additional bit (RFD[2]) is used for more threshold levels as described in bit [22].</p> |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 10:9 | R/W | RFA | 0 | <p>Threshold for activating flow control (in both HD and FD)</p> <p>These bits control the threshold (Fill level of Rx FIFO) at which flow control is activated.</p> <ul style="list-style-type: none"> • 00: Full – 1 KB • 01: Full – 2 KB • 10: Full – 3 KB • 11: Full – 4 KB <p>Note that the above only applies to Rx FIFOs of 4 KB or more when the EFC bit is set high. If the Rx FIFO is 8 KB or more, an additional bit (RFA[2]) is used for more threshold levels as described in bit [23].</p> |
| 8 | R/W | EFC | 0 | <p>Enable HW flow control</p> <p>When this bit is set, the flow control signal operation based on fill-level of Rx FIFO is enabled. When reset, the flow control operation is disabled.</p> <p>This bit is not used (reserved and always reset) when the Rx FIFO is less than 4 KB.</p> |
| 7 | R/W | FEF | 0 | <p>Forward Error Frames</p> <p>When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, GMII_ER, giant frame, watchdog timeout, overflow). However, if the frame's start byte (write) pointer is already transferred to the read controller side (in Threshold mode), then the frames are not dropped. Note that in MAC MTL configuration in which the Frame Length FIFO is also enabled during coreKit configuration, the Rx FIFO drops the error frames if that frame's start byte is not transferred (output) on the ARI bus. When FEF is set, all frames except runt error frames are forwarded to the DMA.</p> |
| 6 | R/W | FUF | 0 | <p>Forward Undersized Good Frames</p> <p>When set, the Rx FIFO will forward Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC).</p> <p>When reset, the Rx FIFO will drop all frames of less than 64 bytes, unless it is already transferred due to lower value of Receive Threshold (e.g., RTC = 01).</p> |
| 5 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 4:3 | R/W | RTC | 0 | <p>Receive Threshold Control</p> <p>These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are transferred automatically. Note that value of 11 is not applicable if the configured Receive FIFO size is 128 bytes. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1.</p> <ul style="list-style-type: none"> • 00: 64 • 01: 32 • 10: 96 • 11: 128 |
| 2 | R/W | OSF | 0 | <p>Operate on Second Frame</p> <p>When this bit is set, this bit instructs the DMA to process a second frame of Transmit data even before status for first frame is obtained.</p> |
| 1 | R/W | SR | 0 | <p>Start/Stop Receive</p> <p>When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes incoming frames. Descriptor acquisition is attempted from the current position in the list, which is the address set by DMA Receive Descriptor List Address Register or the position retained when the Receive process was previously stopped. If no descriptor is owned by the DMA, reception is suspended and Receive Buffer Unavailable (Status Register[7]) is set. The Start Receive command is effective only when reception has stopped. If the command was issued before setting DMA Receive Descriptor List Address Register, DMA behavior is unpredictable.</p> <p>When this bit is cleared, RxDMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|-------------|
| 0 | R | Reserved | 0 | Reserved |

330 . Operation Mode Register Description

E-DMA Interrupt Enable Register

The Interrupt Enable register enables the interrupts reported by Status Register. Setting a bit to 1'b1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:17 | R | Reserved | 0 | Reserved |
| 16 | R/W | NIE | 0 | <p>Normal Interrupt Summary Enable</p> <p>When this bit is set, a normal interrupt is enabled. When this bit is reset, a normal interrupt is disabled. This bit enables the following bits:</p> <ul style="list-style-type: none"> • Status Register[0]: Transmit Interrupt • Status Register[2]: Transmit Buffer Unavailable • Status Register[6]: Receive Interrupt • Status Register[14]: Early Receive Interrupt |
| 15 | R/W | AIE | 0 | <p>Abnormal Interrupt Summary Enable</p> <p>When this bit is set, an Abnormal Interrupt is enabled. When this bit is reset, an Abnormal Interrupt is disabled. This bit enables the following bits</p> <ul style="list-style-type: none"> • Status Register[1]: Transmit Process Stopped • Status Register[3]: Transmit Jabber Timeout • Status Register[4]: Receive Overflow • Status Register[5]: Transmit Underflow • Status Register[7]: Receive Buffer Unavailable • Status Register[8]: Receive Process Stopped • Status Register[9]: Receive Watchdog Timeout • Status Register[10]: Early Transmit Interrupt • Status Register[13]: Fatal Bus Error |
| 14 | R/W | ERE | 0 | <p>Early Receive Interrupt Enable</p> <p>When this bit is set with Normal Interrupt Summary Enable (Interrupt Enable Register[16]), Early Receive Interrupt is enabled.</p> <p>When this bit is reset, Early Receive Interrupt is disabled.</p> |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 13 | R/W | FBE | 0 | <p>Fatal Bus Error Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), the Fatal Bus Error Interrupt is enabled.</p> <p>When this bit is reset, Fatal Bus Error Enable Interrupt is disabled.</p> |
| 12:11 | R | Reserved | 0 | Reserved |
| 10 | R/W | ETE | 0 | <p>Early Transmit Interrupt Enable</p> <p>When this bit is set with an Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), Early Transmit Interrupt is enabled.</p> <p>When this bit is reset, Early Transmit Interrupt is disabled.</p> |
| 9 | R/W | RWE | 0 | <p>Receive Watchdog Timeout Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), the Receive Watchdog Timeout Interrupt is enabled.</p> <p>When this bit is reset, Receive Watchdog Timeout Interrupt is disabled.</p> |
| 8 | R/W | RSE | 0 | <p>Receive Stopped Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), Receive Stopped Interrupt is enabled.</p> <p>When this bit is reset, Receive Stopped Interrupt is disabled.</p> |
| 7 | R/W | RUE | 0 | <p>Receive Buffer Unavailable Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), Receive Buffer Unavailable Interrupt is enabled.</p> <p>When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.</p> |
| 6 | R/W | RIE | 0 | <p>Receive Interrupt Enable</p> <p>When this bit is set with Normal Interrupt Summary Enable (Interrupt Enable Register[16]), Receive Interrupt is enabled.</p> <p>When this bit is reset, Receive Interrupt is disabled.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 5 | R/W | UNE | 0 | <p>Underflow Interrupt Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), Transmit Underflow Interrupt is enabled.</p> <p>When this bit is reset, Underflow Interrupt is disabled.</p> |
| 4 | R/W | OVE | 0 | <p>Overflow Interrupt Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), Receive Overflow Interrupt is enabled.</p> <p>When this bit is reset, Overflow Interrupt is disabled</p> |
| 3 | R/W | TJE | 0 | <p>Transmit Jabber Timeout Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), Transmit Jabber Timeout Interrupt is enabled.</p> <p>When this bit is reset, Transmit Jabber Timeout Interrupt is disabled.</p> |
| 2 | R/W | TUE | 0 | <p>Transmit Buffer Unavailable Enable</p> <p>When this bit is set with Normal Interrupt Summary Enable (Interrupt Enable Register[16]), Transmit Buffer Unavailable Interrupt is enabled.</p> <p>When this bit is reset, Transmit Buffer Unavailable Interrupt is disabled.</p> |
| 1 | R/W | TSE | 0 | <p>Transmit Stopped Enable</p> <p>When this bit is set with Abnormal Interrupt Summary Enable (Interrupt Enable Register[15]), Transmission Stopped Interrupt is enabled.</p> <p>When this bit is reset, Transmission Stopped Interrupt is disabled.</p> |
| 0 | R/W | TIE | 0 | <p>Transmit Interrupt Enable</p> <p>When this bit is set with Normal Interrupt Summary Enable (Interrupt Enable Register[16]), Transmit Interrupt is enabled.</p> <p>When this bit is reset, Transmit Interrupt is disabled.</p> |

331 . Interrupt Enable Register Description

E-DMA Missed Frame and Buffer Overflow Counter Register

The DMA maintains two counters to track the number of missed frames during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames due to the host buffer being unavailable.

Bits[27:17] indicate missed frames due to buffer overflow conditions (MTL and MAC-Core) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

| Bit | Access | Function | POR Value | Description |
|-------|--------|---|-----------|---|
| 31:29 | R | Reserved | 0 | Reserved |
| 28:17 | R/W* | Appl ⁿ missed frames counter | 0 | Indicates the number of frames missed by the application. This counter is incremented each time the MTL asserts the sideband signal mtl_rxoverflow_o. The counter is cleared when this register is read with mci_be_i[2] at 1'b1. |
| 16:0 | R/W* | Host missed frames counter | 0 | Indicates the number of frames missed by the controller due to the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. |

*- Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and is automatically cleared to 1'b0 on a register read. A register write of 1'b0 has no effect on this field.

332 . Missed Frame and Buffer Overflow Counter Register Description

E-DMA Current Host Transmit Descriptor Register

| Bit | Access | Function | POR Value | Description |
|------|--------|--|-----------|--|
| 31:0 | R | Host Transmit Descriptor Address Pointer | 0 | Cleared on Reset. Pointer updated by DMA during operation. |

333 . Current Host Transmit Descriptor Register Description

E-DMA Current Host Receive Descriptor Register

| Bit | Access | Function | POR Value | Description |
|------|--------|---|-----------|--|
| 31:0 | R | Host Receive Descriptor Address Pointer | 0 | Cleared on Reset. Pointer updated by DMA during operation. |

334 . Current Host Receive Descriptor Register Description

E-DMA Current Host Transmit Buffer Address Register

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------------------|-----------|--|
| 31:0 | R | Host Transmit Buffer Address Pointer | 0 | Cleared on Reset. Pointer updated by DMA during operation. |

335 . Current Host Transmit Buffer Address Register Description

E-DMA Current Host Receive Buffer Address Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------------------|-----------|--|
| 31:0 | R | Host Receive Buffer Address Pointer | 0 | Cleared on Reset. Pointer updated by DMA during operation. |

336 . Current Host Receive Buffer Address Register Description

15.2.6 MAC Register Description

MAC Configuration Register

The MAC Configuration register establishes receive and transmit operating modes.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:24 | R | Reserved | 0 | Reserved |
| 23 | R/W | WD | 0 | Watchdog Disable When this bit is set, the MAC-Core disables the watchdog timer on the receiver, and can receive frames of up to 16,384 bytes. When this bit is reset, the MAC-Core allows no more than 2,048 bytes (10,240 if JE is set high) of the frame being received and cuts off any bytes received after that. |
| 22 | R/W | JD | 0 | Jabber Disable When this bit is set, the MAC-Core disables the jabber timer on the transmitter, and can transfer frames of up to 16,384 bytes. When this bit is reset, the MAC-Core cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission. |
| 21 | R | BE | 0 | Frame Burst Enable (Reserved) |
| 20 | R/W | JE | 0 | Jumbo Frame Enable When this bit is set, MAC-Core allows Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 19:17 | R/W | IFG | 0 | <p>Inter-Frame Gap</p> <p>These bits control the minimum IFG between frames during transmission.</p> <ul style="list-style-type: none"> • 000: 96 bit times • 001: 88 bit times • 010: 80 bit times • 111: 40 bit times <p>Note that in Half-Duplex mode, the minimum IFG can be configured for 64 bit times (IFG = 100) only. Lower values are not considered. In 1000-Mbps mode, the minimum IFG supported is 64 bit times (and above) in the MAC-CORE configuration and 80 bit times (and above) in other system configurations.</p> |
| 16 | R/W | DCRS | 0 | <p>Disable Carrier Sense During Transmission</p> <p>When set high, this bit makes the MAC transmitter ignore the (G)MII CRS signal during frame transmission in Half-Duplex mode. This request results in no errors generated due to Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors due to Carrier Sense and will even abort the transmissions.</p> |
| 15 | R | PS | 1 | <p>Port Select</p> <ul style="list-style-type: none"> • 1: RMII (10/100 Mbps) <p>This bit is Read Only with the appropriate value in the 10/100 Mbps-only (always 1)</p> |
| 14 | R | Reserved | 0 | <p>Speed</p> <p>Indicates the speed in Fast Ethernet (RMII) mode:</p> <ul style="list-style-type: none"> • 0: 10 Mbps |
| 13 | R/W | DO | 0 | <p>Disable Receive Own</p> <p>When this bit is set, the MAC-Core disables the reception of frames when the gmii_txen_o is asserted in Half-Duplex mode.</p> <p>When this bit is reset, the MAC-Core receives all packets that are given by the PHY while transmitting.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 12 | R/W | LM | 0 | <p>Loop-back Mode</p> <p>When this bit is set, the MAC-Core operates in loop-back mode at RMII. The RMII Receive clock input (<code>clk_rx_i</code>) is required for the loopback to work properly, as the Transmit clock is not looped-back internally.</p> |
| 11 | R/W | DM | 0 | <p>Duplex Mode</p> <p>When this bit is set, the MAC-Core operates in a Full-Duplex mode where it can transmit and receive simultaneously.</p> |
| 10 | R/W | IPC | 0 | <p>Checksum Offload</p> <p>When this bit is set, the MAC-Core calculates the 16-bit 1's complement of the 1's complement sum of all received Ethernet frame payloads (16-bit). It also checks whether the IPv4 Header checksum (assumed to be bytes 25-26 or 29-30 (VLANTagged) of Received Ethernet frame) is correct for the received frame and gives the status in the Receive Status Word. The MAC-Core also appends the 16-bit Checksum calculated for the payload of the IP header datagram (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application. When this bit is reset, then this function is disabled. This bit is reserved (Read Only) with default value) if IP Checksum Offload feature is not enabled during coreKit configuration.</p> |
| 9 | R/W | DR | 0 | <p>Disable Retry</p> <p>When this bit is set, the MAC-Core will attempt only 1 transmission. When a collision occurs on the MII, the MAC-Core will ignore the current frame transmission and report a Frame Abort with excessive collision error in the transmit frame status. When this bit is reset, the MAC-Core will attempt retries based on the settings of BL.</p> |
| 8 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 7 | R/W | ACS | 0 | <p>Automatic Pad/CRC Stripping</p> <p>When this bit is set, the MAC-Core strips the Pad/FCS field on incoming frames only if the length's field value is less than or equal to 1,500 bytes. All received frames with length field greater than or equal to 1,501 bytes are passed to the application without stripping the Pad/FCS field.</p> <p>When this bit is reset, the MAC-Core will pass all incoming frames to the Host unmodified.</p> |
| 6:5 | R/W | BL | 0 | <p>Back-Off Limit</p> <p>The Back-Off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) the MAC-Core waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only to Half-Duplex mode and is reserved (Read Only) in Full-Duplex-only configuration.</p> <ul style="list-style-type: none"> • 00: $k = \min(n, 10)$ • 01: $k = \min(n, 8)$ • 10: $k = \min(n, 4)$ • 11: $k = \min(n, 1)$, <p>where n = re-transmission attempt. The random integer r takes the value in the range $0 \leq r < 2k$</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 4 | R/W | DC | 0 | <p>Deferral Check</p> <p>When this bit is set, the deferral check function is enabled in the MAC-Core. The MAC-Core will issue a Frame Abort status, along with the excessive deferral error bit</p> <p>set in the transmit frame status when the transmit state machine is deferred for more than 24,288 bit times in 10/100-Mbps mode. If the Core is configured for 1000 Mbps operation, or if the Jumbo frame mode is enabled in 10/100-Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active CRS (carrier sense) signal on the RMII. Defer time is not cumulative. If the transmitter defers for 10,000 bit times, then transmits, collides, backs off, and then has to defer again after completion of back-off, the deferral timer resets to 0 and restarts.</p> <p>When this bit is reset, the deferral check function is disabled and the MAC-Core defers until the CRS signal goes inactive. This bit is applicable only in Half-Duplex mode</p> <p>and is reserved (Read Only) in Full-Duplex-only configuration.</p> |
| 3 | R/W | TE | 0 | <p>Transmitter Enable</p> <p>When this bit is set, the transmit state machine of the MAC-Core is enabled for transmission on the RMII. When this bit is reset, the MAC-Core transmit state machine is disabled after the completion of the transmission of the current frame, and will not transmit any further frames.</p> |
| 2 | R/W | RE | 0 | <p>Receiver Enable</p> <p>When this bit is set, the receiver state machine of the MAC-Core is enabled for receiving frames from the RMII. When this bit is reset, the MAC-Core receive state machine is disabled after the completion of the reception of the current frame, and will not receive any further frames from the RMII.</p> |
| 1:0 | R | Reserved | 0 | Reserved |

337 . MAC Configuration Register Description

MAC Frame Filter

The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering.

The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31 | R/W | RA | 0 | <p>Receive All</p> <p>When this bit is set, the MAC-Core Receiver module passes to the Application all frames received irrespective of whether they pass the address filter. The result of the SA/DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver module passes to the Application only those frames that pass the SA/DA address filter.</p> |
| 30:11 | R | Reserved | 0 | Reserved |
| 10 | R/W | HPF | 0 | <p>Hash or Perfect Filter</p> <p>When set, this bit configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by HMC or HUC bits.</p> <p>When low and if the HUC/HMC bit is set, the frame is passed only if it matches the Hash filter.</p> |
| 9 | R/W | SAF | 0 | <p>Source Address Filter Enable</p> <p>The MAC-Core compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SAMatch bit of RxStatus Word is set high. When this bit is set high and the SA filter fails, the MAC-Core drops the frame. When this bit is reset, then the MAC-Core forwards the received frame to the application and with the updated SA Match bit of the RxStatus depending on the SA address comparison.</p> |
| 8 | R/W | SAIF | 0 | <p>SA Inverse Filtering</p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers will be marked as failing the SA Address filter.</p> <p>When this bit is reset, frames whose SA does not match the SA registers will be marked as failing the SA Address filter.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 7:6 | R/W | PCF | 0 | <p>Pass Control Frames</p> <p>These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). Note that the processing of PAUSE control frames depends only on RFE of Flow Control Register[2].</p> <ul style="list-style-type: none"> • 0x: MAC-Core filters all control frames from reaching the application • 10: MAC-Core forwards all control frames to application even if they fail the Address Filter • 11: MAC-Core forwards control frames that pass the Address Filter. |
| 5 | R/W | DBF | 0 | <p>Disable Broadcast Frames</p> <p>When this bit is set, the AFM module filters all incoming broadcast frames.</p> <p>When this bit is reset, the AFM module passes all received broadcast frames.</p> |
| 4 | R/W | PM | 0 | <p>Pass All Multicast</p> <p>When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed.</p> <p>When reset, filtering of multicast frame depends on HMC bit.</p> |
| 3 | R/W | DAIF | 0 | <p>DA Inverse Filtering</p> <p>When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames.</p> <p>When reset, normal filtering of frames is performed.</p> |
| 2 | R/W | HMC | 0 | <p>Hash Multicast</p> <p>When set, MAC performs destination address filtering of received multicast frames according to the hash table.</p> <p>When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in DA registers.</p> |
| 1 | R/W | HUC | 0 | <p>Hash Unicast</p> <p>When set, MAC performs destination address filtering of unicast frames according to the hash table.</p> <p>When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 0 | R/W | PR | 0 | <p>Promiscuous Mode</p> <p>When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address.</p> <p>The SA/DA Filter Fails status bits of the Receive Status Word will always be cleared when PR is set.</p> |

338 . MAC Frame Filter Register Description

MAC Hash Table High Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:0 | R/W | HTH | 0 | <p>Hash Table High</p> <p>This field contains the upper 32 bits of Hash table.</p> |

339 . Hash Table High Register Description

MAC Hash Table Low Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:0 | R/W | HTL | 0 | <p>Hash Table Low</p> <p>This field contains the lower 32 bits of Hash table.</p> |

340 . Hash Table Low Register Description

MAC GMII Address Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:16 | R | Reserved | 0 | Reserved |
| 15:11 | R/W | PA | 0 | <p>Physical Layer Address</p> <p>This field tells which of the 32 possible PHY devices are being accessed.</p> |
| 10:6 | R/W | GR | 0 | <p>GMII Register</p> <p>These bits select the desired GMII register in the selected PHY device.</p> |
| 5 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|-------------|---------------|-----------|--|-----------|-----------|-----------|-----|------------|--------------|-----|-------------|--------------|-----|-----------|--------------|-----|-----------|--------------|-----|-------------|---------------|-----|-------------|---------------|---------|----------|----------|
| 4:2 | R/W | CR | 0 | <p>CSR Clock Range</p> <p>The CSR Clock Range selection determines the clk_csr_i frequency and is used to decide the frequency of the MDC clock:</p> <table border="1"> <thead> <tr> <th>Selection</th><th>clk_csr_i</th><th>MDC Clock</th></tr> </thead> <tbody> <tr><td>000</td><td>60-100 MHz</td><td>clk_csr_i/42</td></tr> <tr><td>001</td><td>100-150 MHz</td><td>clk_csr_i/62</td></tr> <tr><td>010</td><td>20-35 MHz</td><td>clk_csr_i/16</td></tr> <tr><td>011</td><td>35-60 MHz</td><td>clk_csr_i/26</td></tr> <tr><td>100</td><td>150-250 MHz</td><td>clk_csr_i/102</td></tr> <tr><td>101</td><td>250-300 MHz</td><td>clk_csr_i/122</td></tr> <tr><td>110,111</td><td>Reserved</td><td>Reserved</td></tr> </tbody> </table> | Selection | clk_csr_i | MDC Clock | 000 | 60-100 MHz | clk_csr_i/42 | 001 | 100-150 MHz | clk_csr_i/62 | 010 | 20-35 MHz | clk_csr_i/16 | 011 | 35-60 MHz | clk_csr_i/26 | 100 | 150-250 MHz | clk_csr_i/102 | 101 | 250-300 MHz | clk_csr_i/122 | 110,111 | Reserved | Reserved |
| Selection | clk_csr_i | MDC Clock | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | 60-100 MHz | clk_csr_i/42 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | 100-150 MHz | clk_csr_i/62 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | 20-35 MHz | clk_csr_i/16 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011 | 35-60 MHz | clk_csr_i/26 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | 150-250 MHz | clk_csr_i/102 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 101 | 250-300 MHz | clk_csr_i/122 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 110,111 | Reserved | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | R/W | GW | 0 | <p>GMII Write</p> <p>When set, this bit tells the PHY that this will be a Write operation using the GMII Data register. If this bit is not set, this will be a Read operation, placing the data in the GMII Data register.</p> | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | R/W* | GB | 0 | <p>GMII Busy</p> <p>This bit should read a logic 0 before writing to GMII Address Register and Status Register. This bit must also be set to 0 during a Write to GMII Address Register. During a PHY register access, this bit will be set to 1'b1 by the Application to indicate that a Read or Write access is in progress. GMII Data Register should be kept valid until this bit is cleared by the MAC-Core during a PHY Write operation.</p> <p>GMII Data Register is invalid until this bit is cleared by the MAC-Core during a PHY Read operation. GMII Address Register should not be written to until this bit is cleared.</p> | | | | | | | | | | | | | | | | | | | | | | | | |

* - Register field can be read by the application (Read), can be set to 1'b1 by the application with a register write of 1'b1 (Write Set), and is cleared to 1'b0 by the core (Self Clear). The application cannot clear this type of field, and a register write of 1'b0 to this bit has no effect on this field.

341 . GMII Address Register Description

MAC GMII Data Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:16 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 15:0 | R/W | GD | 0 | <p>GMII Data</p> <p>This contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.</p> |

342 . GMII Data Register Description

MAC Flow Control Register

The Flow Control register controls the generation and reception of the Control (Pause Command) frames by the MAC-Core's Flow control module. A Write to a register with the Busy bit set to '1' triggers the Flow Control block to generate a

Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set until the

control frame is transferred onto the cable. The Host must make sure that the Busy bit is cleared before writing to the register.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:16 | R/W | PT | 0 | <p>Pause Time</p> <p>This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bits is configured to be double synchronized to the RMII clock domain, then consecutive writes to this register should be performed only after at least 4 clock cycles in the destination clock domain.</p> |
| 15:8 | R | Reserved | 0 | Reserved |
| 7 | R/W | DZPQ | 0 | <p>Disable Zero-Quanta Pause</p> <p>When set, this bit disables the automatic generation of Zero-Quanta Pause Control frames on the deassertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i/ mti_flowctrl_i). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.</p> |
| 6 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | |
|-----------|-----------------------------|----------|-----------|--|-----------|-----------|----|---------------------------|----|----------------------------|----|-----------------------------|----|-----------------------------|
| 5:4 | R/W | PLT | 0 | <p>Pause Low Threshold</p> <p>This field configures the threshold of the PAUSE timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic re-transmission of PAUSE Frame. The threshold values should be always greater than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot-times after the first Pause-frame is transmitted.</p> <table border="1" data-bbox="817 842 1262 1044"> <thead> <tr> <th>Selection</th><th>Threshold</th></tr> </thead> <tbody> <tr> <td>00</td><td>Pause time – 4 slot times</td></tr> <tr> <td>01</td><td>Pause time – 28 slot times</td></tr> <tr> <td>10</td><td>Pause time – 144 slot times</td></tr> <tr> <td>11</td><td>Pause time – 256 slot times</td></tr> </tbody> </table> <p>Slot time is defined as time taken to transmit 512 bits (64 bytes) on the RMII interface.</p> | Selection | Threshold | 00 | Pause time – 4 slot times | 01 | Pause time – 28 slot times | 10 | Pause time – 144 slot times | 11 | Pause time – 256 slot times |
| Selection | Threshold | | | | | | | | | | | | | |
| 00 | Pause time – 4 slot times | | | | | | | | | | | | | |
| 01 | Pause time – 28 slot times | | | | | | | | | | | | | |
| 10 | Pause time – 144 slot times | | | | | | | | | | | | | |
| 11 | Pause time – 256 slot times | | | | | | | | | | | | | |
| 3 | R/W | UP | 0 | <p>Unicast Pause Frame Detect</p> <p>When this bit is set, the MAC-Core will detect the Pause frames with the station's unicast address specified in MAC Address0 High Register and MAC Address0 Low Register, in addition to the detecting Pause frames with the unique multicast address. When this bit is reset, the MAC-Core will detect only a Pause frame with the unique multicast address specified in the 802.3x standard.</p> | | | | | | | | | | |
| 2 | R/W | RFE | 0 | <p>Receive Flow Control Enable</p> <p>When this bit is set, the MAC-Core will decode the received Pause frame and disable its transmitter for a specified (Pause Time) time.</p> <p>When this bit is reset, the decode function of the Pause frame is disabled.</p> | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|-----|-----------------------------|----------|-----------|---|
| 1 | R/W | TFE | 0 | <p>Transmit Flow Control Enable</p> <p>In Full-Duplex mode, when this bit is set, the MAC-Core enables the flow control operation to transmit Pause frames.</p> <p>When this bit is reset, the flow control operation in the MAC-Core is disabled, and the MAC-Core will not transmit any Pause frames.</p> <p>In Half-Duplex mode, when this bit is set, the MAC-Core enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.</p> |
| 0 | R/W*(FCB) or R/W(BPA) | FCB/BPA | 0 | <p>Flow Control Busy/Backpressure Activate</p> <p>This bit initiates a Pause Control frame in Full-Duplex mode and activates the backpressure function in Half-Duplex mode if TFE bit is set.</p> <p>In Full-Duplex mode, this bit should be read as 1'b0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1'b1.</p> <p>During a transfer of the Control Frame, this bit will continue to be set to signify that a frame transmission is in progress.</p> <p>After the completion of Pause control frame transmission, the MAC-Core will reset this bit to 1'b0. The Flow Control register should not be written to until this bit is cleared.</p> <p>In Half-Duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the MAC-Core. During backpressure, when the MAC-Core receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically OR'ed with the mti_flowctrl_i input signal for the backpressure function. When the MAC-Core is configured to Full-Duplex mode, the BPA is automatically disabled.</p> |

*- Register field can be read by the application (Read), can be set to 1'b1 by the application with a register write of 1'b1 (Write Set), and is cleared to 1'b0 by the core (Self Clear).

The application cannot clear this type of field, and a register write of 1'b0 to this bit has no effect on this field.

343 . Flow Control Register Description

MAC VLAN Tag Register

The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following 2 bytes are compared with the VLAN tag;

if a match occurs, it sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1518 bytes to 1522 bytes.

If the VLAN Tag register is configured to be double-synchronized to the RMII clock domain, then consecutive writes to these register should be performed only after at least 4 clock cycles in the destination clock domain.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:16 | R | Reserved | 0 | Reserved |
| 15:0 | R/W | VL | 0 | VLAN Tag Identifier. This contains the 802.1Q VLAN Tag to identify the VLAN frames, and is used to compare with the 15th and 16th bytes of the receiving frames for VLAN frames. If the VL is all-zeros, then the MAC-Core does not check the 15th and 16th bytes for VLAN tag comparison. |

344 . VLAN Tag Register Description

MAC Version Register

The Version register's contents identify the version of the core. This register contains two bytes, one of which Synopsys uses to identify the core release number, and the other of which the user sets during coreKit configuration.

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 15:8 | R | User-defined version | 0x10 | User-defined version (configured with coreKit) |
| 7:0 | R | Synopsys-defined version | 0x33 | Synopsys-defined version(3.3) |

345 . Version Register Description

MAC Remote Wake-Up Frame Filter Register

The register wkupfmfilter_reg, address (028H), loads the Wake-up Frame Filter register. To load values in a Wake-up Frame Filter register, the entire register (wkupfmfilter_reg) must be written. The

wkupfmfilter_reg register is loaded by sequentially loading the eight register values in address (028) for wkupfmfilter_reg0, wkupfmfilter_reg1, ... wkupfmfilter_reg7, respectively. wkupfmfilter_reg is read in the same way.

| | | | | | | | | |
|-------------------|--------------------|------------------|-----------------|-------------------|------|------------------|------|------------------|
| wkupfmfilter_reg0 | Filter 0 Byte Mask | | | | | | | |
| wkupfmfilter_reg1 | Filter 1 Byte Mask | | | | | | | |
| wkupfmfilter_reg2 | Filter 2 Byte Mask | | | | | | | |
| wkupfmfilter_reg3 | Filter 3 Byte Mask | | | | | | | |
| wkupfmfilter_reg4 | RSVD | Filter 3 Command | RSVD | Filter 2 Command | RSVD | Filter 1 Command | RSVD | Filter 0 Command |
| wkupfmfilter_reg5 | Filter 3 Offset | Filter 2 Offset | Filter 1 Offset | Filter 0 Offset | | | | |
| wkupfmfilter_reg6 | Filter 1 CRC - 16 | | | Filter 0 CRC - 16 | | | | |
| wkupfmfilter_reg7 | Filter 3 CRC - 16 | | | Filter 2 CRC - 16 | | | | |

Filter i Byte Mask

This register defines which bytes of the frame are examined by filter i (0, 1, 2, and 3) in order to determine whether or not the frame is a wake-up frame. The MSB (thirty-first bit) must be zero. Bit j [30:0] is the Byte Mask. If bit j (byte number) of the Byte Mask is set, then Filter i Offset + j of the incoming frame is processed by the CRC block; otherwise Filter i Offset + j is ignored.

Filter i Command

This 4-bit command controls the filter i operation. Bit 3 specifies the address type, defining the pattern's destination address type. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame. Bit 2 and Bit 1 are reserved. Bit 0 is the enable for filter i; if Bit 0 is not set, filter i is disabled.

Filter i Offset

This register defines the offset (within the frame) from which the frames are examined by filter i. This 8-bit pattern-offset is the offset for the filter i first byte to examined. The minimum allowed is 12.

Filter i CRC-16

This register contains the CRC_16 value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

MAC PMT Control and Status Register

| Bit | Access | Function | POR Value | Description |
|-------|----------------|---|-----------|--|
| 31 | R/W* | Wake-Up Frame Filter Register Pointer Reset | 0 | When set, resets the Remote Wake-up Frame Filter register pointer to 3'b000. It is automatically cleared after 1 clock cycle. |
| 30:10 | R | Reserved | 0 | Reserved |
| 9 | R/W | Global Unicast | 0 | When set, enables any unicast packet filtered by the MAC-Core (DAF) address recognition to be a wake-up frame. |
| 8:7 | R | Reserved | 0 | Reserved |
| 6 | R ¹ | Wake-Up Frame Received | | When set, this bit indicates the power management event was generated due to reception of a wake-up frame. This bit is cleared by a Read into this register |
| 5 | R ¹ | Magic Packet Received | 0 | When set, this bit indicates the power management event was generated by the reception of a Magic Packet. This bit is cleared by a Read into this register. |
| 4:3 | R | Reserved | 0 | Reserved |
| 2 | R/W | Wake-Up Frame Enable | 0 | When set, enables generation of a power management event due to wakeup frame reception. |
| 1 | R/W | Magic Packet Enable | 0 | When set, enables generation of a power management event due to Magic Packet reception. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------|-----------|--|
| 0 | R/W* | Power Down | 0 | When set, all received frames will be dropped. This bit is cleared automatically when a magic packet or Wake-Up frame is received, and Power-Down mode is disabled. Frames received after this bit is cleared are forwarded to the application. This bit must only be set when either the Magic Packet Enable or Wake-Up Frame Enable bit is set high. |

*- Register field can be read by the application (Read), can be set to 1'b1 by the application with a register write of 1'b1 (Write Set), and is cleared to 1'b0 by the core (Self Clear). The application cannot clear this type of field, and a register write of 1'b0 to this bit has no effect on this field.

¹- Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and can be cleared to 1'b0 either by the core itself (Self Clear). A register write of 1'b1 to this bit has no effect on this field.

346 . PMT Control and Status Register Description

MAC Interrupt Status Register

The Interrupt Status register contents identify the events in the MAC-CORE that can generate interrupt.

Note that all the interrupt events are generated only when the corresponding optional feature is selected during coreKit configuration and enabled during operation.

Hence, these bits are reserved when the corresponding features is not present in the core.

| Bit | Access | Function | POR Value | Description |
|------|--------|---|-----------|---|
| 15:8 | R | Reserved | 0 | Reserved |
| 7 | R | MMC Receive Checksum Offload Interrupt Status | 0 | This bit is set high whenever an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. This bit is only valid when the optional MMC module and Checksum Offload Engine (Type 2) are selected during configuration. |
| 6 | R | MMC Transmit Interrupt Status | 0 | This bit is set high whenever an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. |
| 5 | R | MMC Receive Interrupt Status | 0 | This bit is set high whenever an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. |
| 4 | R | MMC Interrupt Status | 0 | This bit is set high whenever any of bits 7:5 is set high and cleared only when all of these bits are low. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|--|
| 3 | R | PMT Interrupt Status | 0 | This bit is set whenever a Magic packet or Wake-on-LAN frame is received in Power-Down mode (See bits 5 and 6 in the PMT Control and Status register) This bit is cleared when both bits[6:5] are cleared due to a read operation to the PMT Control and Status register. |
| 2 | R | Reserved | 0 | Reserved |
| 1 | R | Reserved | 0 | Reserved |
| 0 | R | Reserved | 0 | Reserved |

347 . Interrupt Status Register Description

MAC Interrupt Mask Register

The Interrupt Mask Register bits enables the user to mask the interrupt signal due to the corresponding event in the Interrupt Status Register.

The interrupt signal is sbd_intr_o in MAC-AHB and MAC-DMA configuration while the interrupt signal is mci_intr_o in the MAC-MTL and MAC-CORE configuration.

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------------------|-----------|--|
| 15:4 | R | Reserved | 0 | Reserved |
| 3 | R/W | PMT Interrupt Mask | 0 | This bit when set, will disable the assertion of the interrupt signal due to the setting of PMT Interrupt Status bit in Interrupt Status Register. |
| 2 | R/W | PCS AN Completion Interrupt Mask | 0 | This bit when set, will disable the assertion of the interrupt signal due to the setting of PCS Auto-negotiation complete bit in Interrupt Status Register caused due to the completion of Auto-negotiation event. |
| 1 | R/W | PCS Link Status Interrupt Mask | 0 | This bit when set, will disable the assertion of the interrupt signal due to the setting of PCS Link-status changed bit in Interrupt Status Register caused due to change in link-status event. |
| 0 | R/W | Reserved | 0 | Reserved |

348 . Interrupt Mask Register Description

MAC Address0 High Register

If the MAC address registers are configured to be double-synchronized to the RMII clock domains, then the synchronization is triggered only when Bits[31:24] (in Little-Endian mode) or Bits[7:0] (in Big-Endian mode) of the

MAC Address Low Register are written to. Please note that consecutive writes to this Address Low Register should be performed only after at least 4 clock cycles in the destination clock domain for proper synchronization updates.

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|-------------|
| 31 | R | MO | 0x1 | Always 1 |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 30:16 | R | Reserved | 0 | Reserved |
| 15:0 | R/W | A[47:32] | 0xFFFF | <p>MAC Address0 [47:32]</p> <p>This field contains the upper 16 bits (47:32) of the 6-byte first MAC address.</p> <p>This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.</p> |

349 . MAC Address0 High Register Description

MAC Address0 Low Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | R/W | A[31:0] | 0xFFFF_FFFF | <p>MAC Address0 [31:0]</p> <p>This field contains the lower 32 bits of the 6-byte first MAC address.</p> <p>This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.</p> |

350 . MAC Address0 Low Register Description

MAC Address1 High Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 31 | R/W | AE | 0 | <p>Address Enable</p> <p>When this bit is set, the Address filter module uses the second MAC address for perfect filtering. When reset, the address filter module will ignore the address for filtering.</p> |
| 30 | R/W | SA | 0 | <p>Source Address</p> <p>When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received frame.</p> <p>When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received frame.</p> |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 29:24 | R/W | MBC | 0 | <p>Mask Byte Control</p> <p>These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC core does not compare the corresponding byte of received DA/SA with the contents of Mac Address1 registers. Each bit controls the masking of the bytes as follows:</p> <ul style="list-style-type: none"> • Bit 29: Register18[15:8] • Bit 28: Register18[7:0] • Bit 27: Register19[31:24] ... • Bit 24: Register19[7:0] |
| 23:16 | R | Reserved | 0 | Reserved |
| 15:0 | R/W | A[47:32] | 0xFFFF | <p>MAC Address1 [47:32]</p> <p>This field contains the upper 16 bits (47:32) of the 6-byte second MAC address.</p> |

351 . MAC Address1 High Register Description

MAC Address1 Low Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | R/W | A[31:0] | 0xFFFF_FFFF | <p>MAC Address0 [31:0]</p> <p>This field contains the lower 32 bits of the 6-byte second MAC address. The content of this field is undefined until loaded by the Application after the initialization process.</p> |

352 . MAC Address1 Low Register Description

- The descriptions for registers MAC Address2 High Register to MAC Address4 High Register are the same as for the MAC Address1 High Register.
- The descriptions for registers MAC Address2 Low Register to MAC Address4 Low Register are the same as for the MAC Address1 Low Register.

15.2.7 MMC Register Description

MMC Control Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:4 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|---|
| 3 | R/W | MMC Counter Freeze | 0 | When set, this bit freezes all the MMC counters to their current value. (None of the MMC counters are updated due to any transmitted or received frame until this bit is reset to 0. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.) |
| 2 | R/W | Reset on Read | 0 | When set, the MMC counters will be reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read. |
| 1 | R/W | Counter Stop Rollover | 0 | When set, counter after reaching maximum value will not roll over to zero. |
| 0 | R/W* | Counters Reset | 0 | When set, all counters will be reset. This bit will be cleared automatically after 1 clock cycle |

*- Register field can be read and written by the application (Read and Write), and is cleared to 1'b0 by the core (Self Clear).

353 . MMC Control Register Description

MMC Receive Interrupt Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------|-----------|---|
| 31:22 | R | Reserved | 0 | Reserved |
| 21 | R* | rxfifooverflow_intr | 0 | The bit is set when the rxfifooverflow counter reaches half the maximum value. |
| 20:6 | R | Reserved | 0 | Reserved |
| 5 | R* | rxcrcerror_intr | 0 | The bit is set when the rxrcerror counter reaches half the maximum value. |
| 4:1 | R | Reserved | 0 | Reserved |
| 0 | R* | rxframecount_gb_int | 0 | The bit is set when the rxframecount_gb counter reaches half the maximum value. |

*- Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and is automatically cleared to 1'b0 on a register read. A register write of 1'b0 has no effect on this field.

354 . MMC Receive Interrupt Register Description

MMC Transmit Interrupt Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------|-----------|--|
| 31:22 | R | Reserved | 0 | Reserved |
| 21 | R* | txframecount_g_intr | 0 | The bit is set when the txframecount_g counter reaches half the maximum value. |
| 20:17 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------|-----------|--|
| 16 | R* | txdeferred_intr | 0 | The bit is set when the txdeferred counter reaches half the maximum value. |
| 15 | R | Reserved | 0 | Reserved |
| 14 | R* | txsinglecol_g_intr | 0 | The bit is set when the txsinglecol_g counter reaches half the maximum value. |
| 13 | R* | txunderflowerror_intr | 0 | The bit is set when the txunderflowerror counter reaches half the maximum value. |
| 12:0 | R | Reserved | 0 | Reserved |

*- Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal event (Self Set), and is automatically cleared to 1'b0 on a register read. A register write of 1'b0 has no effect on this field.

355 . MMC Transmit Interrupt Register Description

MMC Receive Interrupt Mask Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|---|
| 31:22 | R | Reserved | 0 | Reserved |
| 21 | R/W | rxfifooverflow_mask | 0 | Setting this bit masks the interrupt when the rxfifooverflow counter reaches half the maximum value. |
| 20:6 | R/W | Reserved | 0 | Reserved |
| 5 | R/W | rxrcerror_mask | 0 | Setting this bit masks the interrupt when the rxrcerror counter reaches half the maximum value. |
| 4:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | rxframecount_gb_m ask | 0 | Setting this bit masks the interrupt when the rxframecount_gb counter reaches half the maximum value. |

356 . MMC Receive Interrupt Mask Register Description

MMC Transmit Interrupt Mask Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------|-----------|--|
| 31:22 | R | Reserved | 0 | Reserved |
| 21 | R/W | txframecount_g_mask | 0 | Setting this bit masks the interrupt when the txframecount_g counter reaches half the maximum value. |
| 20:17 | R | Reserved | 0 | Reserved |
| 16 | R/W | txdeferred_mask | 0 | Setting this bit masks the interrupt when the txdeferred counter reaches half the maximum value. |
| 15 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------|-----------|--|
| 14 | R/W | txsinglecol_g_mask | 0 | Setting this bit masks the interrupt when the txsinglecol_g counter reaches half the maximum value. |
| 13 | R/W | txunderflowerror_mas0k | 0 | Setting this bit masks the interrupt when the txunderflowerror counter reaches half the maximum value. |
| 12:0 | R | Reserved | 0 | Reserved |

357 . MMC Transmit Interrupt Mask Register Description

Ethernet Block Diagram

15.3 High Speed SPI Slave

15.3.1 General Description

The High Speed SPI Slave Interface is a full duplex serial host interface, which supports 8-bit and 32-bit data granularity. It also supports gated mode of SPI clock and both the low, high and the very high frequency modes. In case of low frequency host, the data is driven on the falling edge and sampled on the rising edge and hence, it should be ensured that a valid data is present on the bus at the immediate rising edge after the SPI chip select is driven low. For high frequency transmission the data is driven as well as sampled on rising edge.

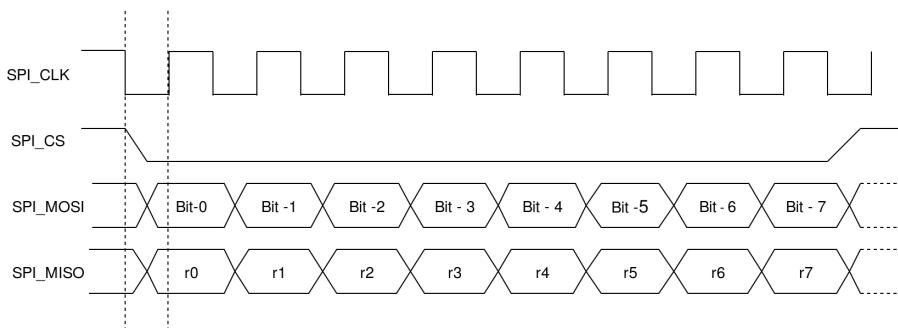
15.3.2 Features

- 4-pin serial interface
- Supports 8-bit and 32-bit data granularity
- Supports frequencies up to 120MHz
- SPI clock can be at the max 4 times higher than AHB clock
- Supports DMA flow control signals
- Supports AHB interface for accessing data from SOC
- Supports system soft reset from Host

15.3.3 Functional Description

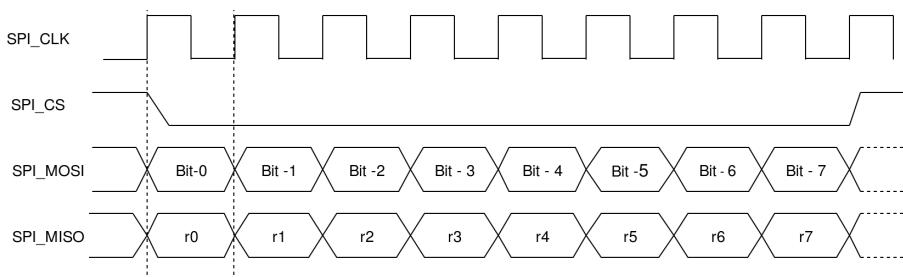
The SPI slave interface is invoked by the Host using a specific pattern in first 32-bits of each transfer. This 32-bit pattern is divided into four 8-bit patterns and each 8-bit set is termed as one command.

Transmission up to 25MHz are termed as low-speed transmissions and during these transmissions the data is driven on the falling edge of the clock and read on the rising edge of the clock.



SPI Low Speed Transmission

Transmissions above 25 MHz are high-speed transmissions and during these transmissions the data is driven on the rising edge of the clock and read on following rising edge of the clock.



SPI High Speed Transmission

However initialization is done only in low-speed mode. To enable the SPI Interface for a high-speed transmission, the Host has to first initialize the SPI Interface and then perform a SPI Slave Internal Register Write operation to SPI_MODE register to set the SPI_MODE [SPI_OP_MODE] bit. The register write is also performed in the low-speed mode. Only if the SPI_MODE [SPI_OP_MODE] bit is set, the host can do a high-speed transfer.

SPI Commands

The SPI interface is programmed to perform a certain transfer using four commands and a 32-bit address. The Host transfers all the Commands and Addresses with 8-bit granularity. At the end of all Commands and Address, the Host should be reconfigured to transfer data with 8-bit or 32-bit granularity depending on the commands issued. The Slave responds to all the commands with a certain response pattern. For transferring the response, start-token and data, the SPI interface follows the same protocol as followed by the host.
The four commands C1, C2, C3, C4 indicate to the SPI interface all the aspects of the transfer.

8 . SPI Command Description

The command description is as follows:

| | | |
|----|-------|--|
| C1 | [7:6] | Command Type "00"- Initialization Command "01"- Read/Write Command "10", "11"- Reserved for future use |
| | 5 | Read/Write '0'- Read Command '1'- Write Command |
| | 4 | Internal/Bus Access '0'- SPI Slave Internal Access '1'- AHB Bus Access |
| | 3 | Master/Slave Access '0'- AHB Master Access '1'- AHB Slave Access |
| | 2 | 2-bit or 16-bit length for the transfer '0'- 2-bit length for the transfer '1'- 16-bit length for the transfer |
| | 1:0 | 2-bit length (in terms of bytes) for the transfer (if bit 2 is cleared) "00"- 4 Bytes length "01"- 1 Byte length "10"- 2 Bytes length "11"- 3 Bytes length |

| | | |
|----|-----|---|
| C2 | 7:6 | Data granularity. Indicates the granularity of the write/read data. Note: The master commands and addresses will always be 8-bit irrespective of this value. "00"- 8-bit granularity "01"- 32-bit granularity "10", "11"- Reserved for future use |
| | 5:0 | Internal Address or miscellaneous Info. This carries the SPI Slave's internal address if bit 4 for Command C1 is cleared. Else it carries miscellaneous information for the Godavari chip. |
| C3 | 7:0 | Length (15:8) MSB of the transfer's length (which is in terms of bytes) in case bit 2 of C1 is set. This command is skipped if bit 2 of C1 is cleared i.e. if 2-bit length is selected |
| C4 | 7:0 | Length (7:0) LSB of the transfer's length (which is in terms of bytes) in case bit 2 of C1 is set. This command is skipped if bit 2 of C1 is cleared. |

358 . SPI Command Description

Depending on the above four commands the SPI interface can be initialized or made to do a read or write operation to an AHB master or an AHB slave. To all these commands the SPI interface responds with set of unique responses.

Slave Response to Commands

The SPI slave gives simultaneous responses to the SPI Master's requests. These are as follows:

- An 8-bit Success/Failure response at the end of receiving the first 8-bits of the Command. This response is continuously driven in the case of a Write Request starting from the time when the first 8-bits are received. This response is driven with 8-bit granularity during the Command and Address phase and then is switched to 8-bit or 32-bit granularity during the Data phase, according to the command issued
 - Success: 0x58
 - Failure: 0x52
- An 8-bit or 32-bit start token is transmitted once the four commands indicating a read request are received and the slave is ready to transmit data. The start token is immediately followed by the read-data
 - Start Token: 0x55
- An 8-bit busy response is driven by the SPI interface in case a new transaction is initiated by the host while the previous transaction is still pending from the system side. In this case the host has to retry the commands.
 - Busy Response: 0x54

Initialization

The Initialization Command is given to the Slave to initialize the SPI interface. The SPI interface remains non functional to any command before initialization and responds only after successful initialization. Initialization should be done only once after the power-on. The SPI Slave treats any subsequent initialization commands before the reset, as errors.

For the initialization command, the host drives C1 command, followed by an 8-bit miscellaneous data. Bits 7:6 of C1 are cleared and 0x15 is driven on bits 5:0. Status response from the SPI Interface is driven during the transmission of the miscellaneous data i.e. after the transfer of 8-bits of command C1.

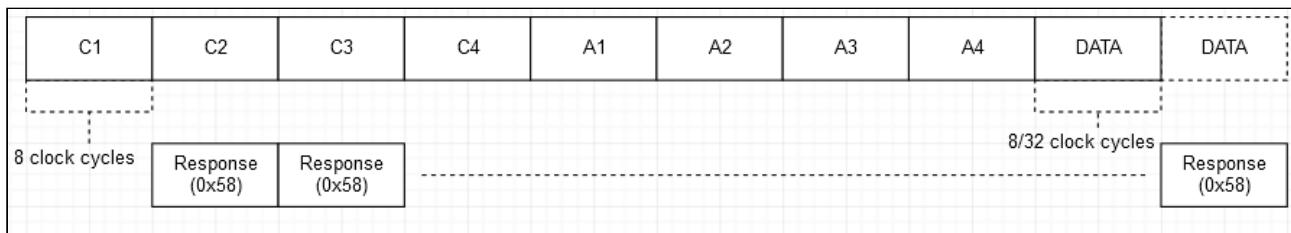
9 . SPI Slave Initialization

Busy Condition

The busy condition arises when the a new transaction is initiated by the host while the previous transaction is still pending from the system side. The SPI Interface indicates this to the Host with a Busy Response 0x54 to the command C1. In such case the Host should re-transmit the commands till it receives a success response 0x58 from the SPI Slave Interface.

AHB Master Write

During a write operation to the AHB master the host first transfers the four commands, then transfers a valid 32-bit address to which write has to be done and then the data. The slave responds to all the commands and data with appropriate response.



10 . SPI SLAVE AHB Master Write

AHB Slave Write

This is similar to the AHB master write except that bit 3 of C1 is set and the Address phase (A1, A2, A3& A4) is skipped.

SPI Interface Internal Registers Write

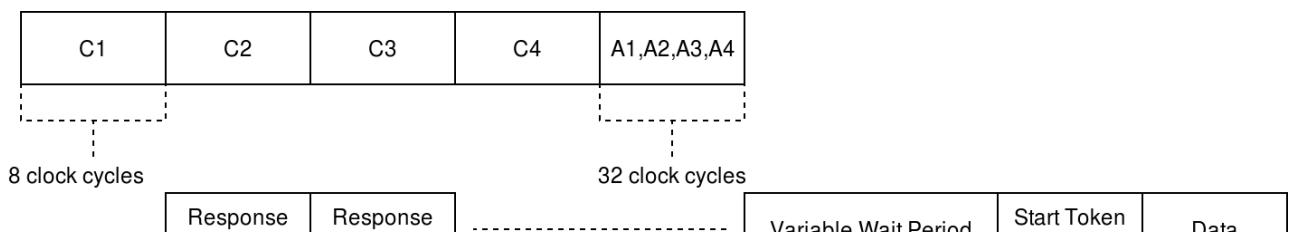
This is similar to the AHB master write except that bit 4 of C1 is cleared and A1, A2, A3 and A4 are skipped. The Valid Data phase starts immediately after C4 is transferred.

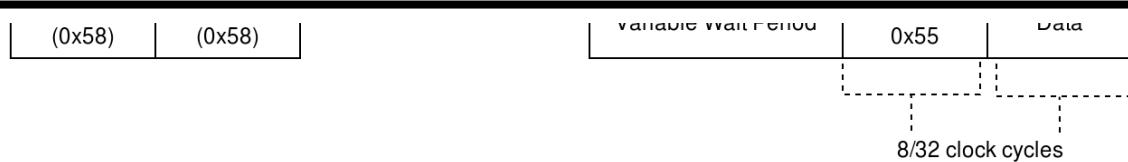
AHB Master/Slave, Internal register write with 2-bit length

This, too, is similar to the AHB master write except that bit 2 of C1 is cleared and C3 and C4 are both skipped. If it's an internal register or AHB Slave access then A1, A2, A3 and A4 are also skipped after C2. The number of bytes to be transferred is inferred from bits [1:0] of C1.

AHB Master Read

During the AHB master read operation, first the four commands are transmitted, then the 32-bit address and then after a variable wait period a start token is transmitted. The start token is followed by valid data. The start token indicates to the host the beginning of valid data.





SPI Slave AHB Master Read

AHB Slave Read

This is similar to the AHB Master Read except that bit 3 of C1 is set and the Address phase (A1, A2, A3& A4) is skipped.

SPI Interface's Internal Registers Read

This is similar to the AHB Master Read except that bit 4 of C1 is cleared and A1, A2, A3 and A4 are skipped. The Variable Wait Period starts immediately after C4 is transferred.

AHB Master/AHB Slave/ Internal register write with 2-bit length

This too is similar to the AHB Master Read except that bit 2 of C1 is cleared and C3 and C4 are both skipped. If it's an internal register or AHB Slave access then A1, A2, A3 and A4 are also skipped after C2. The number of bytes to be transferred is inferred from bits 1:0 of C1.

15.3.4 Register Summary

Base Address: 0x2020_0000

| Register Name | Offset | Description | Access by MCU / SOC | Access by Host |
|---------------------|--------|-------------------------------------|---------------------|----------------|
| SPI_HOST_INTR | 0x0 | SPI Host Interrupt Register | Yes | Yes |
| SPI_RFIFO_START | 0x2 | SPI RFIFO Start Level Register | Yes | Yes |
| SPI_RFIFO_AFULLLEV | 0x4 | SPI RFIFO Almost Full Register | Yes | Yes |
| SPI_RFIFO_AEMPTYLEV | 0x6 | SPI WFIFO Almost Empty Register | Yes | Yes |
| SPI_MODE | 0x8 | SPI Mode Register | Yes | Yes |
| SPI_INTR_STATUS | 0xA | SPI Interrupt Status/Clear Register | Yes | Yes |
| SPI_INTR_EN | 0xC | SPI Interrupt Enable Register | Yes | Yes |
| SPI_INTR_MASK | 0xE | Interrupt Mask Register | Yes | Yes |
| SPI_INTR_UNMASK | 0x10 | SPI/MMIO Interrupt Unmask Register | Yes | Yes |
| SPI_LENGTH | 0x12 | SPI Length Register | Yes | Yes |
| SPI_COMMAND | 0x14 | SPI Command Register | Yes | Yes |
| SPI_DEV_ID | 0x16 | SPI Device ID Register | Yes | Yes |
| SPI_VER_NO | 0x18 | SPI Version Number Register | Yes | Yes |

| Register Name | Offset | Description | Access by MCU / SOC | Access by Host |
|--------------------------|---------------|-----------------------------------|---------------------|----------------|
| SPI_STATUS | 0x1A | SPI Status Register | Yes | Yes |
| SPI_BUS_CONTROLLER_STATE | 0x1C | SPI Bus Controller State Register | Yes | Yes |
| SPI_CONFIG_1 | 0x20 | SPI Configuration 1 Register | Yes | Yes |
| SPI_CONFIG_2 | 0x22 | SPI Configuration 2 Register | Yes | Yes |
| SPI_CONFIG_3 | 0x24 | SPI Configuration 3 Register | Yes | Yes |
| SPI_CONFIG_4 | 0x26 | SPI Configuration 4 Register | Yes | Yes |
| SPI_CONFIG_5 | 0x28 | SPI Configuration 5 Register | Yes | Yes |
| SPI_CONFIG_6 | 0x2A | SPI Configuration 6 Register | Yes | Yes |
| SPI_CONFIG_7 | 0x2C | SPI Configuration 7 Register | Yes | Yes |
| SPI_CONFIG_8 | 0x2E | SPI Configuration 8 Register | Yes | Yes |
| SPI_SYS_RESET_REQ | 0x3E | SPI Reset Register | No | Yes |
| SPI_WAKE_UP | 0x3F | SPI Wake up Register | No | Yes |
| SPI_RFIFO_DATA | 0x380 – 0x3BF | SPI RFIFO Data Register | Yes | Yes |
| SPI_WFIFO_DATA | 0x3C0 – 0x3FF | SPI WFIFO Data Register | Yes | Yes |

359 . Register Summary Table

15.3.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write

SPI Host Interrupt Register

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------|-----------|---|
| 15:8 | R | Reserved | 0x0 | Reserved |
| 7:0 | R/W | SPI_HOST_INTR | 0x0 | These bits indicate the interrupt vector value coming from system side. |

360 . SPI_HOST_INTR Description

SPI RFIFO Start Level Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------|-----------|--|
| 7:0 | R/W | SPI_RFIFO_ST | 0x10 | These bits are used to program the minimum FIFO occupancy level before which data will not be sent out |

361 . SPI_RFIFO_ST Description

SPI RFIFO Almost Full Register

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------|-----------|---|
| 15:9 | R | Reserved | 0x0 | Reserved |
| 8:0 | R/W | SPI_RFIFO_AFULLLEV | 0x40 | These bits are used to program the FIFO occupancy level to trigger the Almost Full indication |

362 . SPI_RFIFO_AFULLLEV Description

SPI WFIFO Almost Empty Register

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|---|
| 15:9 | R | Reserved | 0x0 | Reserved |
| 8:0 | R/W | SPI_RFIFO_AEMPTYLEV | 0x40 | These bits are used to program the occupancy level to trigger the Almost Empty indication |

363 . SPI_RFIFO_AEMPTYLEV Description

SPI Mode Register

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|---|
| 15:3 | R | Reserved | 0x0 | Reserved for future use |
| 2 | R/W | VHS_EN | 0x0 | This bit is used to enable Very high speed mode (120Mhz) '0' – Doesn't enable '1' – Enable |
| 1 | R/W | SPI_FIX_EN | 0x1 | This bit is used to enable the fix made for bus_ctrl_busy being asserted when success_state is being asserted getting deasserted when FSM has decided to move to BUSY_STATE or not '0' – Doesn't enable the fix '1' – Enables the fix |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------|-----------|---|
| 0 | R/W | SPI_OP_MODE | 0x0 | This bit is used to program the mode of working of SPI Interface '0'- Low speed mode '1'- High speed mode |

364 . SPI_MODE Description

SPI Interrupt Status/Clear Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------|-----------|--|
| 15:3 | R | Reserved | 0x0 | Reserved for future use |
| 2 | R/W | SPI_CS_DEASSERT | 0x0 | SPI_CS deasserted without transferring the correct number of bits. This can happen because of glitches in clock or because of a wrong de-assertion of CS. |
| 1 | R/W | SPI_RD_REQ | 0b0 | Read request received |
| 0 | R/W | SPI_WR_REQ | 0b0 | Write request received |

365 . SPI_INTR_STATUS Description

SPI Interrupt Enable Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------|-----------|---|
| 15:3 | R | Reserved | 0x0 | Reserved for future used |
| 2 | R/W | SPI_CS_DEASSERT_IN T_EN | 0x0 | This bit is used to enable the interrupt due to wrong de-assertion of CS. |
| 1 | R/W | SPI_RD_INTR_EN | 0x0 | This bit is used to enable the read interrupt |
| 0 | R/W | SPI_WR_INTR_EN | 0x0 | This bit is used to enable the write interrupt |

366 . SPI_INTR_EN Description

Interrupt Mask Register

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 15:3 | R | Reserved | 0x0 | Reserved for future used |
| 2 | R/W | SPI_CS_DEASSERT_IN T_MSK | 0x0 | This bit is used to mask the CS de-assertion interrupt '1' – Mask interrupt |
| 1 | R/W | SPI_RD_INTR_MSK | 0x0 | This bit is used to mask the write interrupt '1'- Mask interrupt |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|---|
| 0 | R/W | SPI_WR_INTR_MSK | 0x0 | This bit is used to mask the read interrupt '1' - Mask interrupt |

367 . SPI_INTR_MASK Description

SPI / MMIO Interrupt Unmask Register

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------|-----------|--|
| 15:3 | R | Reserved | 0x0 | Reserved for future used |
| 2 | R/W | SPI_CS_DEASSERT_INT_UNMSK | 0x0 | This bit is used to unmask the CS de-assertion interrupt '1' - Unmask interrupt |
| 1 | R/W | SPI_RD_INTR_UNMSK | 0x0 | This bit is used to unmask the write interrupt '1' - Unmask interrupt |
| 0 | R/W | SPI_WR_INTR_UNMSK | 0x0 | This bit is used to mask the read interrupt '1' - Unmask interrupt |

368 . SPI_INTR_UNMASK Description

SPI Length Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 15:0 | R | SPI_LEN | 0x0 | These bits indicate the length of the transfer as transmitted in the Commands C3 and C4 |

369 . SPI_LENGTH Description

SPI Command Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 15:8 | R | SPI_C2 | 0x0 | These bits store the received command C2 |
| 7:0 | R | SPI_C1 | 0x0 | These bits store the received command C1 |

370 . SPI_COMMAND Description

SPI Device ID Register

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|--|
| 15:0 | R | SPI_DEV_ID | 0x9116 | These bits store the Device ID information |

371 . SPI_DEV_ID Description

SPI Version Number Register

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|-------------------------------------|
| 15:8 | R | Reserved | 0x00 | Reserved for future use. |
| 7:0 | R | SPI_VER_NO | 0x02 | These bits store the version number |

372 .SPI_VER_NO Description

SPI Status Register

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------|-----------|--|
| 15:4 | R | Reserved | 0x0 | Reserved for future use |
| 3 | R | SPI_WFIFO_AEMPTY | 0x1 | This bit indicates if write FIFO is almost empty '0'- Write FIFO is almost empty '1'- Write FIFO is not almost empty |
| 2 | R | SPI_WFIFO_EMPTY | 0x0 | This bit indicates if write FIFO is empty '0'- Write FIFO is empty '1'- Write FIFO is not empty |
| 1 | R | SPI_RFIFO_AFULL | 0x0 | This bit indicates if the read FIFO is almost full. '1'- Almost full '0'- Not almost full |
| 0 | R | SPI_RFIFO_FULL | 0x0 | This bit indicates if the read FIFO is almost full. '1'- Almost full '0'- Not almost full |

373 .SPI_STATUS Description

SPI Bus Controller State Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--------------------------|
| 15:14 | R | Reserved | 0x0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 13:0 | R | SPI_BC | 0x0 | <p>These bits indicate the Bus Controller FSM state. (One-Hot Coding)</p> <ul style="list-style-type: none"> "0th bit" - BC_IDLE "1st bit" - BC_EN "2nd bit" - BC_CTRL_BUSY "3rd bit" - BC_INT_REG_RD "4th bit" - BC_INT_REG_FIFO_WR "5th bit" - BC_INT_REG_WR_WAIT "6th bit" - BC_INT_REG_FIFO_RD "7th bit" - BC_INT_REG_WR "8th bit" - BC_AHB_MASTER "9th bit" - BC_AHB_MASTER_WAIT "10th bit" - BC_AHB_SLAVE_WR_LEN "11th bit" - BC_AHB_SLAVE_WR_CMD "12th bit" - BC_AHB_SLAVE_WR_INTR "13th bit" - BC_AHB_SLAVE_WAIT |

374 . SPI_BC_STATE Description

SPI_CONFIG_n

SPI_CONFIG_1 to SPI_CONFIG_8 are general purpose registers used by firmware and Host Driver.

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-----------|--|
| 31:0 | R/W | General purpose bits | 0x0 | These bits are used by firmware and Host Driver. |

375 . SPI_CONFIG_n Description

SPI SYS Reset Req Register

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------|-----------|---|
| 15:1 | R | Reserved | 0x0 | Reserved for future use |
| 0 | R/W | SPI_SYS_RESETREQ | 0x0 | <p>When set generates system reset request to reset controller. This gets reset once, reset controller generates reset.</p> <p>Host shouldn't reset this bit. With this reset request, reset controller generates non por reset</p> |

376 . SPI_SYS_RESET_REQ Description

SPI Wakeup Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------------------|
| 15:1 | R | Reserved | 0x0 | Reserved for future use |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------|-----------|---|
| 0 | R/W | SPI_WAKEUP | 0x0 | Wakeup Interrupt - Interrupt for waking up the system from Deep Sleep |

377 . SPI_WAKE_UP Description

SPI RFIFO Data Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|--|
| 31:0 | W | SPI_RFIFO | 0x0 | These bits are used to write, the data to be sent to the host. |

378 . SPI_RFIFO_DATA Description

SPI WFIFO Data Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|--|
| 31:0 | R | SPI_WFIFO | 0x0 | These bits store the data received from the host |

379 . SPI_WFIFO_DATA Description

15.4 SDIO Slave

15.4.1 General Description

The Secure Digital I/O (SDIO) Slave module implements the functionality of the SDIO card based on the SDIO specifications version 2.0, released by SD Association. During the normal initialization and interrogation of the card by the host, the card identifies itself as an SDIO card. The host software then obtains the card information in a tuple (linked list) format and determines if that card's I/O function(s) are acceptable to activate. This decision is based on such parameters as power requirements or the availability of appropriate software drivers. If the card is acceptable, it is allowed to power up fully and start the I/O function(s) built into it. The salient SDIO slave features are described in the following section.

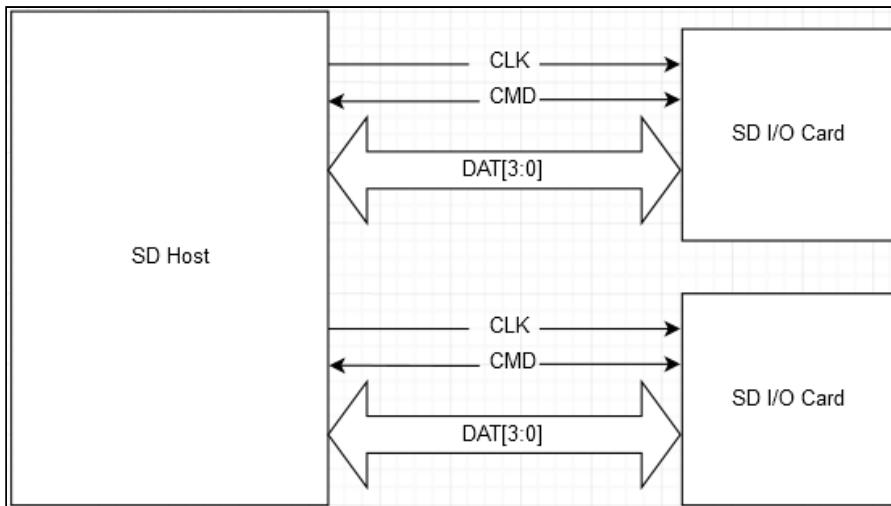
15.4.2 Features

- Full throughput with SDIO 1.2 as well as with SDIO 2.0
- Supports up to 50MHz
- Supports full-speed and high speed modes
- Supports SD-1 bit, SD-4 bit and SPI modes
- Supports up to five functions
- Supports interrupt for both SD and SPI modes
- Supports single as well as multiple block transfers for CMD53 access
- Supports CMD52 while CMD53 data transfer is in progress
- Supports CMD52 Abort
- Supports Read Wait
- Supports co-existence with SD-Mem card which is present in the same SD slot
- Does not support Suspend/Resume
- Provides master and slave interfaces on system side AHB Bus
- Support CIS memory configuration during boot up.
- Supports system soft reset from Host

⚠ There is a constraint on minimum SoC clock relative to SDIO clock. SoC clock has to be minimum half of SDIO clock. This constraint is because of the synchronization mechanism used between SoC clock domain and SDIO clock domain.

15.4.3 Functional Description

The SDIO bus has a single master (Host), multiple slaves (I/O cards), synchronous star topology (refer to Figure 1). Clock and power signals are common to all cards. Command (CMD) and data (DAT0 - DAT3) signals are dedicated to each card providing continues point to point connection to all the cards. During initialization process commands are sent to each card individually, allowing the application to detect the cards and assign logical addresses to the physical slots. Data is always sent (received) to (from) each card individually. Addressing information is provided in the command packet. SD bus allows dynamic configuration of the number of data lines. After power up, by default, the SDIO Card will use only DAT0 for data transfer. After initialization the host can change the bus width (number of active data lines).



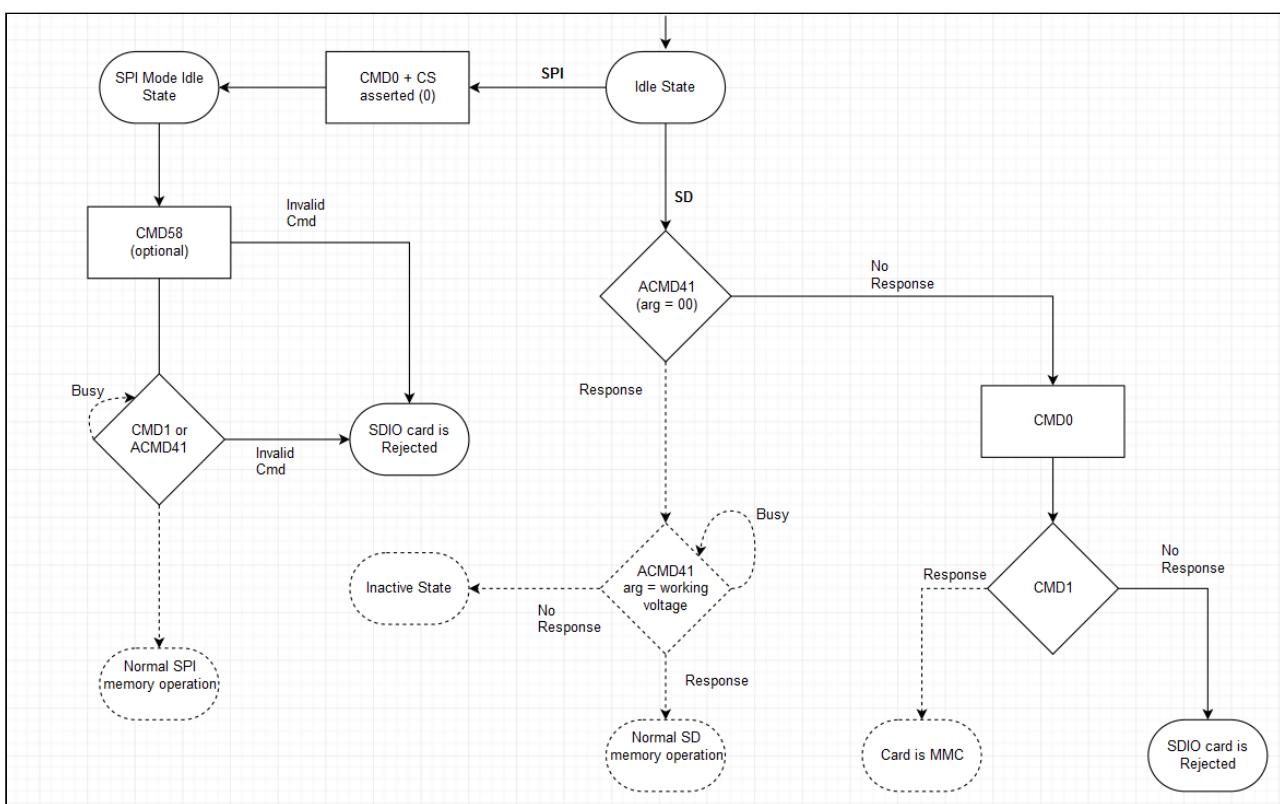
11 . SDIO Host connection to two 4-bit SDIO cards

The figure below illustrates the block diagram of the SDIO slave module. It contains Command and data control logic, WFIFO, RFIFO, Card Information Structure(CIS) registers, Card Common Control Registers(CCCR), Function Basic Registers (FBR), AHB master and slave interfaces.

12 . SDIO Slave Block Diagram

SDIO Card Initialization

An SDIO card shall not cause non-I/O aware hosts to fail when inserted. In order to prevent operation of I/O functions in non-I/O aware hosts, a change to the SD card identification mode flowchart is needed. A new command (IO_SEND_OP_COND, CMD5) is added to replace the ACMD41 for SDIO initialization by I/O aware hosts. After reset or power-up, all I/O functions on the card are disabled and the I/O portion of the card shall not execute any operation except CMD5 or CMD0 with CS=low. If there is SD memory installed on the card (also called a combo card), that memory shall respond normally to all normal mandatory memory commands. An I/O only card shall not respond to the ACMD41 and thus appear initially as an MMC card. The I/O only card shall also not respond to the CMD1 used to initialize the MMC cards and appear as a non-responsive card. The host then gives up and disables this card. Thus, the non-aware host receives no response from an I/O only card and force it to the inactive state. The operation of an I/O card with a non-I/O aware host is shown in Figure 3.



13 . SDIO response to non-I/O aware initialization

An SDIO aware host sends CMD5 prior to the CMD55/ACMD41 pair, and thus would receive a valid OCR in the R4 response to CMD5 and continue to initialize the card. Figure 4 shows the operation of an SDIO aware host operating in the SD modes and Figure 5 shows the same operation for a host that operates in the SPI mode. If the I/O portion of a card has received no CMD5, the I/O section remains inactive and shall not respond to any command except CMD5. A combo card stays in the memory-only mode. If no memory is installed on the card (i.e. an I/O only card in a non-SDIO aware host) the card would not respond to any memory command. This satisfies the condition where a user uses some I/O function on the card.

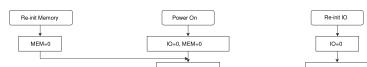
The card is then removed and inserted into a non-SDIO aware host. That host would not enable the I/O function (no CMD5) so would appear to the player as a memory-only card. If the host were I/O aware, it would send the CMD5 to the card and the card would respond with R4. The host reads that R4 value and knows the number of available I/O functions and about the existence of any SD memory.

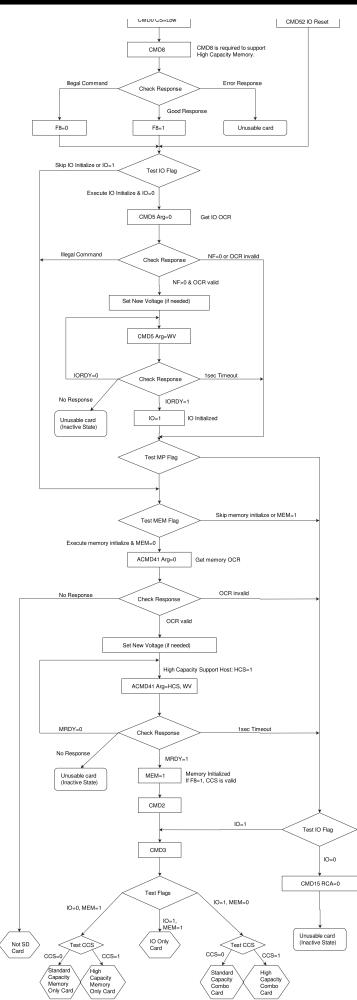
After the host has initialized the I/O portion of the card, it then reads the Common Information Area (CIA) of the card. This is done by issuing a read command, starting with the byte at address 0x00, of I/O function 0.

The CIA contains the Card Common Control Registers (CCCR) and the Function Basic Registers (FBR). Also included in the CIA are pointers to the card's common Card Information Structure (CIS) and each individual function's CIS. The CIS includes information on power, function, manufacturer and other things the host needs to determine if the I/O function(s) is appropriate to power-up. If the host determines that the card should be activated, a register in the CCCR area enables the card and each individual function. At this time, all functions of the I/O card are fully available. In addition, the host can control the power consumption and enable/disable interrupts on a function-by-function basis.

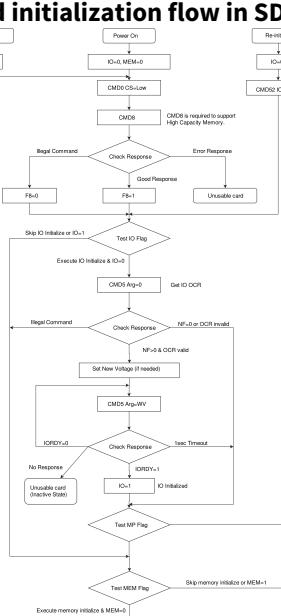
Combo Cards can accept CMD15 with RCA=0000, but there is an exception for SD memory only cards. Memory only cards require a non-zero RCA before the host may issue CMD15. Thus, CMD15 shall be issued after CMD3 in the Standby state. In the case of ACMD41, it shall accept RCA=0x0000.

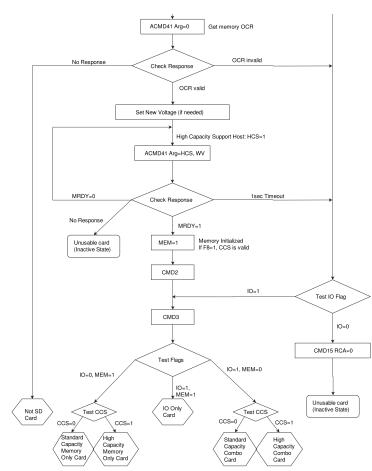
As shown in Figure 4 and Figure 5, an SDIO aware host shall send CMD5 arg=0 as part of the initialization sequence after either Power On or a CMD 52 with write to I/O Reset. Sending CMD5 arg=0 that has not been preceded by one of these two reset conditions shall not result in either the host or card entering the initialization sequence.





| Variablen | Wert | Beschreibung |
|-----------|--|--|
| NF | Number Of Subsystems (ACM41 Response) | GCS: Card Capacity Status (ACM41 Response) |
| MP | Memory Power-up Status (ACM41 Response) | Flag |
| DRDY | 10 Powers Up (0=No DRDY in the ACM41 response) | 10 Function Initiator Flag |
| MRSY | Memory Power-up Status (OCR 80) | Memory Initiated Flag |
| HCS | Host Capacity Support (ACM41 Argument) | CMOS Flag |





Variables

- NP: Number of IO Functions (CMD53 Response)
- MP: Memory Present Flag (CMD53 Response)
- IPRDY: IO Power-up Status (C bit in the CMD53 response)
- MRDY: Memory Power-up Status (CCCR Bit3)
- HCS: Host Capacity Support (ACMD41 Argument)
- CCS: Card Capacity Status (ACMD41 Response)
- Page: Page
- ID: IO Function Initialize Flag
- MEM: Memory Initialized Flag
- rg: ONEW Flag

Card initialization flow in SPI mode (SDIO aware host)

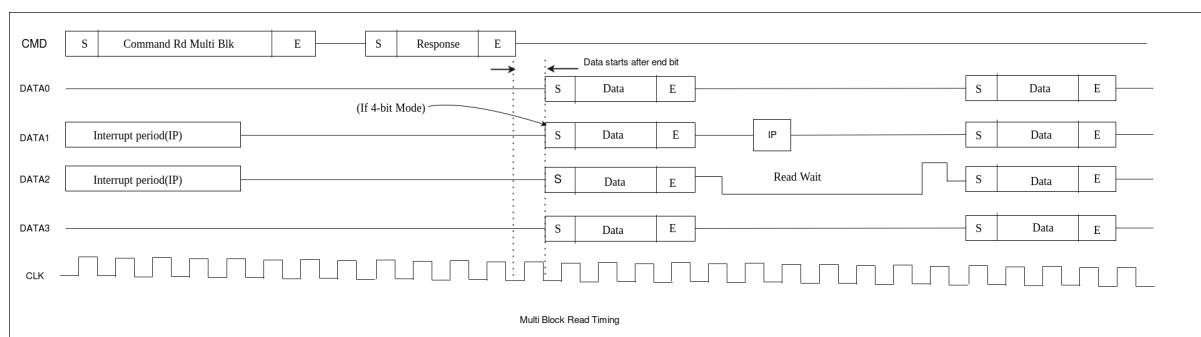
SDIO cards may transfer data in either a multi-byte (1 to 256 bytes) or an optional block format. Any block size from 1 byte to 2048 bytes is possible in order to accommodate the various natural block sizes for I/O functions.

Block Mode

SDIO read or write operation shall be performed on a block basis, rather than the normal byte basis. If Block Mode bit in CMD53 is set, the Byte/Block count value shall contain the number of blocks to be read/written. The block size for functions is set by writing the block size to the I/O block size register in the FBR. The block size for function 0 is set by writing to the FN0 Block Size register in the CCCR. Card and host support of the block I/O mode is optional. The host can determine if a card supports block I/O by reading the Card supports MBIO bit (SMB) in the CCCR. The block size used when Block Mode = 1 and the maximum byte count per command used when Block Mode = 0 can be read from the CIS in the tuple TPLFE_MAX_BLK_SIZE on a per-function basis.

SDIO Read (Multi-Block)

The figure below illustrates the SDIO bus timings for multi-block read operation. For multi block mode, block count is mention in CMD53. The host does not need to stop the transfer, as it continues until the block count is satisfied. If the block count is set to zero, the operation is identical to the memory mode in that the host must stop the transfer.

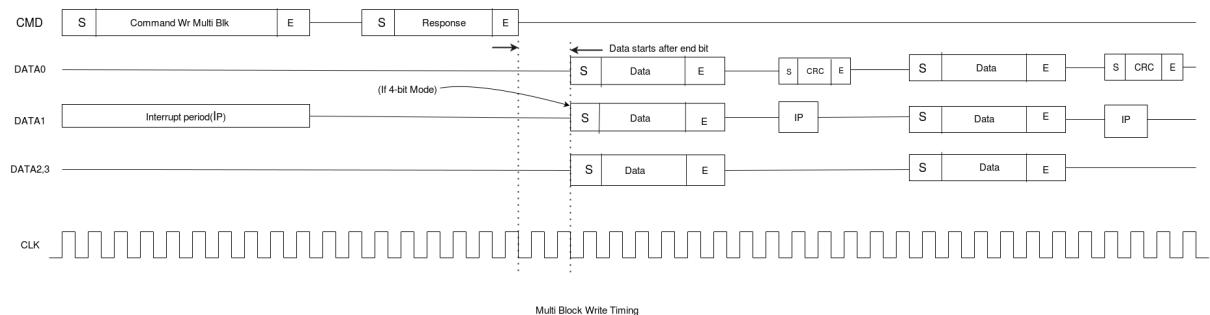


SDIO Multi-Block Read

SDIO Write (Multi-Block)

The figure below illustrates the SDIO bus timings for multi-block write operation. Multiple block write command shall be used rather than continuous single write command to make

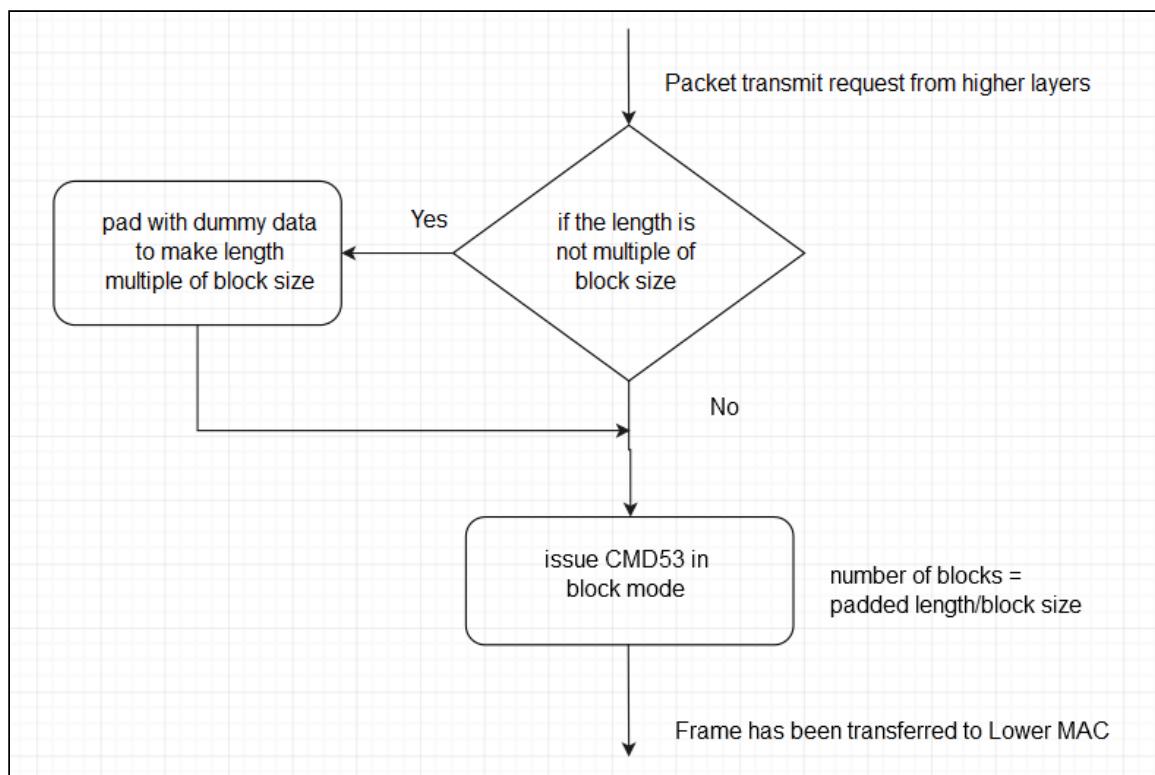
faster write operation.



SDIO Multi-Block Write

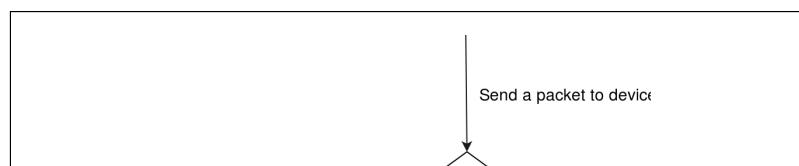
Write Operation Flow diagram

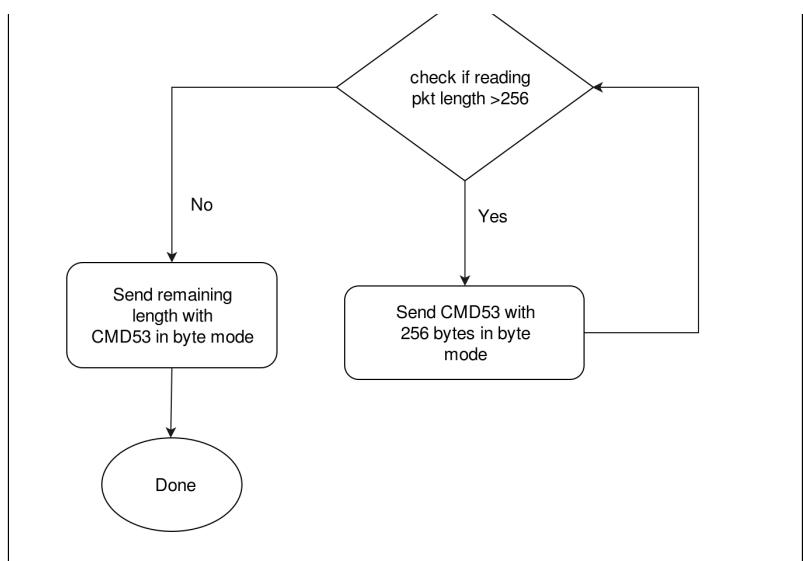
The figures below illustrate the example flow diagrams to be followed by the host driver to send a packet to the SDIO slave in non-block mode and block mode.



14 . SDIO Writing Packet in Block Mode

In Non block mode, Block Mode bit is set to zero in CMD53. The size of the data payload is in the range of 1-256 bytes (due to FIFO size restriction in SDIO slave) in non block mode. The byte count for this transfer is set in the command (CMD53), rather than the fixed block size.

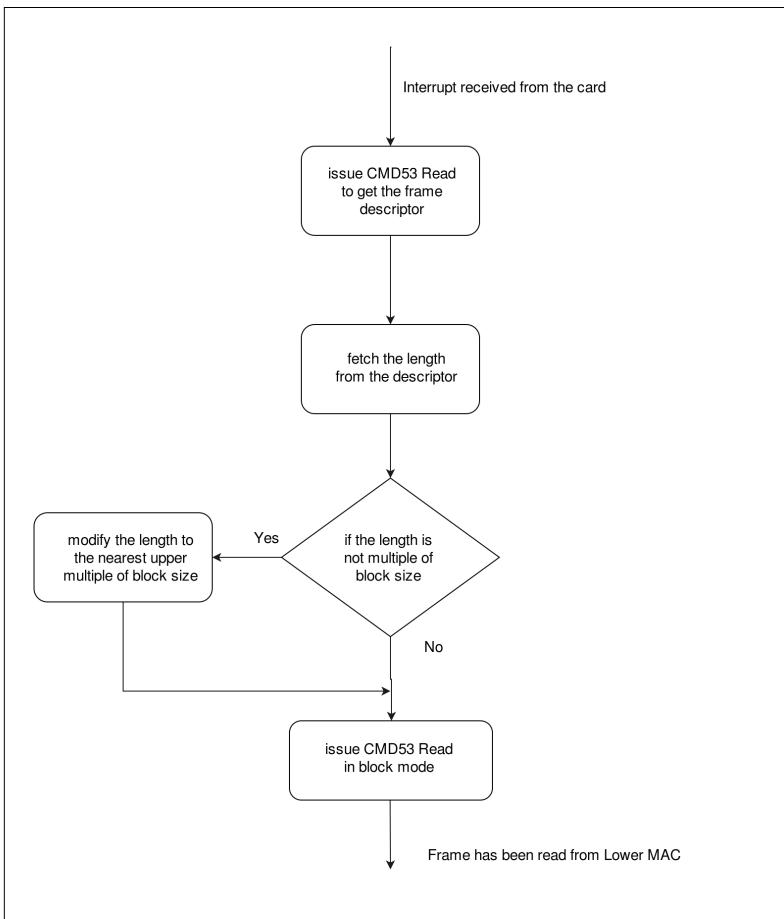




SDIO Writing Packet in Non-block Mode

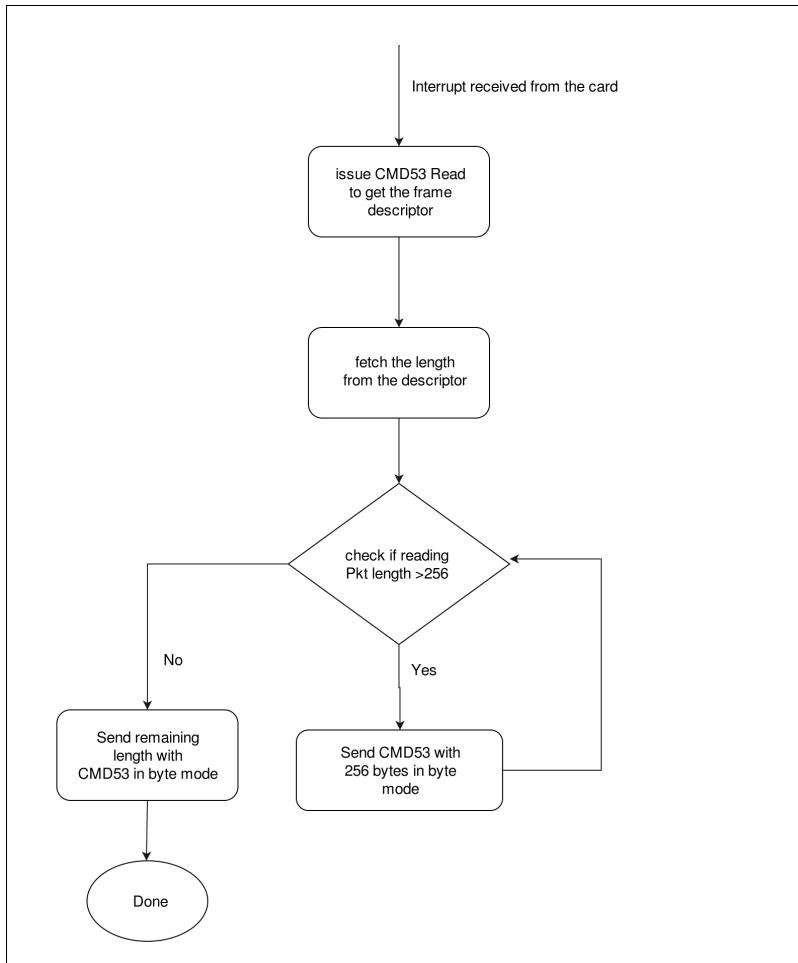
Read Operation Flow diagram

The following figure illustrates the example flow diagrams to be followed by the Host Driver for reading packets from the SDIO slave in block mode.



SDIO Reading Packet in Block Mode

The following figure illustrates the example flow diagrams to be followed by the Host Driver for reading packets from the SDIO slave in non-block mode.



SDIO Reading Packet in Non-Block Mode

Address mapping

The following table shows SDIO address mapping and the allowed commands in the given address space.

| 17-Bit address field | Function Number | CMD52 Access | CMD53 Access | Description |
|----------------------|-----------------|--------------|--------------|-------------------------|
| 0x00000 – 0xFFFF | 0 | Yes | No | SDIO specific registers |

| 17-Bit address field | Function Number | CMD52 Access | CMD53 Access | Description |
|----------------------|-----------------|--------------|--------------|--|
| 0x10000 – 0x1FFFF | 1-5 | Yes | Yes | AHB bus access will be done, with lower 16-bits being picked up from 17-bit address and upper 16-bits being picked up from the fn0 registers |
| 0x00000 – 0x0FFFF | 1-5 | Yes | Yes | An interrupt will be raised to the system side and data has to be read/written through AHB Slave by the system side bus masters |

380 . SDIO Address Mapping

15.4.4 Register Summary

Base Address: 0x2020_0000

The table below describes the vendor specific registers. For description about other registers SDIO standard 2.0 can be referred.

SDIO Function 0 Vendor Specific Register Summary

| Register Name | Offset | Description |
|------------------------------|---------|---|
| OCR [7:0]/SPI_RDATA_WAIT_CNT | 0x000F0 | Operational Conditions Register[7:0] or SPI Read Data Wait count register |
| OCR [15:8] | 0x000F1 | Operational Conditions Register[15:8] |
| OCR [23:16] | 0x000F2 | Operational Conditions Register[23:16] |
| - | 0x000F3 | Reserved |
| RD_NXT_DELAY1 | 0x000F4 | Read Next Delay Register1 |
| RD_NXT_DELAY2 | 0x000F5 | Read Next Delay Register2 |
| DEVICE_ID[7:0] | 0x000F6 | Device ID register |
| DEVICE_ID[15:8] | 0x000F7 | Device ID register |
| VER_NO | 0x000F8 | Version number register |
| INTR_STATUS_REG | 0x000F9 | Function 1 Interrupt Register |
| AHB_MASTER_ACC_ADDR_LSB | 0x000FA | Master Access Address Least Significant Byte Register |
| AHB_MASTER_ACC_ADDR_MSB | 0x000FB | Master Access Address Most Significant Byte Register |
| RFIFO_START_LEVEL | 0x000FC | RFIFO Start Level Register |
| RFIFO_AFULL_LEVEL | 0x000FD | RFIFO Almost Full Level Register |
| WFIFO_AEMPTY_LEVEL | 0x000FE | WFIFO Almost Empty Level Register |
| WAKEUP_REG | 0x000FF | Wakeup Register |

381 . SDIO Function 0 Vendor Specific Register Summary

| Register Name | Offset | Description |
|--------------------------|-------------|--|
| SDIO_INTR_FN1_REG | 0x00 | SDIO Interrupt function1 register |
| SDIO_INTR_FN1_ENABLE_REG | 0x04 | SDIO Interrupt Function1 Enable Register |
| SDIO_INTR_FN1_MASK_REG | 0x08 | SDIO Interrupt Function1 Mask Register |
| SDIO_INTR_FN1_UNMASK_REG | 0x0C | SDIO Interrupt Function1 Unmask Register |
| SDIO_BLK_LEN_REG | 0x08 | SDIO Block Length Register |
| SDIO_BLK_CNT_REG | 0x14 | SDIO Block Count Register |
| SDIO_ADDRESS_REG | 0x18 | SDIO Address Register |
| SDIO_CMD52_RDATA_REG | 0x1C | SDIO Command52 Read Data Register |
| SDIO_CMD52_WDATA_REG | 0x20 | SDIO Command52 Write Data Register |
| SDIO_INTR_REG | 0x24 | SDIO Interrupt Register |
| SDIO_INTR_FN_NUMER_REG | 0x28 | SDIO Interrupt Function Number Register |
| SDIO_FIFO_STATUS_REG | 0x2C | SDIO FIFO Status Register |
| SDIO_FIFO_OCC_REG | 0x30 | SDIO FIFO Occupancy Register |
| SDIO_HOST_INTR_SET_REG | 0x34 | SDIO Host Interrupt Set Register |
| SDIO_HOST_INTR_CLEAR_REG | 0x38 | SDIO Host Interrupt Clear Register |
| SDIO_RFIFO_DATA_REG | 0x40 – 0x7E | SDIO Read FIFO Data Register |
| SDIO_WFIFO_DATA_REG | 0x80 – 0xBE | SDIO Write FIFO Data Register |
| SDIO_INTR_FN2_REG | 0xC0 | SDIO Interrupt function2 register |
| SDIO_INTR_FN2_ENABLE_REG | 0xC4 | SDIO Interrupt Function2 Enable Register |
| SDIO_INTR_FN2_MASK_REG | 0xC8 | SDIO Interrupt Function2 Mask Register |
| SDIO_INTR_FN2_UNMASK_REG | 0xCC | SDIO Interrupt Function2 Unmask Register |
| SDIO_INTR_FN3_REG | 0xD0 | SDIO Interrupt function3 register |
| SDIO_INTR_FN3_ENABLE_REG | 0xD4 | SDIO Interrupt Function3 Enable Register |
| SDIO_INTR_FN3_MASK_REG | 0xD8 | SDIO Interrupt Function3 Mask Register |
| SDIO_INTR_FN3_UNMASK_REG | 0xDC | SDIO Interrupt Function3 Unmask Register |
| SDIO_INTR_FN4_REG | 0xE0 | SDIO Interrupt function4 register |
| SDIO_INTR_FN4_ENABLE_REG | 0xE4 | SDIO Interrupt Function4 Enable Register |
| SDIO_INTR_FN4_MASK_REG | 0xE8 | SDIO Interrupt Function4 Mask Register |
| SDIO_INTR_FN4_UNMASK_REG | 0xEC | SDIO Interrupt Function4 Unmask Register |
| SDIO_INTR_FN5_REG | 0xF0 | SDIO Interrupt function5 register |
| SDIO_INTR_FN5_ENABLE_REG | 0xF4 | SDIO Interrupt Function5 Enable Register |

| Register Name | Offset | Description |
|---------------------------------|---------------|--|
| SDIO_INTR_FN5_MASK_REG | 0xF8 | SDIO Interrupt Function5 Mask Register |
| SDIO_INTR_FN5_UNMASK_REG | 0xFC | SDIO Interrupt Function5 Unmask Register |
| SDIO_ERROR_COND_CTRL_ENABLE_REG | 0x100 | SDIO Error Condition Control Enable Register |
| SDIO_ERROR_COND_BLK_CNT | 0x104 | SDIO Error Condition Block Count Register |
| SDIO_BOOT_CONFIG_VALS_0 | 0x108 | SDIO Boot Config Vals 0 Register |
| SDIO_BOOT_CONFIG_VALS_1 | 0x10C | SDIO Boot Config Vals 1 Register |
| | Fn0 Registers | |
| SDIO_CCCR Registers | 0x200 – 0x23E | CCCR Registers (0x0000 – SDIO space address) |
| SDIO Vendor Unique Registers | 0x240 – 0x25E | Vendor Specific Registers (0x00F0) |
| SDIO Function1 FBR Registers | 0x260 – 0x27E | Function1 FBR Registers (0x0100) |
| SDIO Function2 FBR Registers | 0x280 – 0x29E | Function2 FBR Registers (0x0200) |
| SDIO Function3 FBR Registers | 0x2A0 – 0x2BE | Function3 FBR Registers (0x0300) |
| SDIO Function4 FBR Registers | 0x2C0 – 0x2DE | Function4 FBR Registers (0x0400) |
| SDIO Function5 FBR Registers | 0x2E0 – 0x2FE | Function5 FBR Registers (0x0500) |
| SDIO CIS Register | 0x300 – 0x4FE | CIS Registers (0x1000) |

382 . SDIO AHB Slave Register Summary

15.4.5 Register Description

Function0 Registers

OCR [7:0]/ SPI_RDATA_WAIT_CNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------|-----------|--|
| [7:0] | R/W | SDIO_OCR1 | 0x00/0x80 | <p>During bootup configuration, writes to this address will get written into OCR [7:0] register. After bootup configuration is over accesses to this address will access SPI_RDATA_WAIT_CNT register.</p> <p>OCR – Operational conditions register indicates the card operating conditions which will be read by the host through CMD5</p> <p>SPI_RDATA_WAIT_CNT – Indicates the number cycles to be waited for RFIFO to reach minimum occupancy before issuing interrupt during SD SPI mode reads</p> |

383 . SDIO Operational Conditions Register[7:0] Description

OCR[15:8]/vendor_specific_reg0

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------|-----------|---|
| [7:0] | R | SDIO_OCR2 | 0x80/0x00 | OCR [15:8] This register can be written only during bootload configuration After bootup configuration, this register will act as vendor_specific_reg0 |

384 . SDIO Operational Conditions Register [15:8] Description

OCR [23:16]/vendor_specific_reg1

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------|-----------|---|
| [7:0] | R/W | SDIO_OCR3 | 0xFF/0x00 | OCR [23:16] - This register can be written only during bootload configuration After bootup configuration, this register will act as vendor_specific_reg1 |

385 . SDIO Operational Conditions Register [23:16] Description

RD_NXT_DELAY1

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------------|-----------|--|
| [7:0] | R/W | SDIO_RD_NXT_DELA Y1 | 0xC8 | Minimum delay in terms of number of SDIO clocks, between read blocks (SDIO_RD_NXT_DELAY [7:0]) |

386 . Read Next Delay Register 1 Description

RD_NXT_DELAY2

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------|-----------|---|
| 7 | R | Reserved | 0x0 | Reserved for future use. |
| 6 | R/W | pass_async_interru pt_en | 0x0 | pass asynchronous interrupt enable |
| 5 | R/W | SDIO_WRPTR_INCR0 _32BIT | 0x0 | If this bit is set FIFO write pointer is incremented by 2 once 32 bits of data is received. (Usually set in sleep mode during which SoC clock will be lower than host clock) |
| 4 | R/W | SDIO_CSA_THROU GH-SLAVE | 0x0 | This bit is used to enable CSA through AHB Slave |
| 3 | R/W | SDIO_CRC_STATUS _DIS | 1'b0 | SDIO Write CRC error status disable 0 – CRC status enabled 1 – CRC status disabled If the driver is exercising this option, it has to be done preferably after loading the instructions and templates. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|--|
| 2 | R/W | SDIO_RD_NXT_DEL AY_EN | 1'b0 | Enable for introducing minimum delay between consecutive read blocks |
| [1:0] | R/W | SDIO_RD_NXT_DEL AY2 | 2'b00 | Minimum delay in terms of number of SDIO clocks, between read blocks (SDIO_RD_NXT_DELAY [9:8]) |

387 . Read Next Delay Register 2 Description

DEVICE_ID[7:0]

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------|-----------|------------------------------|
| [7:0] | R | SDIO_DEVID | 0x16 | This indicates the Device ID |

388 . Device ID Description

DEVICE_ID[15:8]

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------|-----------|------------------------------|
| [15:8] | R | SDIO_DEVID | 0x91 | This indicates the Device ID |

389 . Device ID Description

VER_NO

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------|-----------|---|
| [7:0] | R | SDIO_VER_NO | 0x02 | This indicates the version number of the device |

390 . Version number register Description

INTR_STATUS_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------|-----------|--|
| [7:0] | R/W | SDIO_FN1_INTR | 0x00 | The Function1 Interrupt status from the System |

391 . Function 1 interrupt register Description

AHB_MASTER_ACC_ADDR_LSB

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------|-----------|--|
| [7:0] | R/W | SDIO_MASTER_ADR_LSB | 0x00 | Bits 23 to 16 of the AHB Master access address |

392 . Master Access Address Least Significant Byte Register Description

AHB_MASTER_ACC_ADDR_MSB

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 7:0] | R/W | SDIO_MASTER_ADR_MSB 0x00 | | Bits 31 to 24 of the AHB Master access address |

393 . Master Access Address Most Significant Byte Register Description

RFIFO_START_LEVEL

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------|-----------|---|
| [7:0] | R/W | SDIO_RFIFO_START_lev | 0x04 | Minimum FIFO occupancy level before which data will not be read out. This has to be specified in number of half words (16-bit). |

394 . RFIFO Start Level Register Description

RFIFO_AFULL_LEVEL

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------|-----------|--|
| [7:0] | R/W | SDIO_RFIFO_AFULL_lev | 0x08 | If the read FIFO occupancy is greater than this level, read FIFO's almost full signal will be asserted. This has to be specified in number of half words (16-bit). |

395 . RFIFO Almost Full Level Register Description

WFIFO_AEMPTY_LEVEL

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|--|
| [7:0] | R/W | SDIO_WFIFO_AEMPTY_lev | 0x08 | If the write FIFO occupancy is less than this level, write FIFO's almost empty signal will be asserted. This has to be specified in number of half words (16-bit). |

396 . WFIFO Almost Empty Level Register Description

WAKEUP_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|--|
| [7:1] | R | Reserved | 0x00 | Reserved |
| 0 | W | SDIO_WAKEUP_INTERRUPT | 0b0 | Wakeup Interrupt – Writing '1' into this register will give wakeup interrupt to the sleep fsm. This is a self clearing bit |

397 . Wake up Register Description

SDIO_INTR_FN1_REGISTER

| Bit | Access | Function | POR Value | Description |
|---------|--------|------------------|-----------|---|
| [31:10] | R/W | Reserved | 0x0 | Reserved |
| 9 | R/W | SDIO_CSA_ACCE_SS | 0x0 | cса_window_access When set, indicates that current request is for CSA window register. This is only status signal |
| 8 | R/W | SDIO_WR_RDz | 0x0 | wr_rdz 0 – read request 1 – write request This is not an interrupt signal. This is only status signal |
| 7 | R/W | SDIO_RD_TOUT_INT | 0x0 | When set, indicates that Read FIFO hasn't reached minimum threshold value before read wait timeout during SDIO SPI mode read. |
| 6 | R/W | SDIO_ABORT_INT | 0x0 | When set, indicates that host issued an abort command |
| 5 | R/W | SDIO_CRC_ERR_INT | 0x0 | When set, indicates that data block is received with crc error |
| 4 | R/W | SDIO_PWRLEV_INT | 0x0 | When set, indicates that power control register value has been changed by the host |
| 3 | R/W | SDIO_CMD52_INT | 0x0 | When set, indicates that CMD52 is received |
| 2 | R/W | SDIO_CSA_INT | 0x0 | When set, indicates that CMD53 request is received to CSA |
| 1 | R/W | SDIO_RD_INT | 0x0 | When set, indicates that CMD53 read request is received |
| 0 | R/W | SDIO_WR_INT | 0x0 | When set, indicates that CMD53 write request is received |

398 . SDIO Function1 Interrupt Status/Clear Register Description

SDIO_INTR_FN1_ENABLE_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_INT_EN | 0x0 | This bit is used to enable "read FIFO wait time over" interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 6 | R/W | SDIO_ABORT_INT_EN | 0x0 | This bit is used to enable abort interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 5 | R/W | SDIO_CRC_ERR_INT_EN | 0x0 | This bit is used to enable CRC error interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|--------------------|------------------|---|
| 4 | R/W | SDIO_PWRLEV_INT_EN | 0x0 | This bit is used to enable power level change interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 3 | R/W | SDIO_CMD52_INT_EN | 0x0 | This bit is used to enable CMD52 interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 2 | R/W | SDIO_CSA_INT_EN | 0x0 | This bit is used to enable CMD53 CSA interrupt |
| 1 | R/W | SDIO_RDINTEN | 0x0 | This bit is used to enable CMD53 read interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 0 | R/W | SDIO_WRINTEN | 0x0 | This bit is used to enable CMD53 write interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

399 . SDIO Function1 Interrupt Enable Register Description

SDIO_INTR_FN1_MASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|------------------|------------------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_MSK | 0x0 | This bit is used to mask "read FIFO wait time over" interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_MSK | 0x0 | This bit is used to mask abort interrupt Setting this bit will mask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_MSK | 0x0 | This bit is used to mask CRC error interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_MSK | 0x0 | This bit is used to mask power level change interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_MSK | 0x0 | This bit is used to mask CMD52 interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_MSK | 0x0 | This bit is used to mask CMD53 CSA interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RDINT_MSK | 0x0 | This bit is used to mask CMD53 read interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------|-----------|--|
| 0 | R/W | SDIO_WR_INT_M SK | 0x0 | This bit is used to mask CMD53 write interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |

400 . SDIO Function1 Interrupt Mask Register Description

SDIO_INTR_FN1_UNMASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_UN MSK | 0x0 | This bit is used to unmask "read FIFO wait time over" interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_ UNMSK | 0x0 | This bit is used to unmask abort interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_ UNMSK | 0x0 | This bit is used to unmask CRC error interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_ UNMSK | 0x0 | This bit is used to unmask power level change interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_UN MSK | 0x0 | This bit is used to unmask CMD52 interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_UNMSK | 0x0 | This bit is used to unmask CMD53 CSA interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 read interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 write interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |

401 . SDIO Function1 Interrupt Unmask Register Description

SDIO_BLK_LEN_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------------|--------|----------|-----------|-------------------------|
| [31:1] 2] | R | Reserved | 0x0 | Reserved for future use |

| Bit | Access | Function | POR Value | Description |
|----------|--------|--------------|-----------|--|
| [11 : 0] | R | SDIO_BLK_LEN | 0x0 | Length of each block for the last received CMD53 |

402 . SDIO Block Length Register Description

SDIO_BLK_CNT_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------|-----------|---|
| [31:9] | R | Reserved | 0x0 | Reserved for future use. |
| [8:0] | R | SDIO_BLK_CNT | 0x0 | Block count for the last received CMD53 |

403 . SDIO Block Count Register Description

SDIO_ADDRESS_REGISTER

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------|-----------|--|
| [31:16] | R | Reserved | 0x0 | Reserved |
| [15:0] | R | SDIO_ADDR | 0x0 | Lower 16-bits of the 17-bit address field in the last received CMD53 |

404 . SDIO Address Register Description

SDIO_CMD52_RDATA_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------|-----------|---|
| [31:8] | R | Reserved | 0x0 | Reserved for future use. |
| [7:0] | R | RDATA | 0x0 | Data to be given to host for CMD52 slave mode access read command has to written into this register |

405 . SDIO CMD52 RDATA Register Description

SDIO_CMD52_WDATA_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------|-----------|---|
| [31:8] | R | Reserved | 0x0 | Reserved for future use. |
| [7:0] | R | WDATA | 0xFF | Data from host in CMD52 slave mode access write command is available in this register |

406 . SDIO CMD52 WDATA Register Description

SDIO_INTR_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------|-----------|---|
| [31:6] | R | Reserved | 0x0 | Reserved for future use. |
| 5 | R | SDIO_INT_FN5 | 0x0 | Interrupt is pending for function5 |
| 4 | R | SDIO_INT_FN4 | 0x0 | Interrupt is pending for function4 |
| 3 | R | SDIO_INT_FN3 | 0x0 | Interrupt is pending for function3 |
| 2 | R | SDIO_INT_FN2 | 0x0 | Interrupt is pending for function2 |
| 1 | R | SDIO_INT_FN1 | 0x0 | Interrupt is pending for function1 |
| 0 | R | SDIO_INT_ERRO | 0x0 | Interrupt is pending because of error condition from any of the functions |

407 . SDIO Interrupt Status Register Description

SDIO_INTR_FN_NUMER_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------|-----------|---|
| [31:3] | R | Reserved | 0x0 | Reserved |
| [2:0] | R | SDIO_INTR_FUN_NO | 0x0 | <p>Indicates the function number to which interrupt is pending. This register is provided to enable the software to decode the interrupt register easily. Once this interrupt is cleared this register gets the next function number to which interrupt is pending (if simultaneous interrupts are pending).</p> <p>2 – function 2 3 – function 3 4 – function 4 5 – function 5 0 – there is no pending interrupt</p> <p>There are two interrupt lines coming out of SDIO. One line is dedicate to function 1. Remaining function interrupts will come on second line. Since function 1 has dedicate interrupt line, pending of the same is not mapped into this register</p> |

408 . SDIO Interrupt Function Number Register Description

SDIO_FIFO_STATUS_REGISTER

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------|-----------|-------------|
| [31:12] | R | | 0x0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------------------------|-----------|---|
| [11:7] | R | SDIO_BUS_CONT ROL_STATE | 0x1 | SDIO bus control state 1– When set, indicates FSM is in idle state 2– When set, indicates FSM is in CMD52 read state 4– When set, indicates FSM is in CMD52 write state 8– When set, indicates FSM is CMD53 read state 16 – When set, indicates FSM is CMD53 write state |
| [6:4] | R | SDIO_CURRENT_ FN_NO | 0x0 | Indicates the function number of the last received command |
| 3 | R | SDIO_RFIFO_AE MPTY | 0x0 | When set, indicates that RFIFO is almost empty |
| 2 | R | SDIO_RFIFO_EM PTY | 0x0 | When set, indicates that RFIFO is empty RFIFO is used in SDIO writes from host for sending data from host to AHB |
| 1 | R | SDIO_WFIFO_AF ULL | 0x0 | When set, indicates that WFIFO is almost full |
| 0 | R | SDIO_WFIFO_FU LL | 0x0 | When set, indicates that WFIFO is full WFIFO is used in SDIO reads from host for sending data from AHB to Host |

409 . SDIO FIFO Status Register Description

SDIO_FIFO_OCC_REGISTER

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------------------|-----------|---|
| [31:16] | R | Reserved | 0x0 | Reserved |
| [15:8] | R | SDIO_RFIFO_AVA IL | 0x80 | Indicates the available space in the read FIFO |
| [7:0] | R | SDIO_WFIFO_OC C | 0x0 | Indicates the occupancy level of the write FIFO |

410 . SDIO FIFO Occupancy Register Description

SDIO_HOST_INTR_SET_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------------|-----------|--|
| [31:4] | R/W | Reserved | 0x0 | Reserved for future use |
| 3 | R/W | SDIO_INTSET_FN 5 | 0x0 | This bit is used to raise an interrupt to host for function5 Setting this bit will raise the interrupt Clearing this bit has no effect |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 2 | R/W | SDIO_INTSET_FN4 | 0x0 | This bit is used to raise an interrupt to host for function4. Setting this bit will raise the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_INTSET_FN3 | 0x0 | This bit is used to raise an interrupt to host for function3. Setting this bit will raise the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_INTSET_FN2 | 0x0 | This bit is used to raise an interrupt to host for function2. Setting this bit will raise the interrupt Clearing this bit has no effect |

411 . SDIO Host Interrupt Set Register Description

SDIO_HOST_INTR_CLEAR_REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| [31:4] | R/W | Reserved | 0x0 | Reserved for future use. |
| 3 | R/W | SDIO_INTCLR_FN5 | 0x0 | This bit is used to clear the interrupt to host for function5. Setting this bit will clear the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_INTCLR_FN4 | 0x0 | This bit is used to clear the interrupt to host for function4. Setting this bit will clear the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_INTCLR_FN3 | 0x0 | This bit is used to clear the interrupt to host from function3. Setting this bit will clear the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_INTCLR_FN2 | 0x0 | This bit is used to clear the interrupt to host for function2. Setting this bit will clear the interrupt Clearing this bit has no effect |

412 . SDIO Host Interrupt Clear Register Description

SDIO_INTR_FN2_REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| [31:10] | R/W | Reserved | 0x0 | Reserved |
| 9 | R/W | SDIO_CSA_ACCESS | 0x0 | csa_window_access When set, indicates that current request is for CSA window register. This is only status signal |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------|-----------|---|
| 8 | R/W | SDIO_WR_RDz | 0x0 | wr_rdz 0 – read request 1 – write request This is not an interrupt signal. This is only status signal |
| 7 | R/W | SDIO_RD_TOUT_ INT | 0x0 | When set, indicates that Read FIFO hasn't reached minimum threshold value before read wait timeout during SDIO SPI mode read. |
| 6 | R/W | SDIO_ABORT_ INT | 0x0 | When set, indicates that host issued an abort command |
| 5 | R/W | SDIO_CRC_ ERR_INT | 0x0 | When set, indicates that data block is received with crc error |
| 4 | R/W | SDIO_PWRLEV_ INT | 0x0 | When set, indicates that power control register value has been changed by the host |
| 3 | R/W | SDIO_CMD52_ INT | 0x0 | When set, indicates that CMD52 is received |
| 2 | R/W | SDIO_CSA_ INT | 0x0 | When set, indicates that CMD53 request is received to CSA |
| 1 | R/W | SDIO_RD_ INT | 0x0 | When set, indicates that CMD53 read request is received |
| 0 | R/W | SDIO_WR_ INT | 0x0 | When set, indicates that CMD53 write request is received |

413 . SDIO Function2 Interrupt Status/Clear Register Description

SDIO_INTR_FN2_ENABLE_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_ INT _EN | 0x0 | This bit is used to enable "read FIFO wait time over" interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 6 | R/W | SDIO_ABORT_ INT_EN | 0x0 | This bit is used to enable abort interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 5 | R/W | SDIO_CRC_ ERR_INT_EN | 0x0 | This bit is used to enable CRC error interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 4 | R/W | SDIO_PWRLEV_ INT_EN | 0x0 | This bit is used to enable power level change interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 3 | R/W | SDIO_CMD52_ INT_EN | 0x0 | This bit is used to enable CMD52 interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|--|
| 2 | R/W | SDIO_CSA_INT_EN | 0x0 | This bit is used to enable CMD53 CSA interrupt |
| 1 | R/W | SDIO_RD_INT_EN | 0x0 | This bit is used to enable CMD53 read interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 0 | R/W | SDIO_WR_INT_EN | 0x0 | This bit is used to enable CMD53 write interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

414 . SDIO Function2 Interrupt Enable Register Description

SDIO_INTR_FN2_MASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_MSK | 0x0 | This bit is used to mask "read FIFO wait time over" interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_MSK | 0x0 | This bit is used to mask abort interrupt Setting this bit will mask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_MSK | 0x0 | This bit is used to mask CRC error interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_MSK | 0x0 | This bit is used to mask power level change interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_MSK | 0x0 | This bit is used to mask CMD52 interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_MSK | 0x0 | This bit is used to mask CMD53 CSA interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_MSK | 0x0 | This bit is used to mask CMD53 read interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_MSK | 0x0 | This bit is used to mask CMD53 write interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |

415 . SDIO Function2 Interrupt Mask Register Description

SDIO_INTR_FN2_UNMASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_UNMSK | 0x0 | This bit is used to unmask "read FIFO wait time over" interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_UNMSK | 0x0 | This bit is used to unmask abort interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_UNMSK | 0x0 | This bit is used to unmask CRC error interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_UNMSK | 0x0 | This bit is used to unmask power level change interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_UNMSK | 0x0 | This bit is used to unmask CMD52 interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_UNMSK | 0x0 | This bit is used to unmask CMD53 CSA interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_UNMSK | 0x0 | This bit is used to unmask CMD53 read interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_UNMSK | 0x0 | This bit is used to unmask CMD53 write interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |

416 . SDIO Function2 Interrupt Unmask Register Description

SDIO_INTR_FN3_REGISTER

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|--|
| [31:10] | R/W | Reserved | 0x0 | Reserved |
| 9 | R/W | SDIO_CSA_ACCESS | 0x0 | cса_window_access When set, indicates that current request is for CSA window register. This is only status signal |
| 8 | R/W | SDIO_WR_RDZ | 0x0 | wr_rdz 0 – read request 1 – write request This is not an interrupt signal. This is only status signal |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------|-----------|---|
| 7 | R/W | SDIO_RD_TOUT_INT | 0x0 | When set, indicates that Read FIFO hasn't reached minimum threshold value before read wait timeout during SDIO SPI mode read. |
| 6 | R/W | SDIO_ABORT_INT | 0x0 | When set, indicates that host issued an abort command |
| 5 | R/W | SDIO_CRC_ERR_INT | 0x0 | When set, indicates that data block is received with crc error |
| 4 | R/W | SDIO_PWRLEV_INT | 0x0 | When set, indicates that power control register value has been changed by the host |
| 3 | R/W | SDIO_CMD52_INT | 0x0 | When set, indicates that CMD52 is received |
| 2 | R/W | SDIO_CSA_INT | 0x0 | When set, indicates that CMD53 request is received to CSA |
| 1 | R/W | SDIO_RD_INT | 0x0 | When set, indicates that CMD53 read request is received |
| 0 | R/W | SDIO_WR_INT | 0x0 | When set, indicates that CMD53 write request is received |

417 . SDIO Function3 Interrupt Status/Clear Register Description

SDIO_INTR_FN3_ENABLE_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_INT_EN | 0x0 | This bit is used to enable "read FIFO wait time over" interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 6 | R/W | SDIO_ABORT_INT_EN | 0x0 | This bit is used to enable abort interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 5 | R/W | SDIO_CRC_ERR_INT_EN | 0x0 | This bit is used to enable CRC error interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 4 | R/W | SDIO_PWRLEV_INT_EN | 0x0 | This bit is used to enable power level change interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 3 | R/W | SDIO_CMD52_INT_EN | 0x0 | This bit is used to enable CMD52 interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 2 | R/W | SDIO_CSA_INT_EN | 0x0 | This bit is used to enable CMD53 CSA interrupt |
| 1 | R/W | SDIO_RD_INT_EN | 0x0 | This bit is used to enable CMD53 read interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|--|
| 0 | R/W | SDIO_WR_INT_E_N | 0x0 | This bit is used to enable CMD53 write interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

418 . SDIO Function3 Interrupt Enable Register Description

SDIO_INTR_FN3_MASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_MSK | 0x0 | This bit is used to mask "read FIFO wait time over" interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_MSK | 0x0 | This bit is used to mask abort interrupt Setting this bit will mask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_MSK | 0x0 | This bit is used to mask CRC error interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_MSK | 0x0 | This bit is used to mask power level change interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_MSK | 0x0 | This bit is used to mask CMD52 interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_MSK | 0x0 | This bit is used to mask CMD53 CSA interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_MSK | 0x0 | This bit is used to mask CMD53 read interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_MSK | 0x0 | This bit is used to mask CMD53 write interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |

419 . SDIO Function3 Interrupt Mask Register Description

SDIO_INTR_FN3_UNMASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------|-----------|-------------|
| [31:8] | R/W | Reserved | 0x0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 7 | R/W | SDIO_TOUT_UN MSK | 0x0 | This bit is used to unmask "read FIFO wait time over" interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_ UNMSK | 0x0 | This bit is used to unmask abort interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_ UNMSK | 0x0 | This bit is used to unmask CRC error interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_ UNMSK | 0x0 | This bit is used to unmask power level change interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_UN MSK | 0x0 | This bit is used to unmask CMD52 interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_UNMS K | 0x0 | This bit is used to unmask CMD53 CSA interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 read interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 write interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |

420 . SDIO Function3 Interrupt Unmask Register Description

SDIO_INTR_FN4_REGISTER

| Bit | Access | Function | POR Value | Description |
|-------------|--------|----------------------|-----------|---|
| [31:1 0] | R/W | Reserved | 0x0 | Reserved |
| 9 | R/W | SDIO_CSA_ACCE SS | 0x0 | csa_window_access When set, indicates that current request is for CSA window register. This is only status signal |
| 8 | R/W | SDIO_WR_RDz | 0x0 | wr_rdz 0 – read request 1 – write request This is not an interrupt signal. This is only status signal |
| 7 | R/W | SDIO_RD_TOUT_ INT | 0x0 | When set, indicates that Read FIFO hasn't reached minimum threshold value before read wait timeout during SDIO SPI mode read. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------|-----------|--|
| 6 | R/W | SDIO_ABORT_INT | 0x0 | When set, indicates that host issued an abort command |
| 5 | R/W | SDIO_CRC_ERR_INT | 0x0 | When set, indicates that data block is received with crc error |
| 4 | R/W | SDIO_PWRLEV_INT | 0x0 | When set, indicates that power control register value has been changed by the host |
| 3 | R/W | SDIO_CMD52_INT | 0x0 | When set, indicates that CMD52 is received |
| 2 | R/W | SDIO_CSA_INT | 0x0 | When set, indicates that CMD53 request is received to CSA |
| 1 | R/W | SDIO_RD_INT | 0x0 | When set, indicates that CMD53 read request is received |
| 0 | R/W | SDIO_WR_INT | 0x0 | When set, indicates that CMD53 write request is received |

421 . SDIO Function4 Interrupt Status/Clear Register Description

SDIO_INTR_FN4_ENABLE_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_INT_EN | 0x0 | This bit is used to enable "read FIFO wait time over" interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 6 | R/W | SDIO_ABORT_INT_EN | 0x0 | This bit is used to enable abort interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 5 | R/W | SDIO_CRC_ERR_INT_EN | 0x0 | This bit is used to enable CRC error interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 4 | R/W | SDIO_PWRLEV_INT_EN | 0x0 | This bit is used to enable power level change interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 3 | R/W | SDIO_CMD52_INT_EN | 0x0 | This bit is used to enable CMD52 interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 2 | R/W | SDIO_CSA_INT_EN | 0x0 | This bit is used to enable CMD53 CSA interrupt |
| 1 | R/W | SDIO_RD_INT_EN | 0x0 | This bit is used to enable CMD53 read interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|--|
| 0 | R/W | SDIO_WR_INT_E_N | 0x0 | This bit is used to enable CMD53 write interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

422 . SDIO Function4 Interrupt Enable Register Description

SDIO_INTR_FN4_MASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_MSK | 0x0 | This bit is used to mask "read FIFO wait time over" interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_MSK | 0x0 | This bit is used to mask abort interrupt Setting this bit will mask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_MSK | 0x0 | This bit is used to mask CRC error interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_MSK | 0x0 | This bit is used to mask power level change interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_MSK | 0x0 | This bit is used to mask CMD52 interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_MSK | 0x0 | This bit is used to mask CMD53 CSA interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_MSK | 0x0 | This bit is used to mask CMD53 read interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_MSK | 0x0 | This bit is used to mask CMD53 write interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |

423 . SDIO Function4 Interrupt Mask Register Description

SDIO_INTR_FN4_UNMASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------|-----------|-------------|
| [31:8] | R/W | Reserved | 0x0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 7 | R/W | SDIO_TOUT_UN MSK | 0x0 | This bit is used to unmask "read FIFO wait time over" interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_ UNMSK | 0x0 | This bit is used to unmask abort interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_ UNMSK | 0x0 | This bit is used to unmask CRC error interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_ UNMSK | 0x0 | This bit is used to unmask power level change interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_UN MSK | 0x0 | This bit is used to unmask CMD52 interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_UNMS K | 0x0 | This bit is used to unmask CMD53 CSA interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 read interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 write interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |

424 . SDIO Function4 Interrupt Unmask Register Description

SDIO_INTR_FN5_REGISTER

| Bit | Access | Function | POR Value | Description |
|-------------|--------|----------------------|-----------|---|
| [31:1 0] | R/W | Reserved | 0x0 | Reserved |
| 9 | R/W | SDIO_CSA_ACCE SS | 0x0 | csa_window_access When set, indicates that current request is for CSA window register. This is only status signal |
| 8 | R/W | SDIO_WR_RDz | 0x0 | wr_rdz 0 – read request 1 – write request This is not an interrupt signal. This is only status signal |
| 7 | R/W | SDIO_RD_TOUT_ INT | 0x0 | When set, indicates that Read FIFO hasn't reached minimum threshold value before read wait timeout during SDIO SPI mode read. |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|------------------|------------------|--|
| 6 | R/W | SDIO_ABORT_INT | 0x0 | When set, indicates that host issued an abort command |
| 5 | R/W | SDIO_CRC_ERR_INT | 0x0 | When set, indicates that data block is received with crc error |
| 4 | R/W | SDIO_PWRLEV_INT | 0x0 | When set, indicates that power control register value has been changed by the host |
| 3 | R/W | SDIO_CMD52_INT | 0x0 | When set, indicates that CMD52 is received |
| 2 | R/W | SDIO_CSA_INT | 0x0 | When set, indicates that CMD53 request is received to CSA |
| 1 | R/W | SDIO_RD_INT | 0x0 | When set, indicates that CMD53 read request is received |
| 0 | R/W | SDIO_WR_INT | 0x0 | When set, indicates that CMD53 write request is received |

425 . SDIO Function5 Interrupt Status/Clear Register Description

SDIO_INTR_FN5_ENABLE_REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|---------------------|------------------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_INT_EN | 0x0 | This bit is used to enable "read FIFO wait time over" interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 6 | R/W | SDIO_ABORT_INT_EN | 0x0 | This bit is used to enable abort interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 5 | R/W | SDIO_CRC_ERR_INT_EN | 0x0 | This bit is used to enable CRC error interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 4 | R/W | SDIO_PWRLEV_INT_EN | 0x0 | This bit is used to enable power level change interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 3 | R/W | SDIO_CMD52_INT_EN | 0x0 | This bit is used to enable CMD52 interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |
| 2 | R/W | SDIO_CSA_INT_EN | 0x0 | This bit is used to enable CMD53 CSA interrupt |
| 1 | R/W | SDIO_RD_INT_EN | 0x0 | This bit is used to enable CMD53 read interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|--|
| 0 | R/W | SDIO_WR_INT_E_N | 0x0 | This bit is used to enable CMD53 write interrupt. '1'- Interrupt is enabled '0'- Interrupt is disabled |

426 . SDIO Function5 Interrupt Enable Register Description

SDIO_INTR_FN5_MASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------|-----------|---|
| [31:8] | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | SDIO_TOUT_MSK | 0x0 | This bit is used to mask "read FIFO wait time over" interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_MSK | 0x0 | This bit is used to mask abort interrupt Setting this bit will mask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_MSK | 0x0 | This bit is used to mask CRC error interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_MSK | 0x0 | This bit is used to mask power level change interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_MSK | 0x0 | This bit is used to mask CMD52 interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_MSK | 0x0 | This bit is used to mask CMD53 CSA interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_MSK | 0x0 | This bit is used to mask CMD53 read interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_MSK | 0x0 | This bit is used to mask CMD53 write interrupt. Setting this bit will mask the interrupt Clearing this bit has no effect |

427 . SDIO Function5 Interrupt Mask Register Description

SDIO_INTR_FN5_UNMASK_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------|-----------|-------------|
| [31:8] | R/W | Reserved | 0x0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 7 | R/W | SDIO_TOUT_UN MSK | 0x0 | This bit is used to unmask "read FIFO wait time over" interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 6 | R/W | SDIO_ABORT_ UNMSK | 0x0 | This bit is used to unmask abort interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 5 | R/W | SDIO_CRC_ERR_ UNMSK | 0x0 | This bit is used to unmask CRC error interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 4 | R/W | SDIO_PWRLEV_ UNMSK | 0x0 | This bit is used to unmask power level change interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 3 | R/W | SDIO_CMD52_UN MSK | 0x0 | This bit is used to unmask CMD52 interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 2 | R/W | SDIO_CSA_UNMS K | 0x0 | This bit is used to unmask CMD53 CSA interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 1 | R/W | SDIO_RD_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 read interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |
| 0 | R/W | SDIO_WR_INT_U NMSK | 0x0 | This bit is used to unmask CMD53 write interrupt. Setting this bit will unmask the interrupt Clearing this bit has no effect |

428 . SDIO Function5 Interrupt Unmask Register Description

SDIO_ERROR_COND_CTRL_ENABLE_REGISTER

| Bit | Access | Function | POR Value | Description |
|--------|--------|-------------------------------|-----------|--|
| [31:3] | R | Reserved | 0x0 | Reserved for future use. |
| 2 | R/W | SDIO_SPI_RD_D ATA_ERROR_EN | 0x0 | When set, stops the DMA from doing data accesses till read data error interrupt is cleared in SPI mode |
| 1 | R/W | SDIO_ABORT_EN | 0x0 | When set, stops the DMA from doing data accesses till ABORT interrupt is cleared |
| 0 | R/W | SDIO_CRC_EN | 0x0 | When set, stops the DMA from doing data accesses till CRC error interrupt is cleared |

429 . SDIO Error Condition Enable Register Description

SDIO_ERROR_COND_BLK_CNT

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------------------|-----------|---|
| [31:23] | R | Reserved | 0x0 | Reserved for future use. |
| [22:16] | R/W | SDIO_ERROR_BL_K_CNT | 0x0 | Indicates block count when one of error condition occurred |
| [15:12] | R | Reserved | 0x0 | Reserved for future use. |
| [11:0] | R/W | SDIO_ERROR_BY_TE_CNT | 0x0 | Indicates byte count when one of the error condition occurred |

430 . SDIO Error Condition State Register Description

SDIO_BOOT_CONFIG_VALS_0

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|---|
| 31:8 | R | csa_msbyte | 0xff8000 | MS byre of CSA address. Lower 24 bits of CSA will come through SDIO CSA registers. Whenever CSA access is done, 32-bit address will be prepared using these fields. |
| 7:0 | R | ocr_r | 0x09 | Operating conditions. The value written by bootloader can be read here. |

431 . SDIO Boot Config Values Register 0 Description

SDIO_BOOT_CONFIG_VALS_1

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|--|
| 31:8 | NA | Reserved | 0x0 | Reserved |
| 7 | R | ignore_disable_hs | 0x0 | if ignore_disable_hs is set, sdmem_disable_high_speed_switching coming from combo mode module is ignored |
| 6 | R | sdmem_disable_interrupt_mb_read | 0x0 | When set, interrupt will be not be driven during sd memory mb read transfer |
| 5 | R | sdmem_drive_hi_z_mb_read | 0x0 | When set, High will be driven in the second cycle of interrupt period during sd memory mb read transfer |
| 4 | R | sdmem_ignore_sdmem_present | 0x0 | When set, sdmem_present signal, coming from GPIO, will be ignored. |
| 3 | R | combocard | 0x0 | When set, combo mode will be enabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|---|
| 2:0 | R | no_of_io_functions | 0x1 | Indicates number functions supported. The value written by bootloader can be read here. |

432 . SDIO Boot Config Values Register 1 Description

non-I/O aware initialization

Multiple block write command shall be used rather than continuous single write command to make faster write operation.

Hardware Reference Manual

15.5 SMIHC

15.5.1 General Description

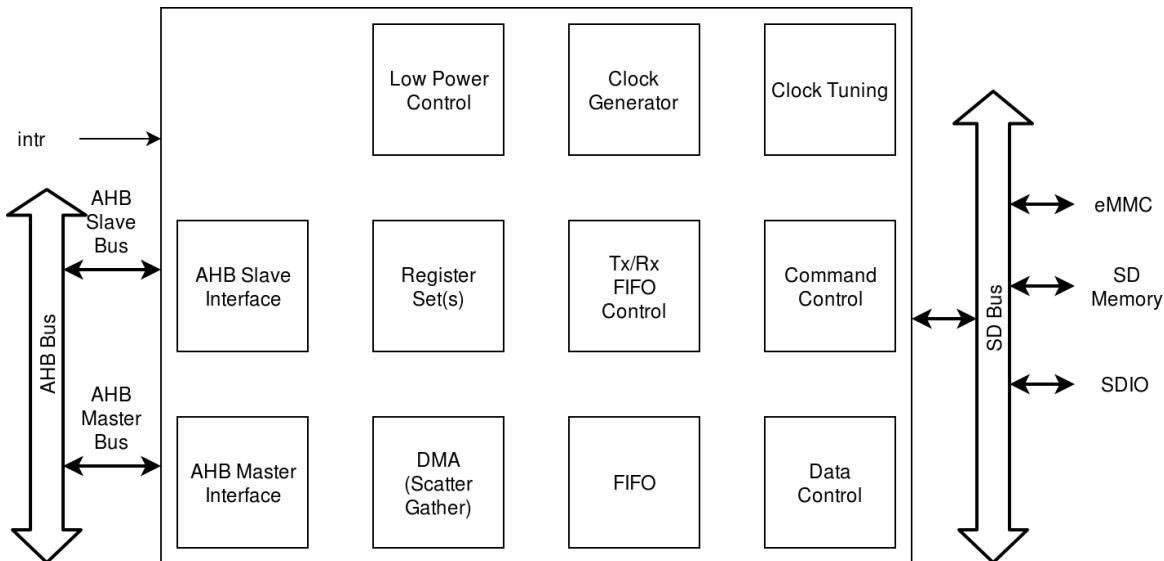
The SD/SDIO/MMC Host Controller (SMIHC) is compatible with Standard SD Host 3.0 and eMMC5.0 Specification. The SMIHC supports three key interfaces (SD memory, SDIO and eMMC). This supports Programmed IO mode (PIO), Simple DMA (SDMA) and powerful Scatter Gather DMA (ADMA) for data transfers. Using SMIHC, the Host driver can access memory up to 2TB.

15.5.2 Features

- Spec Compliance:
 - Compliant to SD Host Controller Standard Specification Version 3.0
 - Compliant to SD 3.0 Physical Layer Specification Version 3.01
 - Compliant to SDIO Specification Version 3.0
 - Compliant to JEDEC JESD84-B50 eMMC 5.0 Specification
- SD3.0 Supported Features
 - UHSI modes are supported - SDR50/DDR50
 - Clock Tuning
 - All Slot types are supported (Removable / Embedded Slot / Shared Bus Slot)
 - Asynchronous Interrupt Support
 - Programmable Clock Generator
 - SDSC/SDHC/SDXC/SDHS Cards
- MMCA 5.0 Supported Features
 - Supports Extended Security Commands
 - Supports eMMC mode.
 - Supports Tuning Sequence
 - Data Tag Mechanism and Packet Commands support for eMMC.
 - Supports regular Boot Operation mode
 - Supports Alternate Boot Operation Mode.
 - Access to Boot/RPMB/General purpose partition.
 - Supprts Double Data Rate (DDR) mode – 4bit
 - MMC plus and MMC mobile cards
- General Supported Features
 - Suspend/Resume
 - Read Wait
 - Tap Delay Circuit
 - Backward Compatible
 - SD/MMC Bus Width are 1,4 and 8bits

- Supports Internal DMA
- Supports generation of interrupt for different events
- Low Power Features
 - Entire module can run on Low Power Clock, during power saving mode
 - Wakeup Events for Card Insertion, Card Removal and Card Interrupt
 - SD Clock gating

15.5.3 Functional Description



SMIHC block diagram

The SMIHC has two bus interfaces, the System Bus Interface and the SD/MMC Bus Interface. The SMIHC assumes that these interfaces are Asynchronous. The SD/MMC card is on SD/MMC Bus time (that is, its operation is synchronized by SDCLK). The Host Controller synchronizes signals to communicate between these interfaces. The SD3.0 Standard Register set(s) are accessed via AHB Slave Interface. The SMIHC supports Programmed IO mode via AHB Slave interface, Simple DMA and ADMA (Scatter Gather) through AHB Master Interface.

Blocks of data are synchronized at the buffer module. All status registers are synchronized by the system clock and maintain synchronization during output to the system interface. Control registers, which trigger SD Bus transactions, are synchronized by SDCLK. Therefore, there will be a timing delay when propagating signals between the two interfaces. This means the Host Driver cannot do real time control of the SD Bus and needs to rely on the Host Controller to control the SD Bus according to register settings.

Reset Control module generates both Hardware (Power on Reset) and Software Reset (CMD, DAT, ALL) on corresponding clock domain. The Transfer Complete interrupt status indicates completion of the read / writes transfer for both DMA and non DMA transfers. However, the timing is different between reads and writes. Read transfers shall be completed after all valid data have been transferred to the Host System and are ready for the Host Driver to access. Write transfers shall be completed after all valid data have been transferred to the SD card and the busy state is over.

eMMC Boot Mode Operation

The SMIH Controller supports two types of Boot Mode Operation:

- Boot Operation
- Alternate Boot Operation

Additional Register Fields to support Boot Mode / Alternate Boot Mode Operation

The following bits are added to support both Boot Mode and Alternate Boot Mode Operation:

Command Register - Bit15 for Boot Operation:

Host Driver should set this bit only for Boot Operation. For Normal Transaction, this bit should always be zero.

1 – Start Boot Operation.

0 – Stop Boot Operation.

Whenever this bit is set to 1, the SMIH Controller pulls the command line low and wait for the boot data from the Card. The SMIH Controller uses push-pull mode until boot operation is terminated (SMIH Controller drive SMIH_OD_PP to 1, during push -pull mode). Host Driver set this bit to zero, to terminate the Boot Operation

Command Register - Bit14 for Alternate Boot Operation:

Host Driver should set this bit only for Alternate Boot Operation. For Normal Transaction, this bit should always be zero.

1 – Start Alternate Boot Operation.

0 – Stop Alternate Boot Operation.

Whenever this bit is set to 1, the SMIH Controller issues CMD0 with Argument 0xFFFF_FFFA. The Host Driver should take care of the 74 Clock cycle period. The SMIH Controller will issue this CMD0 as soon as the Host Driver sets this bit to 1 in Command Register. The SMIH Controller uses push-pull mode until boot operation is terminated (-SMIH Controller drive SMIH_OD_PP to 1, during push -pull mode). Host Driver set this bit to zero, to terminate the Alternate Boot Operation.

Normal Interrupt Status Register - Bit14 for Boot Done Interrupt (RW1C):

1 – Boot Mode / Alternate Boot Mode Operation is Done

0 – Boot Mode / Alternate Boot Mode Operation is in Progress

Normal Interrupt Status Register - Bit13 – Boot Ack Complete Interrupt (RW1C):

1 – Boot ACK is received successfully from the Card

0 – Boot ACK is not received.

Normal Interrupt Status Enable Register - Bit14 – Boot Done Interrupt Status Enable (RW):

1 – Enabled

0 – Masked

Normal Interrupt Status Enable Register - Bit13 – Boot Ack Complete Interrupt Status Enable (RW):

1 – Enabled

0 – Masked

Normal Interrupt Signal Enable Register - Bit14 – Boot Done Interrupt Signal Enable (RW):

1 – Enabled

0 – Masked

Normal Interrupt Signal Enable Register - Bit13 – Boot Ack Complete Interrupt Signal Enable (RW):

1 – Enabled

0 – Masked

Boot Operation using SDMA

Boot operation is explained in below table16-15.

| Steps | ROM/SRAM code | SMIHC | eMMC card |
|-------|---|-------|-----------|
| 1 | Wr Sys_Addr Reg * (System Address for DMA Opn) | | |
| 2 | Wr Block Reg *. (Block Size, Block Count – Boot Data) | | |

| Steps | ROM/SRAM code | SMIHC | eMMC card |
|-------|--|---|---------------------|
| 3 | Wr Arg Reg * (SD command Argument) | | |
| 4 | Wr Txfr_Cmd_Reg * (SD Command Control) | | |
| 5 | | If (boot) Pull CMD line Low Else Send CMD0-0xFFFFFFF | |
| 6 | | Wait for Ack or Timeout | Send Ack (Optional) |
| 7 | | If (Ack) Set Boot Ack Rcvd (INT) Else Set Data Timeout (INT) | |
| 8 | Poll for Ack or Timeout | | Send Data |
| 9 | | Wait for Boot Data | |
| 10 | Poll for Timeout or Boot Done | Block of Data Rcvd | |
| 11 | | Decrement Block Count Transfer data to System memory (Internal DMA) If (blk_cnt == 0) { Set boot_done (INT) Stop clock to the card Goto Step 12 } Else Goto step 9 | |
| 12 | More data to read. Set Block Count | If (wr2blkcnt reg && (blk_cnt == 0)) Goto Step 13 Else Goto Step 9 | |
| 13 | Clear Boot bit | If (boot mode) Pull command line high Else // Alt boot CMD0 - Arg 0x00000000 | |
| 14 | | | Stop Sending Data |

433 .Boot Operation using SDMA

Clock Tuning

Consecutive Sampling – Bits[13-8] of Host Control2 register:

This field denotes the minimum number of consecutive passing sampling points.

Example scenarios

Tuning Sequence:

1. The SMIH Controller Resets Sampling Control Block (at the start of every Re-Tuning Procedure).
2. The HC issues the Send Tuning Block command to read tuning block
3. The cards sends Tuning block as read data. The HC receives it and compares with a known tuning block pattern
4. The HC increments the Sampling Control Block by one step.
5. The HC sends a read command for the next tuning block.

Repeat Steps 3 to 5 above to cover full UI.

Consecutive Sampling field is set to 6'h 5 – Minimum 5 passing sampling points

- 1 Read Tuning Block - fail
- 2 Read Tuning Block - fail
- 3 Read Tuning Block - pass
- 4 Read Tuning Block - pass

5 Read Tuning Block – pass – **Here only 3 consecutive passes. So HC will ignore this sampling point**

6 Read Tuning Block - fail

7 Read Tuning Block - fail

8 Read Tuning Block - fail

9 Read Tuning Block - fail

10 Read Tuning Block - fail

11 Read Tuning Block - fail

12 Read Tuning Block - fail

13 Read Tuning Block - pass

14 Read Tuning Block - pass

15 Read Tuning Block – pass

16 Read Tuning Block - pass

17 Read Tuning Block – pass

18 Read Tuning Block - pass

19 Read Tuning Block - pass

20 Read Tuning Block - pass

21 Read Tuning Block - pass

22 Read Tuning Block - fail

No need to issue 23rd Tuning command at this point. Because the HC already got 5 consecutive passing sampling points (13 – 21). The HC will take the median of (13 and 21). i.e. $(13+21)/2 = 17$. So HC uses the 17th command tap delay as the correct sampling point for further data transactions.

Sequences

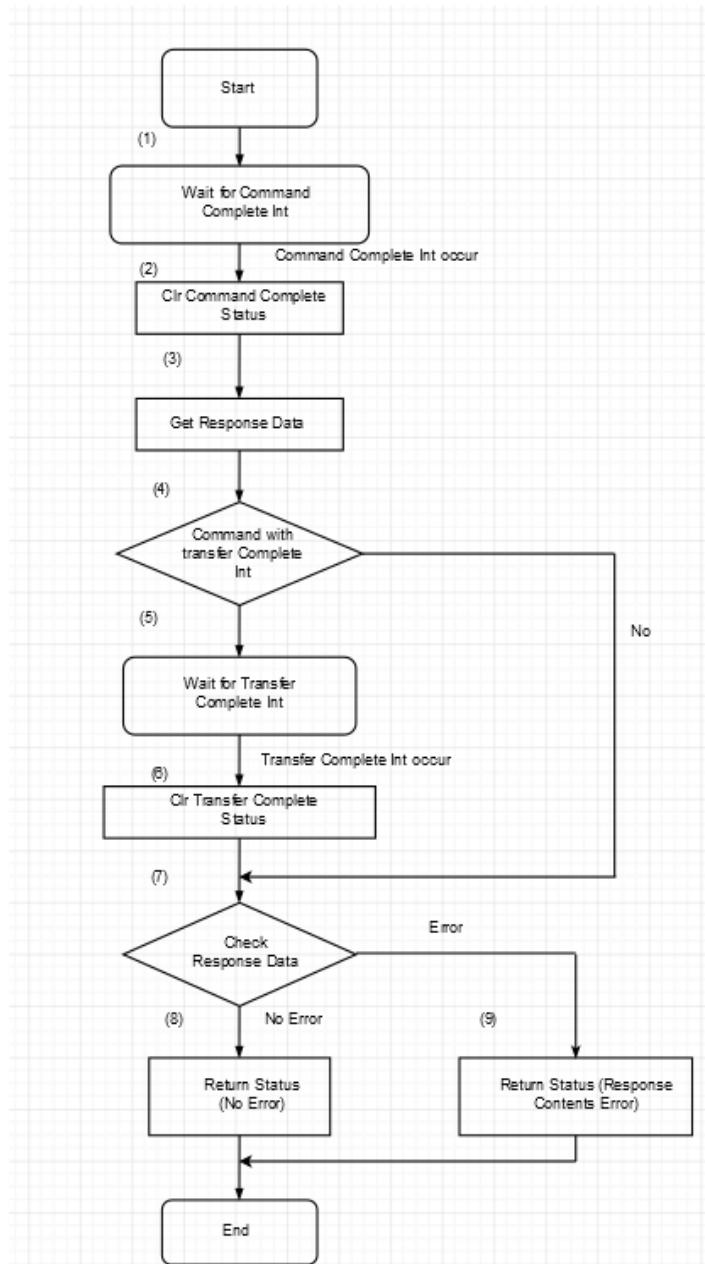
SD Command Sequence

- (1) Check Command Inhibit (CMD) in the Present State register. Repeat this step until Command Inhibit (CMD) is 0.
- (2) If the Host Driver issues a SD Command with busy signal, go to step (3). If without busy signal, go to step (5).
- (3) If the Host Driver issues an abort command, go to step (5). In the case of no abort command, go to step (4).
- (4) Check Command Inhibit (DAT) in the Present State register. Repeat this step until Command Inhibit (DAT) is set to 0.
- (5) Set the value of command argument to the Argument 1 register.
- (6) Set the Command register.

Note: Writing the upper byte [3] in the Command register causes the SMIH Controller to issue a SD command to the SD card.

- (7) Perform Command Completion Sequence

SD Command Completion Sequence



15 . SD Command Complete Sequence Flow chart

- (1) Wait for the Command Complete Interrupt. If the Command Complete Interrupt has occurred, go to step (2).
- (2) Write 1 to Command Complete in the Normal Interrupt Status register to clear this bit.
- (3) Read the Response register and get necessary information of the issued command.
- (4) Judge whether the command uses the Transfer Complete Interrupt or not. If it uses Transfer Complete, go to step (5). If not, go to step (7)
- (5) Wait for the Transfer Complete Interrupt. If the Transfer Complete Interrupt has occurred, go to step (6).

- (6) Write 1 to Transfer Complete in the Normal Interrupt Status register to clear this bit.
- (7) Check for errors in Response Data. If there is no error, go to step (8). If there is an error, go to step (9).
- (8) Return Status of "No Error".
- (9) Return Status of "Response Contents Error".



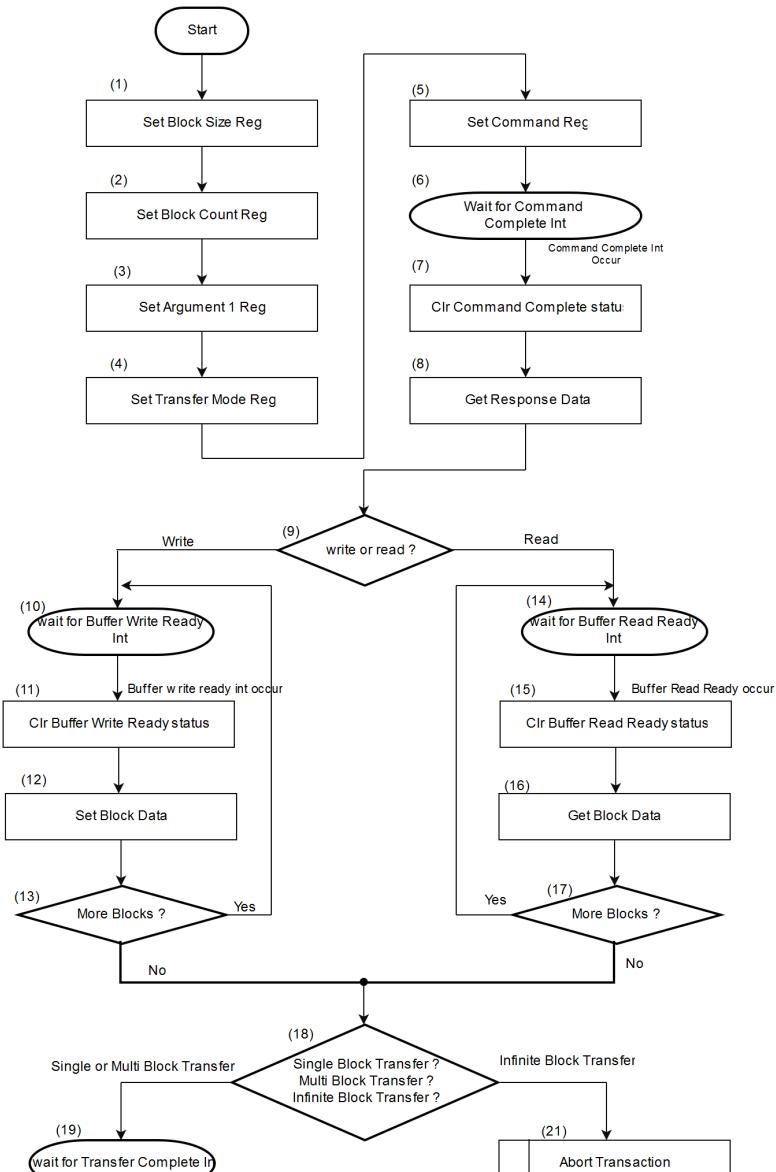
Note

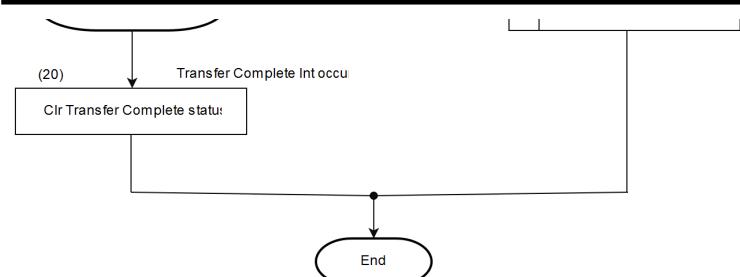
Note1: While waiting for the Transfer Complete interrupt, the Host Driver should only issue commands that do not use the busy signal.

Note2: The Host Driver should judge the Auto CMD12 complete by monitoring Transfer Complete.

Note3: When the last block of un-protected area is read using memory multiple block read command (CMD18), OUT_OF_RANGE error may occur even if the sequence is correct. The Host Driver should ignore it. This error will appear in the response of Auto CMD12 or in the response of the next memory command.

Data Transfer using PIO Mode





Transaction Control with Data Transfer Using DAT Line Sequence

Data transfer using PIO Mode Flow chart

- (1) Set the value corresponding to the executed data byte length of one block to Block Size register
- (2) Set the value corresponding to the executed data block count to Block Count register
- (3) Set the argument value to Argument 1 register.
- (4) Set the value to the Transfer Mode register. The host driver determines Multi / Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable.
- (5) Set the value to Command register.



Note

When writing the upper byte [3] of Command register, SD command is issued.

- (6) Then, wait for the Command Complete Interrupt.
- (7) Write 1 to the Command Complete in the Normal Interrupt Status register for clearing this bit.
- (8) Read Response register and get necessary information of the issued command.
- (9) In the case where this sequence is for write to a card, go to step (10). In case of read from a card, go to step (14).
- (10) Then wait for Buffer Write Ready Interrupt.
- (11) Write 1 to the Buffer Write Ready in the Normal Interrupt Status register for clearing this bit.
- (12) Write block data (in according to the number of bytes specified at the step (1)) to Buffer Data Port register.
- (13) Repeat until all blocks are sent and then go to step (18).
- (14) Then wait for the Buffer Read Ready Interrupt.
- (15) Write 1 to the Buffer Read Ready in the Normal Interrupt Status register for clearing this bit.
- (16) Read block data (in according to the number of bytes specified at the step (1)) from the Buffer Data Port register.
- (17) Repeat until all blocks are received and then go to step (18).
- (18) If this sequence is for Single or Multiple Block Transfer, go to step (19). In case of Infinite Block Transfer, go to step (21).
- (19) Wait for Transfer Complete Interrupt.
- (20) Write 1 to the Transfer Complete in the Normal Interrupt Status register for clearing this bit.
- (21) Perform the sequence for Abort Transaction in accordance with Section 3.8.



Note

Step (1) and Step (2) can be executed at same time. Step (4) and Step (5) can be executed at same time.

Data Transfer Using SDMA Mode

- (1) Data location of system memory is set to the SDMA System Address register.
- (2) Set the value corresponding to the executed data byte length of one block in the Block Size register.
- (3) Set the value corresponding to the executed data block count in the Block Count register
- (4) Set the argument value to the Argument 1 register.
- (5) Set the value to the Transfer Mode register. The host driver determines Multi / Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable.
- (6) Set the value to the Command register.



Note

When writing to the upper byte [3] of the Command register, the SD command is issued and SDMA is started.

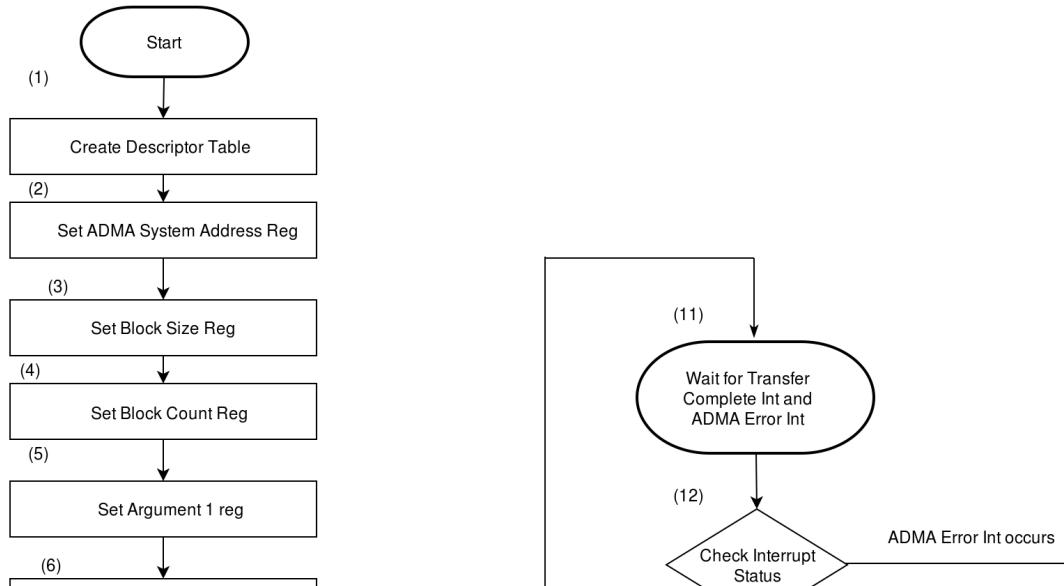
- (7) Then wait for the Command Complete Interrupt.
- (8) Write 1 to the Command Complete in the Normal Interrupt Status register to clear this bit.
- (9) Read Response register and get necessary information of the issued command.
- (10) Wait for the Transfer Complete Interrupt and DMA Interrupt.
- (11) If Transfer Complete is set 1, go to Step (14) else if DMA Interrupt is set to 1, go to Step (12)
- (12) Transfer Complete is higher priority than DMA Interrupt.
- (12) Write 1 to the DMA Interrupt in the Normal Interrupt Status register to clear this bit.
- (13) Set the next system address of the next data position to the System Address register and go to Step (10).
- (14) Write 1 to the Transfer Complete and DMA Interrupt in the Normal Interrupt Status register to clear this bit.

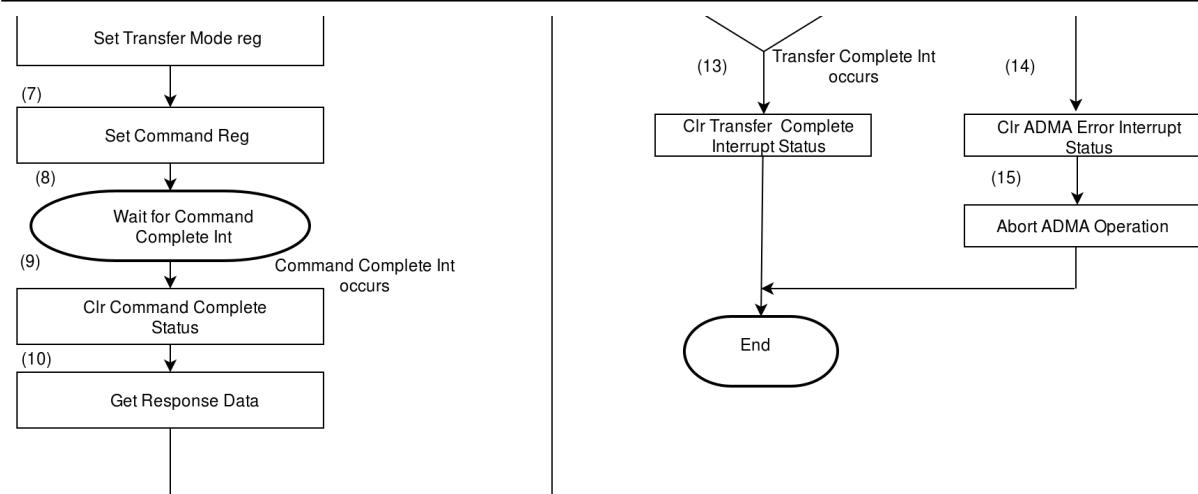


Note

Step (2) and Step (3) can be executed simultaneously. Step (5) and Step (6) can also be executed simultaneously.

Data Transfer Using ADMA Mode





Transaction Control with Data Transfer Using DAT Line Sequence (Using ADMA)

Data transfer using ADMA Mode Flow chart

- (1) Create Descriptor table for ADMA in the system memory
- (2) Set the Descriptor address for ADMA in the ADMA System Address register.
- (3) Set the value corresponding to the executed data byte length of one block in the Block Size register.
- (4) Set the value corresponding to the executed data block count in the Block Count register. If the Block Count Enable in the Transfer Mode register is set to 1, total data length can be designated by the Block Count register and the Descriptor Table. These two parameters shall indicate same data length. However, transfer length is limited by the Block Count register. If the Block Count Enable in the Transfer Mode register is set to 0, total data length is designated by not Block Count register but the DescriptorTable. In this case, ADMA reads more data than length programmed in descriptor from SD card. Too much read operation is aborted asynchronously and extra read data is discarded when the ADMA is completed
- (5) Set the argument value to the Argument 1 register.
- (6) Set the value to the Transfer Mode register. The host driver determines Multi / Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable.
- (7) Set the value to the Command register.



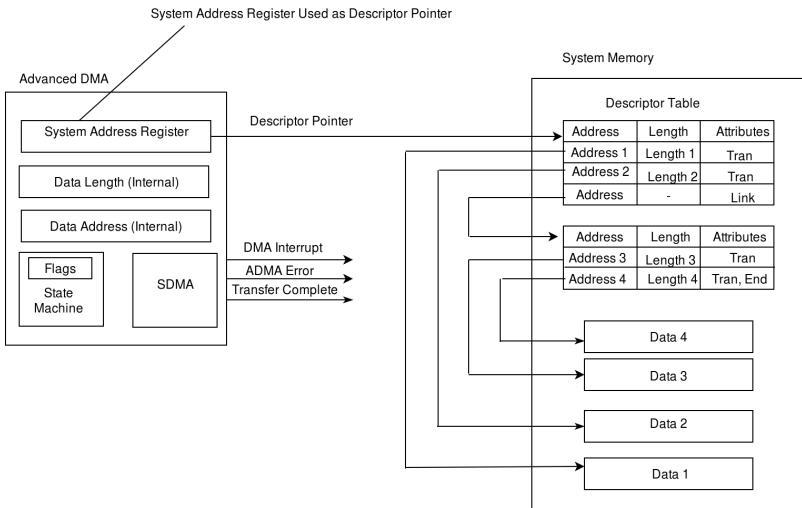
Note

When writing to the upper byte [3] of the Command register, the SD command is issued and DMA is started.

- (8) Then wait for the Command Complete Interrupt.
- (9) Write 1 to the Command Complete in the Normal Interrupt Status register to clear this bit.
- (10) Read Response register and get necessary information of the issued command.
- (11) Wait for the Transfer Complete Interrupt and ADMA Error Interrupt.
- (12) If Transfer Complete is set 1, go to Step (13) else if ADMA Error Interrupt is set to 1, go to Step (14).
- (13) Write 1 to the Transfer Complete Status in the Normal Interrupt Status register to clear this bit.
- (14) Write 1 to the ADMA Error Interrupt Status in the Error Interrupt Status register to clear this bit.
- (15) Abort ADMA operation. SD card operation should be stopped by issuing abort command. If necessary, the host driver checks ADMA Error Status register to detect why ADMA error is generated.

Note

Step (3) and Step (4) can be executed simultaneously. Step (6) and Step (7) can also be executed simultaneously.

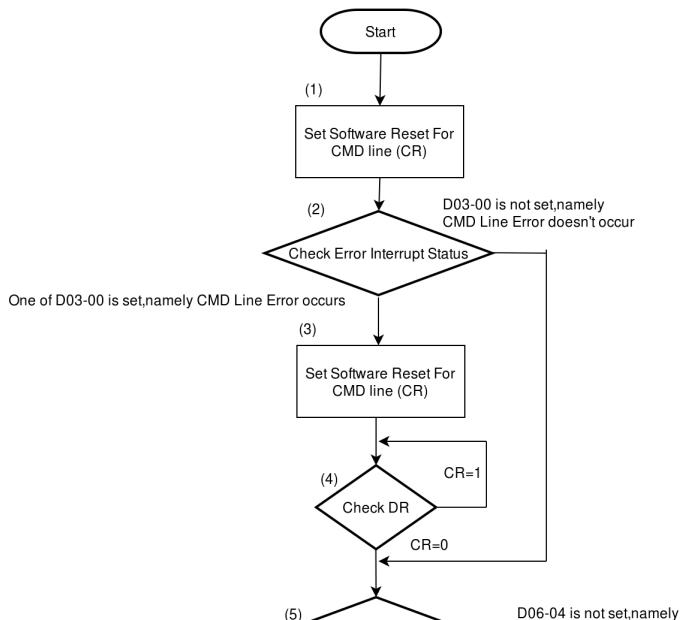


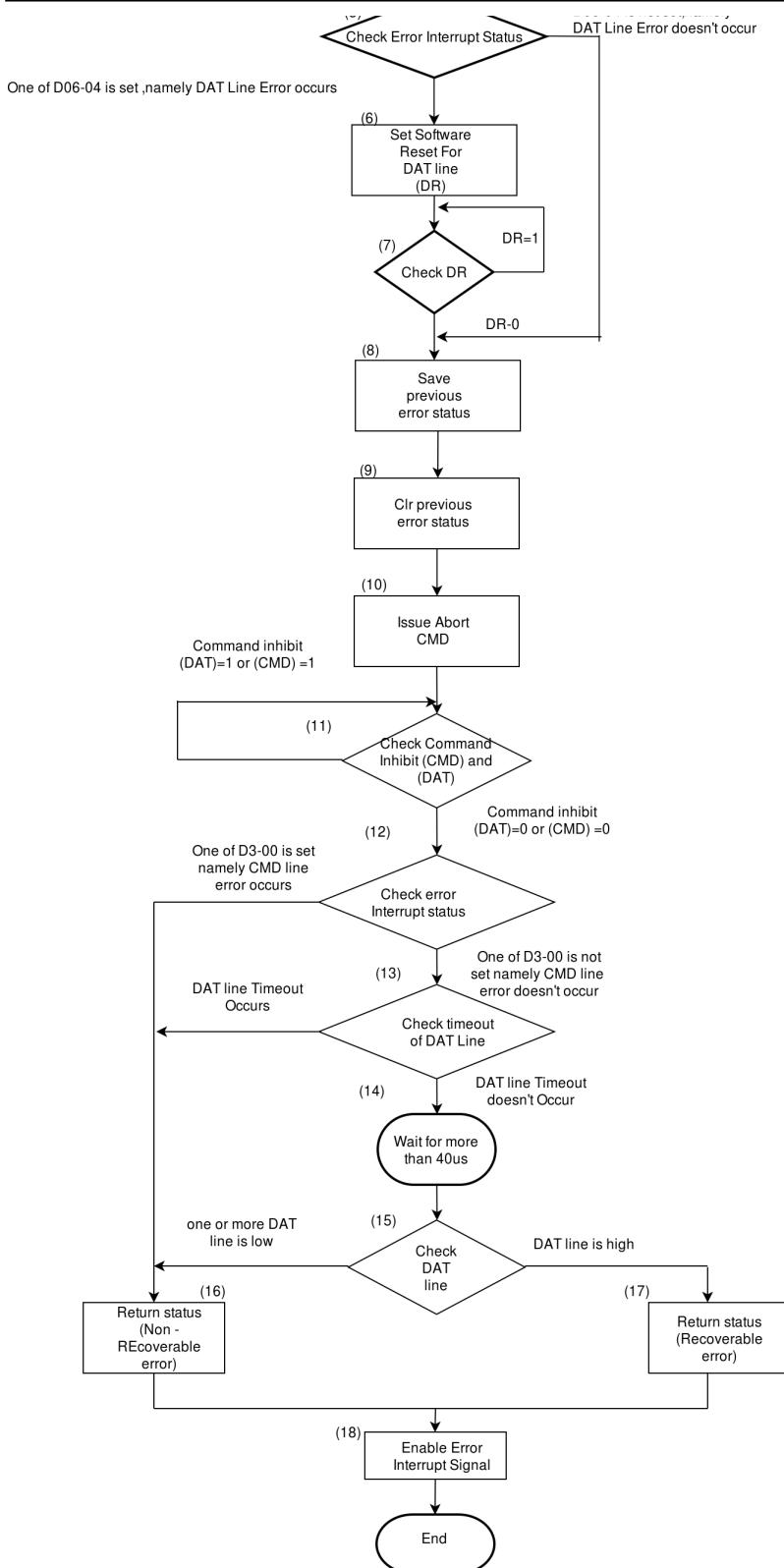
Block Diagram of ADMA2

Block Diagram of ADMA2

The above figure shows block diagram of ADMA2. The Descriptor Table is created in system memory by the Host Driver. 32-bit Address Descriptor Table is used for the system with 32-bit addressing and 64-bit Address Descriptor Table is used for the system with 64-bit addressing. Each descriptor line (one executable unit) consists with address, length and attribute field. The attribute specifies operation of the descriptor line. ADMA2 includes SDMA, State Machine and Registers circuits. ADMA2 does not use 32-bit DMA System Address Register (offset 0) but uses the 64-bit Advanced DMA System Address register (offset 058h) for descriptor pointer. Writing Command register triggers off ADMA2 transfer. ADMA2 fetches one descriptor line and execute it. This procedure is repeated until end of descriptor is found (End=1 in attribute)

Error Interrupt Recovery Sequence





Error Interrupt Recovery Sequence

Error Interrupt Recovery Sequence Flow chart

- (1) Disable the Error Interrupt Signal.
- (2) Check bits D03-00 in the Error Interrupt Status register. If one of these bits (D03-00) is set to 1, go to step (3). If none are set to 1 (all are 0), go to step (5).
- (3) Set Software Reset For CMD Line to 1 in the Software Reset register for software reset of the CMD line.
- (4) Check Software Reset For CMD Line in the Software Reset register. If Software Reset For CMD Line is 0, go to step (5). If it is 1, go to step (4).
- (5) Check bits D06-04 in the Error Interrupt Status register. If one of these bits (D06-04) is set to 1, go to step (6). If none are set to 1 (all are 0), go to step (8).
- (6) Set Software Reset For DAT Line to 1 in the Software Reset register for software reset of the DAT line.
- (7) Check Software Reset For DAT Line in the Software Reset register. If Software Reset For DAT Line is 0, go to step (8). If it is 1, go to step (7).
- (8) Save previous error status.
- (9) Clear previous error status with setting them to 1.
- (10) Issue Abort Command.
- (11) Check Command Inhibit (DAT) and Command Inhibit (CMD) in the Present State register. Repeat this step until both Command Inhibit (DAT) and Command Inhibit (CMD) are set to 0.
- (12) Check bits D03-00 in the Error Interrupt Status register for Abort Command. If one of these bits is set to 1, go to step (16). If none of these bits are set to 1 (all are 0), go to step (13).
- (13) Check Data Timeout Error in the Error Interrupt Status register. If this bit is set to 1, go to step (16). If it is 0, go to step (14).
- (14) Wait for more than 40 us.
- (15) By monitoring the DAT [3:0] Line Signal Level in the Present State register, judge whether the level of the DAT line is low or not. If one or more DAT lines are low, go to step (16). If the DAT lines are high, go to step (17).
- (16) Return Status of "Non-recoverable Error".
- (17) Return Status of "Recoverable Error".
- (18) Enable the Error Interrupt Signal.

15.5.4 Register Summary

Base Address: 0x2022_0000

| Register Name | Offset | Description |
|-----------------------------------|--------|------------------------------|
| SMIH SDMA System Address Register | 0x0 | SDMA System Address Register |
| SMIH Block Size Register | 0x4 | Block Size Register |
| SMIH Block Count Register | 0x6 | Block Count Register |
| SMIH Argument 1 Register | 0x8 | Argument 1 Register |
| SMIH Transfer Mode Register | 0xC | Transfer Mode Register |
| SMIH Command Register | 0XE | Command Register |
| SMIH Response Register | 0x10 | Response Register |
| SMIH Buffer Data Port Register | 0x20 | Buffer Data Port Register |
| SMIH Present State Register | 0x24 | Present State Register |
| SMIH Host Control 1 Register | 0x28 | Host Control 1 Register |
| SMIH Power Control Register | 0x29 | Power Control Register |
| SMIH Block Gap Control Register | 0x2A | Block Gap Control Register |
| SMIH Wakeup Control Register | 0x2B | Wake up Control Register |

| Register Name | Offset | Description |
|--|-----------|---|
| SMIH Clock Control Register | 0x2C | Clock Control Register |
| SMIH Timeout Control Register | 0x2E | Timeout Control Register |
| SMIH Software Reset Register | 0x2F | Software Reset Register |
| SMIH Normal Interrupt Status Register | 0x30 | Normal Interrupt Status Register |
| SMIH Error Interrupt Status Register | 0x32 | Error Interrupt Status Register |
| SMIH Normal Interrupt Status Enable Register | 0x34 | Normal Interrupt Status Enable Register |
| SMIH Error Interrupt Status Enable Register | 0x36 | Error Interrupt Status Enable Register |
| SMIH Normal Interrupt Signal Enable Register | 0x38 | Normal Interrupt Signal Enable Register |
| SMIH Error Interrupt Signal Enable Register | 0x3A | Error Interrupt Signal Enable Register |
| SMIH Auto CMD Error Status Register | 0x3C | Auto CMD Error Status Register |
| SMIH Host Control 2 Register | 0x3E | Host Control 2 Register |
| SMIH Capabilities Register | 0x40 | Capabilities Register |
| SMIH Maximum Current Capabilities Register | 0x48 | Maximum Current Capabilities Register |
| Reserved | 0x50-0x52 | |
| SMIH ADMA Error Status Register | 0x54 | ADMA Error Status Register |
| SMIH ADMA System Address Register | 0x58 | ADMA System Address Register |
| SMIH Preset Value Register0 | 0x60 | Preset Value for Initialization(3.3V or 1.8V) |
| SMIH Preset Value Register1 | 0x62 | Preset Value for Default Speed(3.3V) |
| SMIH Preset Value Register2 | 0x64 | Preset Value for High Speed(3.3V) |
| SMIH Preset Value Register3 | 0x66 | Preset Value for SDR12(1.8V) |
| SMIH Preset Value Register4 | 0x68 | Preset Value for SDR25(1.8V) |
| SMIH Preset Value Register5 | 0x6A | Preset Value for SDR50(1.8V) |
| SMIH Preset Value Register6 | 0x6C | Preset Value for SDR104(1.8V) |
| SMIH Preset Value Register7 | 0x6E | Preset Value for DDR50(1.8V) |
| Reserved | 0x70 | Reserved |
| SMIH Tx Tune Register | 0xE0 | Tx Tune Register |
| SMIH Rx Tune Register | 0xE4 | Rx Tune Register |
| SMIH Ds Tune Register | 0xE8 | Ds Tune Register |
| SMIH AHB Burst Size Register | 0xEC | AHB Burst Size Register |
| SMIH SDH Revision ID Register | 0xF4 | SDH Revision ID Register |
| SMIH Slot Interrupt Status Register | 0xFC | Slot Interrupt Status Register |
| SMIH Host Controller Version Register | 0xFE | Host Controller Version Register |

434 . Register Summary

15.5.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write

SMIH SDMA System Address / Argument 2 Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------------------|-----------|--|
| 31:0 | R/W | SDMA System Address / Argument 2 | 0 | <p>This register contains the physical system memory address used for DMA transfers or the second argument for the Auto CMD23.</p> <p>(1) SDMA System Address</p> <p>This register contains the system memory address for a SDMA transfer.</p> <p>When the Host Controller stops a SDMA transfer, this register shall point to the system address of the next contiguous data position.</p> <p>It can be accessed only if no transaction is executing (i.e., after a transaction has stopped). Read operations during transfers may return an invalid value.</p> <p>The Host Driver shall initialize this register before starting a SDMA transaction. After SDMA has stopped, the next system address of the next contiguous data position can be read from this register. The SDMA transfer waits at the every boundary specified by the Host SDMA Buffer Boundary in the Block Size register.</p> <p>The Host Controller generates DMA Interrupt to request the Host Driver to update this register. The Host Driver sets the next system address of the next data position to this register. When the most upper byte of this register (0x003) is written, the Host Controller restarts the SDMA transfer.</p> <p>When restarting SDMA by the Resume command or by setting Continue Request in the Block Gap Control register, the Host Controller shall start at the next contiguous address stored here in the SDMA System Address register. ADMA does not use this register.</p> <p>(2) Argument 2</p> <p>This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23.</p> <p>If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| | | | | If Auto CMD23 is used without AMDA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case. |

435 . SDMA System Address / Argument 2 Description

SMIH Block Size Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|-------------|
| 15 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 14:12 | R/W | Host SDMA Buffer Boundary | 0 | <p>The large contiguous memory space may not be available in the virtual memory system. To perform a long SDMA transfer, SDMA System Address register shall be updated at every system memory boundary during SDMA transfer. These bits specify the size of the contiguous buffer in the system memory.</p> <p>The SDMA transfer shall wait at every boundary specified by these fields and the SMIHC Controller will generate the DMA Interrupt to request the Host Driver to update the SDMA System Address register. At the end of transfer, the SMIHC Controller may issue or may not issue DMA Interrupt.</p> <p>In particular, DMA Interrupt shall not be issued after Transfer Complete Interrupt is issued. If this register is set to 0 (buffer size = 4K bytes), the lower 12-bit of byte address points to data in the contiguous buffer and the upper 20-bit points to the location of the buffer in the system memory. The SDMA transfer stops when the SMIHC Controller detects carry out of the address from bit 11 to 12. These bits shall be supported when the SDMA Support in the Capabilities register is set to 1 and this function is active when the DMA Enable in the Transfer Mode register is set to 1. ADMA does not use this register.</p> <ul style="list-style-type: none"> 000b 4K bytes (Detects A11 carry out) 001b 8K bytes (Detects A12 carry out) 010b 16K Bytes (Detects A13 carry out) 011b 32K Bytes (Detects A14 carry out) 100b 64K bytes (Detects A15 carry out) 101b 128K Bytes (Detects A16 carry out) 110b 256K Bytes (Detects A17 carry out) 111b 512K Bytes (Detects A18 carry out) |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | | | |
|-----------------------|------------------|---------------------|-----------|--|------|------------|-----------------------|--|-----|-----------|-------------------|-----------|-----------------------|--|-----|---------|-------------------|---------|-------------------|---------|-------------------|--------|-------------------|------------------|
| 11:0 | R/W | Transfer Block Size | 0 | <p>This register specifies the block size of data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. Values ranging from 1 up to the maximum buffer size can be set. In case of memory, it shall be set up to 512 bytes . It can be accessed only if no transaction is executing (i.e., after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations shall be ignored.</p> <table border="1" style="margin-left: 20px;"> <tr><td>000b</td><td>2048 Bytes</td></tr> <tr><td>0800_h...</td><td></td></tr> <tr><td>...</td><td>512 Bytes</td></tr> <tr><td>0200_h</td><td>511 Bytes</td></tr> <tr><td>01FF_h...</td><td></td></tr> <tr><td>...</td><td>4 Bytes</td></tr> <tr><td>0004_h</td><td>3 Bytes</td></tr> <tr><td>0003_h</td><td>2 Bytes</td></tr> <tr><td>0002_h</td><td>1 Byte</td></tr> <tr><td>0001_h</td><td>No data transfer</td></tr> </table> | 000b | 2048 Bytes | 0800 _h ... | | ... | 512 Bytes | 0200 _h | 511 Bytes | 01FF _h ... | | ... | 4 Bytes | 0004 _h | 3 Bytes | 0003 _h | 2 Bytes | 0002 _h | 1 Byte | 0001 _h | No data transfer |
| 000b | 2048 Bytes | | | | | | | | | | | | | | | | | | | | | | | |
| 0800 _h ... | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | 512 Bytes | | | | | | | | | | | | | | | | | | | | | | | |
| 0200 _h | 511 Bytes | | | | | | | | | | | | | | | | | | | | | | | |
| 01FF _h ... | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | 4 Bytes | | | | | | | | | | | | | | | | | | | | | | | |
| 0004 _h | 3 Bytes | | | | | | | | | | | | | | | | | | | | | | | |
| 0003 _h | 2 Bytes | | | | | | | | | | | | | | | | | | | | | | | |
| 0002 _h | 1 Byte | | | | | | | | | | | | | | | | | | | | | | | |
| 0001 _h | No data transfer | | | | | | | | | | | | | | | | | | | | | | | |

436 . Block Size Register Description

SMIH Block Count Register

| Bit | Access | Function | POR Value | Description | | | | | | | | | | |
|-------------------|--------------|----------------------------------|-----------|---|-------------------|--------------|-----|-----|-------------------|----------|-------------------|---------|-------------------|------------|
| 15:0 | R/W | Block Count For Current Transfer | 0 | <p>This register is enabled when Block Count Enable in the Transfer Mode register is set to 1 and is valid only for multiple block transfers.</p> <p>The Host Driver shall set this register to a value between 1 and the maximum block count. The SMIHC Controller decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to 0 results in no data blocks being transferred.</p> <p>This register should be accessed only when no transaction is executing (i.e., after transactions are stopped). During data transfer, read operations on this register may return an invalid value and write operations are ignored.</p> <p>When a suspend command is completed, the number of blocks yet to be transferred can be determined by reading this register.</p> <p>Before issuing a resume command, the Host Driver shall restore the previously saved block count.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>FFFF_h</td><td>65535 blocks</td></tr> <tr><td>...</td><td>...</td></tr> <tr><td>0002_h</td><td>2 blocks</td></tr> <tr><td>0001_h</td><td>1 block</td></tr> <tr><td>0000_h</td><td>Stop Count</td></tr> </table> | FFFF _h | 65535 blocks | ... | ... | 0002 _h | 2 blocks | 0001 _h | 1 block | 0000 _h | Stop Count |
| FFFF _h | 65535 blocks | | | | | | | | | | | | | |
| ... | ... | | | | | | | | | | | | | |
| 0002 _h | 2 blocks | | | | | | | | | | | | | |
| 0001 _h | 1 block | | | | | | | | | | | | | |
| 0000 _h | Stop Count | | | | | | | | | | | | | |

437 . Block Count Register Description

SMIH Argument 1 Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------|-----------|---|
| 31:0 | R/W | Command Argument 0 1 | 0 | The SD command argument is specified as bit 39-8 of Command-Format in the Physical Layer Specification. |

438 . Argument 1 Register Description

SMIH Transfer Mode Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------------|-----------|---|
| 15 | R/W | Boot Operation | 0 | <p>Host Driver should set this bit only for Boot Operation. For Normal Transaction, this bit should always be zero.</p> <p>1 – Start Boot Operation. 0 – Stop Boot Operation.</p> <p>Whenever this bit is set to 1, the SMIHC Controller pulls the command line low and waits for the boot data from the Card.</p> <p>The SMIHC Controller uses push-pull mode until boot operation is terminated (SMIHC Controller drive SMIH_OD_PP to 1, during push -pull mode).</p> <p>Host Driver set this bit to zero, to terminate the Boot Operation. The SMIHC Controller also clear this bit, when the Boot Operation results in timeout.</p> |
| 14 | R/W | Alternate Boot Operation | 0 | <p>Host Driver should set this bit only for Alternate Boot Operation.</p> <p>For Normal Transaction, this bit should always be zero.</p> <p>Whenever this bit is set to 1, the SMIHC Controller issues CMD0 with Argument 0xFFFF_FFFA. The Host Driver should take care of the 74 Clock cycle period.</p> <p>The SMIHC Controller will issue this CMD0 as soon as the Host Driver sets this bit to 1 in Command Register. The SMIHC Controller uses push-pull mode until</p> <p>boot operation is terminated (SMIHC Controller drive SMIH_OD_PP to 1, during push -pull mode). Host Driver set this bit to zero, to terminate the Alternate Boot Operation. The SMIHC Controller also clear this bit, when the Boot Operation results in timeout.</p> |
| 13 | R/W | Boot Ack Enable | 0 | <p>1 – Card will send Boot Ack 0 – Card will not send Boot Ack</p> |
| 12 | R/W | SPI Mode Enable | 0 | <p>1 – Enable SPI Mode 0 – Disable SPI Mode</p> |
| 11 | R/W | Stream Mode Enable | 0 | <p>The Host driver has to set this bit for MMC CMD11 / CMD20 Stream Read/Write Operations.</p> <p>1 – Stream Mode is Enabled 0 – Stream Mode is Disabled</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------------------|-----------|--|
| 10 | R/W | MMC CMD23 | 0 | The Host driver has to set this bit for MMC CMD23 1 – CMD23 Format in MMC mode 0 – CMD23 Format in SD mode |
| 9:6 | R | Reserved | 0 | Reserved |
| 5 | R/W | Multi / Single Block Select | 0 | This bit is set when issuing multiple-block transfer commands using DAT line. For any other commands, this bit shall be set to 0. If this bit is 0, it is not necessary to set the Block Count register 1 – Multiple Block 0 – Single Block |
| 4 | R/W | Data Transfer Direction Select | 0 | This bit defines the direction of DAT line data transfers. The bit is set to 1 by the Host Driver to transfer data from the SD card to the SMIHC Controller and it is set to 0 for all other commands. 1 – Read (Card to Host) 0 – Write (Host to Card) |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|---|
| 3:2 | R/W | Auto CMD Enable | 0 | <p>This field determines use of auto command functions.</p> <p>00 – Auto Command Disabled</p> <p>01 – Auto CMD12 Enabled</p> <p>10 – Auto CMD23 Enabled</p> <p>11 – Reserved</p> <p>There are two methods to stop Multiple-block read and write operation.</p> <p>(1) Auto CMD12 Enable</p> <p>When this field is set to 01b, the SMIHC Controller issues CMD12 automatically when last block transfer is completed.</p> <p>Auto CMD12 error is indicated to the Auto CMD Error Status register. The Host Driver shall not set this bit if the command does not require CMD12.</p> <p>In particular, secure commands defined in the Part 3 File Security specification do not require CMD12.</p> <p>(2) Auto CMD23 Enable</p> <p>When this bit field is set to 10b, the SMIHC Controller issues a CMD23 automatically before issuing a command specified in the Command Register.</p> <p>The following conditions are required to use the Auto CMD23.</p> <ul style="list-style-type: none"> • A memory card that supports CMD23 (SCR[33]=1). • If DMA is used, it shall be ADMA. • Only when CMD18 or CMD25 is issued. (Note, the SMIHC Controller does not check command index.) <p>Auto CMD23 can be used with or without ADMA. By writing the Command register, the SMIHC Controller issues a CMD23 first and then issues a command specified by the Command Index in Command register. If response errors of CMD23 are detected, the second command is not issued.</p> <p>A CMD23 error is indicated in the Auto CMD Error Status register. 32-bit block count value for CMD23 is set to SDMA System Address / Argument 2 register.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|--|
| 1 | R/W | Block Count Enable | 0 | <p>This bit is used to enable the Block Count register, which is only relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled,</p> <p>which is useful in executing an infinite transfer. If ADMA2 data transfer is more than 65535 blocks, this bit shall be set to 0.</p> <p>In this case, data transfer length is designated by Descriptor Table.</p> <p>1 – Enable 0 – Disable</p> |
| 0 | R/W | DMA Enable | 0 | <p>This bit enables DMA functionality. DMA can be enabled only if it is supported as indicated in the Capabilities register.</p> <p>One of the DMA modes can be selected by DMA Select in the Host Control 1 register. If DMA is not supported, this bit is meaningless and shall always read 0.</p> <p>If this bit is set to 1, a DMA operation shall begin when the Host Driver writes to the upper byte of Command register (0x00F).</p> <p>1 – DMA Data transfer 0 – No data transfer or Non DMA data transfer</p> |

439 . Transfer Mode Register Description

SMIH Command Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------|-----------|--|
| 15:14 | R/W | Reserved | 0 | Reserved |
| 13:8 | R/W | Command Index | 0 | These bits shall be set to the command number (CMD0-63, ACMD0-63) that is specified in bits 45-40 of the Command-Format in the Physical Layer Specification and SDIO Card Specification. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------|-----------|--|
| 7:6 | R/W | Command Type | 0 | <p>There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00_b for all other commands.</p> <p>(1) Suspend Command</p> <p>If the Suspend command succeeds, the SMIHC Controller shall assume the SD Bus has been released and that it is possible to issue the next command, which uses the DAT line.</p> <p>The SMIHC Controller shall de-assert Read Wait for read transactions and stop checking busy for write transactions. The interrupt cycle shall start, in 4-bit mode. If the Suspend command fails, the SMIHC Controller shall maintain its current state, and the Host Driver shall restart the transfer by setting Continue Request in the Block Gap Control register.</p> <p>(2) Resume Command</p> <p>The Host Driver re-starts the data transfer by restoring the registers in the range of $000\text{-}00D_h$. The SMIHC Controller shall check for busy before starting write transfers.</p> <p>(3) Abort Command</p> <p>If this command is set when executing a read transfer, the SMIHC Controller shall stop reads to the buffer. If this command is set when executing a write transfer,</p> <p>the SMIHC Controller shall stop driving the DAT line. After issuing the Abort command, the Host Driver should issue a software reset.</p> <p>11_b – Abort CMD12, CMD52 for writing "I/O Abort" in CCCR</p> <p>10_b – Resume CMD52 for writing "Function Select" in CCCR</p> <p>01_b – Suspend CMD52 for writing "Bus Suspend" in CCCR</p> <p>00_b – Normal Other commands</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------|-----------|---|
| 5 | R/W | Data Present Select | 0 | <p>This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line.</p> <p>It is set to 0 for the following:</p> <ul style="list-style-type: none"> (1) Commands using only CMD line (ex. CMD52). (2) Commands with no data transfer but using busy signal on DAT[0] line (R1_b or R5_b ex. CMD38) (3) Resume command <p>1 – Data present</p> <p>0 – No Data present</p> |
| 4 | R/W | Command Index Check Enable | 0 | <p>If this bit is set to 1, the SMIHC Controller shall check the Index field in the response to see if it has the same value as the command index.</p> <p>If it is not, it is reported as a Command Index Error.</p> <p>If this bit is set to 0, the Index field is not checked.</p> <p>1 – Enable</p> <p>0 – Disable</p> |
| 3 | R/W | Command CRC Check Enable | 0 | <p>If this bit is set to 1, the SMIHC Controller shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error.</p> <p>If this bit is set to 0, the CRC field is not checked. The position of CRC field is determined according to the length of the response.</p> <p>1 – Enable</p> <p>0 – Disable</p> |
| 2 | R/W | Reserved | 0 | Reserved |
| 1:0 | R/W | Response Type Select | 0 | <p>00 – No Response</p> <p>01 – Response Length 136</p> <p>10 – Response Length 48</p> <p>11 – Response Length 48 check Busy after response</p> |

440 . Command Register Description

SMIH Response Register

| Bit | Access | Function | POR Value | Description | | | |
|-------|--------|------------------|-----------|---------------------------------------|------------------------------|----------------|-------------------|
| 127:0 | R | Command Response | 0 | Response Type | Meaning of Response | Response Field | Response Register |
| | | | | R1, R1 _b (normal response) | Card Status | R [39:8] | REP [31:0] |
| | | | | R1 _b (Auto CMD12 response) | Card Status for Auto CMD12 | R [39:8] | REP [127:96] |
| | | | | R1 (Auto CMD23 response) | Card Status for Auto CMD23 | R [39:8] | REP [127:96] |
| | | | | R2 (CID, CSD register) | CID or CSD reg. incl. | R [127:8] | REP [119:0] |
| | | | | R3 (OCR register) | OCR register for memory | R [39:8] | REP [31:0] |
| | | | | R4 (OCR register) | OCR register for I/O etc | R [39:8] | REP [31:0] |
| | | | | R5,R5 _b | SDIO response | R [39:8] | REP [31:0] |
| | | | | R6 (Published RCA response) | New published RCA[31:16] etc | R [39:8] | REP [31:0] |

441 . Response Register Description

SMIH Buffer Data Port Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------|-----------|---|
| 31:0 | R/W | Buffer Data | 0 | The SMIHC Controller buffer can be accessed through this 32-bit Data Port register. |

442 . Buffer Data Port Register Description

SMIH Present State Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|---|
| 31:25 | R/W | Reserved | 0 | Reserved |
| 24 | R | CMD Line Signal Level | 0 | This status is used to check the CMD line level to recover from errors and for debugging. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------------------|-----------|--|
| 23:20 | R | DAT[3:0] Line Signal Level | 0 | <p>This status is used to check the DAT line level to recover from errors and for debugging.</p> <p>This is especially useful in detecting the busy signal level from DAT [0].</p> <p>D23DAT[3] D22DAT[2] D21DAT[1] D20DAT[0]</p> |
| 19 | R | Write Protect Switch Pin Level | 0 | <p>The Write Protect Switch is supported for memory and combo cards. This bit reflects the SDWP# pin.</p> <p>1 – Write enabled (SDWP#=1) 0 – Write protected (SDWP#=0)</p> |
| 18 | R | Card Detect Pin Level | 0 | <p>This bit reflects the inverse value of the SDCD# pin.</p> <p>Debouncing is not performed on this bit. This bit may be valid when Card State Stable is set to 1, but it is not guaranteed because of propagation delay.</p> <p>Use of this bit is limited to testing since it must be debounced by software.</p> <p>1 – Card present (SDCD#=0) 0 – No card present (SDCD#=1)</p> |
| 17 | R | Card State Stable | 0 | <p>This bit is used for testing. If it is 0, the Card Detect Pin Level is not stable. If this bit is set to 1, it means the Card Detect Pin Level is stable.</p> <p>No Card state can be detected by this bit if set to 1 and Card Inserted is set to 0.</p> <p>The Software Reset For All in the Software Reset register shall not affect this bit.</p> <p>1 – No Card Inserted 0 – Reset or Debouncing</p> |

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------|-----------|--|
| 16 | R | Card Inserted | 0 | <p>This bit indicates whether a card has been inserted. The SMIHC Controller shall debounce this signal so that the Host Driver will not need to wait for it to stabilize. Changing from 0 to 1 generates a Card Insertion interrupt in the Normal Interrupt Status register and changing from 1 to 0 generates a Card Removal interrupt in the Normal Interrupt Status register.</p> <p>The Software Reset For All in the Software Reset register shall not affect this bit.</p> <p>If a card is removed while its power is on and its clock is oscillating, the Host Controller shall clear SD Bus Power in the Power Control register and SD Clock Enable in the Clock Control register. When this bit is changed from 1 to 0, the SMIHC Controller shall immediately stop driving CMD and DAT[3:0] (tri-state). In addition, the Host Driver should clear the SMIH Controller by the Software Reset For All in Software Reset register.</p> <p>The card detect is active regardless of the SD Bus Power.</p> <p>1 – Card Inserted 0 – Reset or Debouncing or No Card</p> |
| 15:12 | R | Reserved | 0 | Reserved |
| 11 | R | Buffer Read Enable | 0 | <p>This status is used for non-DMA read transfers. The SMIHC Controller may implement multiple buffers to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this bit is 1, readable data exists in the buffer. A change of this bit from 1 to 0 occurs when all the block data is read from the buffer.</p> <p>A change of this bit from 0 to 1 occurs when block data is ready in the buffer and generates the Buffer Read Ready interrupt.</p> <p>1 – Read enable 0 – Read disable</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|--|
| 10 | R | Buffer Write Enable | 0 | <p>This status is used for non-DMA write transfers.</p> <p>The SMIHC Controller can implement multiple buffers to transfer data efficiently. This read only flag indicates if space is available for write data.</p> <p>If this bit is 1, data can be written to the buffer. A change of this bit from 1 to 0 occurs when all the block data is written to the buffer. A change of this bit from 0 to 1 occurs when top of block data can be written to the buffer and generates the Buffer Write Ready interrupt. The SMIHC Controller should neither set Buffer Write Enable nor generate Buffer Write Ready Interrupt after the last block data is written to the Buffer Data Port Register.</p> <p>1 – Write enable 0 – Write disable</p> |
| 9 | R | Read Transfer Active | 0 | <p>This status is used for detecting completion of a read transfer.</p> <p>This bit is set to 1 for either of the following conditions:</p> <ul style="list-style-type: none"> (1) After the end bit of the read command. (2) When read operation is restarted by writing a 1 to Continue Request in the Block Gap Control register. <p>This bit is cleared to 0 for either of the following conditions::</p> <ul style="list-style-type: none"> (1) When the last data block as specified by block length is transferred to the System. (2) In case of ADMA2, end of read operation is designated by Descriptor Table. (3) When all valid data blocks in the SMIHC Controller have been transferred to the System and no current block transfers are being sent as a result of the Stop At Block Gap Request being set to 1. <p>A Transfer Complete interrupt is generated when this bit changes to 0.</p> <p>1 – Write enable 0 – Write disable</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|--|
| 8 | R | Write Transfer Active | 0 | <p>This status indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the SMIHC Controller.</p> <p>This bit is set in either of the following cases:</p> <ul style="list-style-type: none"> • After the end bit of the write command. • When write operation is restarted by writing a 1 to Continue Request in the Block Gap Control register. <p>This bit is cleared in either of the following cases:</p> <ul style="list-style-type: none"> • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple) In case of ADMA2, transfer count is designated by Descriptor Table. • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. <p>During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to 0, as the result of the Stop At Block Gap Request begin set. This status is useful for the Host Driver in determining non DAT line commands can be issued during write busy.</p> <p>1 – Transferring Data</p> <p>0 – No valid data</p> |
| 7:4 | R | Reserved | 0 | Reserved |
| 3 | R | Re-Tuning Request | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|---|
| 2 | R | DAT Line Active | 0 | <p>This bit indicates whether one of the DAT line on SD Bus is in use.</p> <p>(a) In the case of read transactions</p> <p>This status indicates whether a read transfer is executing on the SD Bus. Changing this value from 1 to 0 generates a Block Gap Event interrupt in the Normal Interrupt Status register, as the result of the Stop At Block Gap Request being set.</p> <p>This bit shall be set in either of the following cases:</p> <ul style="list-style-type: none"> (1) After the end bit of the read command. (2) When writing a 1 to Continue Request in the Block Gap Control register to restart a read transfer. <p>This bit shall be cleared in either of the following cases</p> <ul style="list-style-type: none"> (1) When the end bit of the last data block is sent from the SD Bus to the SMIHC Controller. (2) When a read transfer is stopped at the block gap initiated by a Stop At Block Gap Request. <p>The SMIHC Controller shall stop read operation at the start of the interrupt cycle of the next block gap by driving</p> <p>Read Wait or stopping SD clock. If the Read Wait signal is already driven (due to data buffer cannot receive data),</p> <p>the SMIHC Controller can continue to stop read operation by driving the Read Wait signal.</p> <p>It is necessary to support Read Wait in order to use suspend / resume function.</p> <p>(b) In the case of write transactions</p> <p>This status indicates that a write transfer is executing on the SD Bus.</p> <p>Changing this value from 1 to 0 generate a Transfer Complete interrupt in the Normal Interrupt Status register.</p> <p>This bit shall be set in either of the following cases:</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| | | | | <p>(1) After the end bit of the write command.</p> <p>(2) When writing to 1 to Continue Request in the Block Gap Control register to continue a write transfer.</p> <p>This bit shall be cleared in either of the following cases:</p> <p>(1) When the SD card releases write busy of the last data block.</p> <p>If SD card does not drive busy signal for 8 SD Clocks, the SMIHC Controller shall consider the card drive "Not Busy".</p> <p>In case of ADMA2, the last block is designated by the last transfer of Descriptor Table.</p> <p>(2) When the SD card releases write busy prior to waiting for write transfer as a result of a Stop At Block Gap Request.</p> <p>(c) Command with busy</p> <p>This status indicates whether a command indicates busy (ex. erase command for memory) is executing on the SD Bus.</p> <p>This bit is set after the end bit of the command with busy and cleared when busy is deasserted.</p> <p>Changing this bit from 1 to 0 generate a Transfer Complete interrupt in the Normal Interrupt Status register.</p> <p>1 – DAT Line Active</p> <p>0 – DAT Line Inactive</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|--|
| 1 | R | Command Inhibit (DAT) | 0 | <p>This status bit is generated if either the DAT Line Active or the Read Transfer Active is set to 1.</p> <p>If this bit is 0, it indicates the SMIHC Controller can issue the next SD Command.</p> <p>Commands with busy signal belong to Command Inhibit (DAT) (ex. R1b, R5b type).</p> <p>Changing from 1 to 0 generates a Transfer Complete interrupt in the Normal Interrupt Status register.</p> <p>Note: The SD Host Driver can save registers in the range of 000-00Dh for a suspend transaction after this bit has changed from 1 to 0.</p> <p>1 – Cannot issue command which uses the DAT line</p> <p>0 – Can issue command which uses the DAT line</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|---|
| 0 | R | Command Inhibit (CMD) | 0 | <p>If this bit is 0, it indicates the CMD line is not in use and the SMIHC Controller can issue a SD Command using the CMD line.</p> <p>This bit is set immediately after the Command register (00Fh) is written.</p> <p>This bit is cleared when the command response is received. Auto CMD12 and Auto CMD23 consist of two responses.</p> <p>In this case, this bit is not cleared by the response of CMD12 or CMD23 but cleared by the response of a read/write command.</p> <p>Status issuing Auto CMD12 is not read from this bit. So if a command is issued during Auto CMD12 operation,</p> <p>SMIHC Controller shall manage to issue two commands:</p> <p>CMD12 and a command set by Command register.</p> <p>Even if the Command Inhibit (DAT) is set to 1, commands using only the CMD line can be issued if this bit is 0.</p> <p>Changing from 1 to 0 generates a Command Complete Interrupt in the Normal Interrupt Status register.</p> <p>If the SMIHC Controller cannot issue the command because of a command conflict error or because of Command Not Issued By Auto CMD12 Error, this bit shall remain 1 and the Command Complete is not set.</p> <p>1 – Cannot issue command</p> <p>0 – Can issue command using only CMD line</p> |

443 . Present State Register Part 1 Description

SMIH Host Control 1 Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------------|-----------|--|
| 7 | R/W | Card Detect Signal Selection | 0 | <p>This bit selects source for the card detection.</p> <p>1 – The Card Detect Test Level is selected (for test purpose)</p> <p>0 – SDCD# is selected (for normal use)</p> <p>When the source for the card detection is switched, the interrupt should be disabled during the switching period by clearing the Interrupt Status/Signal Enable register in order to mask unexpected interrupt being caused by the glitch.</p> <p>The Interrupt Status/Signal Enable should be disabled during over the period of debouncing.</p> |
| 6 | R/W | Card Detect Test Level | 0 | <p>This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not.</p> <p>1 – Card Inserted</p> <p>0 – No Card</p> |
| 5 | R/W | Extended Data Transfer Width | 0 | <p>This bit controls 8-bit bus width mode for embedded device. Support of this function is indicated in 8-bit Support for Embedded Device in the Capabilities register. If a device supports 8-bit bus mode, this bit may be set to 1. If this bit is 0, bus width is controlled by Data Transfer Width in the Host Control 1 register. This bit is not effective when multiple devices are installed on a bus slot (Slot Type is set to 10b in the Capabilities register). In this case, each device bus width is controlled by Bus Width Preset field in the Shared Bus register.</p> <p>1 – 8-bit Bus Width</p> <p>0 – Bus Width is Selected by Data Transfer Width</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 4:3 | R/W | DMA Select | 0 | <p>One of supported DMA modes can be selected. The host driver shall check support of DMA modes by referring</p> <p>the Capabilities register. Use of Selected DMA is Determined by DMA Enable of the Transfer Mode register.</p> <p>00 – SDMA is selected</p> <p>01 – Reserved</p> <p>10 – 32-bit Address ADMA2 is selected</p> <p>11 – Reserved</p> |
| 2 | R/W | High Speed Enable | 0 | <p>This bit is optional. Before setting this bit, the Host Driver shall check the High Speed Support in the Capabilities register.</p> <p>If this bit is set to 0 (default), the SMIHC Controller outputs CMD line and DAT lines at the falling edge of the SD Clock</p> <p>(up to 25MHz). If this bit is set to 1, the SMIHC Controller outputs CMD line and DAT lines at the rising edge of the</p> <p>SD Clock (up to 50MHz).</p> <p>If Preset Value Enable in the Host Control 2 register is set to 1, Host Driver needs to reset SD Clock Enable before</p> <p>changing this field to avoid generating clock glitches. After setting this field, the Host Driver sets SD Clock Enable again.</p> <p>1 – High Speed mode</p> <p>0 – Normal Speed mode</p> |
| 1 | R/W | Data Transfer Width | 0 | <p>This bit selects the data width of the SMIHC Controller.</p> <p>The Host Driver shall set it to match the data width of the SD card.</p> <p>1 – 4-bit mode</p> <p>0 – 1-bit mode</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------|-----------|--|
| 0 | R/W | LED Control | 0 | <p>This bit is used to caution the user not to remove the card while the SD card is being accessed.</p> <p>If the software is going to issue multiple SD commands, this bit can be set during all these transactions.</p> <p>It is not necessary to change for each transaction.</p> <p>1 – LED on 0 – LED off</p> |

444 . Host Control 1 Register Description

SMIH Power Control Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|--|
| 7:6 | R/W | Reserved | 0 | Reserved |
| 5 | R/W | SMIH_RST_N | 1 | External Hardware reset, used only in MMC mode |
| 4 | R/W | SMIH_OD_PP | 0 | <p>This bit is used only in MMC mode.</p> <p>1 – Push Pull Mode 0 – Open Drain Mode</p> |
| 3:1 | R/W | SD Bus Voltage Select | 0 | <p>By setting these bits, the Host Driver selects the voltage level for the SD card. Before setting this register, the Host Driver shall check the Voltage Support bits in the Capabilities register. If an unsupported voltage is selected, the Host System shall not supply SD Bus voltage.</p> <p>111b – 3.3V (Typ.) 110b – 3.0V (Typ.) 101b – 1.8V (Typ.) 100b-000b – Reserved</p> |
| 0 | R/W | SD Bus Power | 0 | <p>Before setting this bit, the SD Host Driver shall set SD Bus Voltage Select. If the SMIHC Controller detects the No Card state, this bit shall be cleared.</p> <p>If this bit is cleared, the SMIHC Controller shall immediately stop driving CMD and DAT[3:0] (tri-state) and drive SDCLK to low level.</p> <p>1 – Power on 0 – Power off</p> |

445 . Power Control Register Description

SMIH Block Gap Control Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 7:4 | R/W | Reserved | 0 | Reserved |
| 3 | R/W | Interrupt At Block Gap | 0 | <p>This bit is valid only in 4-bit mode of the SDIO card and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0. When the Host Driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card.</p> <p>1 – Enabled 0 – Disabled</p> |
| 2 | R/W | Read Wait Control | 0 | <p>The read wait function is optional for SDIO cards. If the card supports read wait, set this bit to enable use of the read wait protocol to stop read data using the DAT[2] line. Otherwise, the SMIHC Controller has to stop the SD Clock to hold read data, which restricts commands generation.</p> <p>When the Host Driver detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. If the card does not support read wait, this bit shall never be set to 1 otherwise DAT line conflict may occur. If this bit is set to 0, Suspend/Resume cannot be supported.</p> <p>1 – Enable Read Wait Control 0 – Disable Read Wait Control</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------------|-----------|---|
| 1 | R/W | Continue Request | 0 | <p>This bit is used to restart a transaction which was stopped using Stop At Block Gap request. To cancel stop at block gap Stop At Block Gap to Zero And set this bit to restart the transfer.</p> <p>The Host controller Automatically clears the bit in either of the following cases</p> <ul style="list-style-type: none"> (1) In case of a Read Transaction Dat Line Active changes from 0 to 1 as a read transaction restarts (2) In case of a Write Transaction Write Transfer Active changes from 0 to 1 as write transaction restarts. <p>Therefore it is not necessary for the Host Driver to set this bit to 0. If Stop At Block Gap request is set to 1. Any write to this bit is ignored.</p> <p>1 – Restart 0 – Not affect Automatic clear Bit</p> |
| 0 | R/W | Stop At Block Gap request | 0 | <p>This bit is used to stop executing read and write transaction at the next block gap for non-DMA, SDMA and ADMA transfers. The Host Driver shall leave this bit set to 1 until the Transfer Complete is set to 1. Clearing both Stop At Block Gap Request and Continue Request shall not cause the transaction to restart. The SMIHC Controller shall stop read transfer by using Read Wait or stopping SD clock. In case of write transfers in which the Host Driver writes data to the Buffer Data Port register, the Host Driver shall set this bit after all block data is written.</p> <p>If this bit is set to 1, the Host Driver shall not write data to Buffer Data Port register.</p> <p>This bit affects Read Transfer Active, Write Transfer Active, DAT Line Active and Command Inhibit (DAT) in the Present State register.</p> <p>1 – Stop 0 – Transfer</p> |

446 . Block Gap Control Register Description

SMIH Wake up Control Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|-------------|
| 7:3 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---|-----------|--|
| 2 | R/W | Wake up Event Enable On SD Card Removal | 0 | <p>This bit enables wakeup event via Card Removal assertion in the Normal Interrupt Status register.</p> <p>FN_WUS (Wake Up Support) in CIS does not affect this bit.</p> <p>1 – Stop 0 – Transfer</p> |
| 1 | R/W | Wake up Event Enable On SD Card Insertion | 0 | <p>This bit enables wakeup event via Card Insertion assertion in the Normal Interrupt Status register.</p> <p>FN_WUS (Wake Up Support) in CIS does not affect this bit.</p> <p>1 – Stop 0 – Transfer</p> |
| 0 | R/W | Wake up Event Enable On Card Interrupt | 0 | <p>This bit enables wakeup event via Card Interrupt assertion in the Normal Interrupt Status register.</p> <p>This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1.</p> <p>1 – Enable 0 – Disable</p> |

447 . Wakeup Control Register Description

SMIH Clock Control Register

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | |
|-----------------|---------------------------|------------------------|-----------|--|-----------------|---------------------------|-----------------|---------------------------|-----------------|--------------------------|-----------------|--------------------------|-----------------|--------------------------|-----------------|-------------------------|-----------------|-------------------------|-----------------|-------------------------|
| 15:8 | R/W | SDCLK Frequency Select | 0 | <p>This register is used to select the frequency of SDCLK pin.</p> <p>(1) 8-bit Divided Clock Mode</p> <p>The frequency is not programmed directly; rather this register holds the divisor of the Base Clock Frequency</p> <p>For SD Clock in the Capabilities register. Only the following settings are allowed.</p> <table border="1" style="margin-left: 20px;"> <tr><td>80_h</td><td>base clock divided by 256</td></tr> <tr><td>40_h</td><td>base clock divided by 128</td></tr> <tr><td>20_h</td><td>base clock divided by 64</td></tr> <tr><td>10_h</td><td>base clock divided by 32</td></tr> <tr><td>08_h</td><td>base clock divided by 16</td></tr> <tr><td>04_h</td><td>base clock divided by 8</td></tr> <tr><td>02_h</td><td>base clock divided by 4</td></tr> <tr><td>01_h</td><td>base clock divided by 2</td></tr> </table> <p>Setting 01h specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor</p> <p>but it should not be set. The three default divider values can be calculated by the frequency that is defined by the Base Clock Frequency</p> <p>For SD Clock in the Capabilities register.</p> <p>400KHz divider value</p> <p>25MHz divider value</p> <p>50MHz divider value</p> <p>According to the Physical Layer Specification, the maximum SD Clock frequency is 25 MHz in normal speed mode and 50MHz in high speed mode, and shall never exceed this limit.</p> <p>The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / divisor</p> <p>Thus, choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency.</p> <p>For example, if the Base Clock Frequency For SD Clock in the Capabilities register has the value 33MHz, and the target</p> | 80 _h | base clock divided by 256 | 40 _h | base clock divided by 128 | 20 _h | base clock divided by 64 | 10 _h | base clock divided by 32 | 08 _h | base clock divided by 16 | 04 _h | base clock divided by 8 | 02 _h | base clock divided by 4 | 01 _h | base clock divided by 2 |
| 80 _h | base clock divided by 256 | | | | | | | | | | | | | | | | | | | |
| 40 _h | base clock divided by 128 | | | | | | | | | | | | | | | | | | | |
| 20 _h | base clock divided by 64 | | | | | | | | | | | | | | | | | | | |
| 10 _h | base clock divided by 32 | | | | | | | | | | | | | | | | | | | |
| 08 _h | base clock divided by 16 | | | | | | | | | | | | | | | | | | | |
| 04 _h | base clock divided by 8 | | | | | | | | | | | | | | | | | | | |
| 02 _h | base clock divided by 4 | | | | | | | | | | | | | | | | | | | |
| 01 _h | base clock divided by 2 | | | | | | | | | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | |
|------------------|-------------------------------|--------------------------------------|-----------|---|------------------|----------------------|-------|-------|---|-------------------------------|-------|-------|------------------|-------------------|------------------|-------------------|------------------|---------------------------|
| | | | | <p>frequency is 25MHz, then choosing the divisor value of 01h will yield 16.5MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400KHz, the divisor value of 40h yields the optimal clock value of 258KHz.</p> <p>(2) 10-bit Divided Clock Mode</p> <p>The length of divider is extended to 10 bits and all divider values shall be supported.</p> <table border="1" data-bbox="857 707 1246 954"> <tr><td>3FF_h</td><td>1/2046 Divided Clock</td></tr> <tr><td>.....</td><td>.....</td></tr> <tr><td>N</td><td>1/2N Divided Clock (Duty 50%)</td></tr> <tr><td>.....</td><td>.....</td></tr> <tr><td>002_h</td><td>1/4 Divided Clock</td></tr> <tr><td>001_h</td><td>1/2 Divided Clock</td></tr> <tr><td>000_h</td><td>Base Clock (10MHz-254MHz)</td></tr> </table> <p>(3) Programmable Clock Mode</p> <p>Not Supported</p> | 3FF _h | 1/2046 Divided Clock | | | N | 1/2N Divided Clock (Duty 50%) | | | 002 _h | 1/4 Divided Clock | 001 _h | 1/2 Divided Clock | 000 _h | Base Clock (10MHz-254MHz) |
| 3FF _h | 1/2046 Divided Clock | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| N | 1/2N Divided Clock (Duty 50%) | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| 002 _h | 1/4 Divided Clock | | | | | | | | | | | | | | | | | |
| 001 _h | 1/2 Divided Clock | | | | | | | | | | | | | | | | | |
| 000 _h | Base Clock (10MHz-254MHz) | | | | | | | | | | | | | | | | | |
| 7:6 | R/W | Upper Bits of SDCLK Frequency Select | 0 | <p>PE_SMIH Controller shall support these bits to expand SDCLK</p> <p>Frequency Select to 10-bit. Bit 07-06 is assigned to bit 09-08 of clock</p> <p>divider in SDCLK Frequency Select.</p> | | | | | | | | | | | | | | |
| 5:3 | R/W | Reserved | 0 | Reserved | | | | | | | | | | | | | | |
| 2 | R/W | SD Clock Enable | 0 | <p>The SMIHC Controller shall stop SDCLK when writing this bit to 0.</p> <p>SDCLK Frequency Select can be changed when this bit is 0. Then, the SMIHC Controller shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK=0). If the Card Inserted in the Present State register is cleared, this bit shall be cleared.</p> <p>1 – Enable</p> <p>0 – Disable</p> | | | | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|---|
| 1 | R | Internal Clock Stable | 0 | <p>This bit is set to 1 when SD Clock is stable after writing to Internal Clock Enable in this register to 1.</p> <p>The SD Host Driver shall wait to set SD Clock Enable until this bit is set to 1.</p> <p>Note: This is useful when using PLL for a clock oscillator that requires setup time.</p> <p>1 – Enable 0 – Disable</p> |
| 0 | R/W | Internal Clock Enable | 0 | <p>This bit is set to 0 when the Host Driver is not using the SMIHC Controller or the SMIHC Controller awaits a wakeup interrupt.</p> <p>The SMIHC Controller should stop its internal clock to go very low power state.</p> <p>Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to 1.</p> <p>When clock oscillation is stable, the SMIHC Controller shall set Internal Clock Stable in this register to 1.</p> <p>This bit shall not affect card detection.</p> <p>1 – Enable 0 – Disable</p> |

448 .Clock Control Register Description

SMIH Timeout Control Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|-------------|
| 7:4 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | |
|-------------------|-----------|----------------------------|-----------|---|-------------------|----------|-------------------|-----------|-------|-------|-------------------|-----------|-------------------|-----------|
| 3:0 | R/W | Data Timeout Counter Value | 0 | <p>This value determines the interval by which DAT line timeouts are detected.</p> <p>For more information about timeout generation, refer to the Data Timeout Error in the Error Interrupt Status register.</p> <p>Timeout clock frequency will be generated by dividing the base clock TMCLK value by this value.</p> <p>When setting this register, prevent inadvertent timeout events by clearing the Data Timeout Error Status Enable (in the Error Interrupt Status Enable register)</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>1111_b</td><td>Reserved</td></tr> <tr> <td>1110_b</td><td>TMCLK x 2</td></tr> <tr> <td>.....</td><td>.....</td></tr> <tr> <td>0001_b</td><td>TMCLK x 2</td></tr> <tr> <td>0000_b</td><td>TMCLK x 2</td></tr> </table> | 1111 _b | Reserved | 1110 _b | TMCLK x 2 | | | 0001 _b | TMCLK x 2 | 0000 _b | TMCLK x 2 |
| 1111 _b | Reserved | | | | | | | | | | | | | |
| 1110 _b | TMCLK x 2 | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| 0001 _b | TMCLK x 2 | | | | | | | | | | | | | |
| 0000 _b | TMCLK x 2 | | | | | | | | | | | | | |

449 . Timeout Control Register Description

SMIH Software Reset Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|-------------|
| 7:3 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------|-----------|--|
| 2 | R/W* | Software Reset For DAT Line | 0 | <p>Only part of data circuit is reset. DMA circuit is also reset.</p> <p>The following registers and bits are cleared by this bit:</p> <ul style="list-style-type: none"> Buffer Data Port register Buffer is cleared and initialized. Present State register Buffer Read Enable Buffer Write Enable Read Transfer Active Write Transfer Active DAT Line Active Command Inhibit (DAT) Block Gap Control register Continue Request Stop At Block Gap Request Normal Interrupt Status register Buffer Read Ready Buffer Write Ready DMA Interrupt Block Gap Event Transfer Complete 1 – Reset 0 – Work |
| 1 | R/W* | Software Reset For CMD Line | 0 | <p>Only part of command circuit is reset.</p> <p>The following registers and bits are cleared by this bit:</p> <ul style="list-style-type: none"> Present State register Command Inhibit (CMD) Normal Interrupt Status register Command Complete 1 – Reset 0 – Work |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 0 | R/W* | Software Reset For All | 0 | <p>This reset affects the entire SMIHC Controller except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to 0. During its initialization, the Host Driver shall set this bit to 1 to reset the SMIHC Controller. The SMIHC Controller shall reset this bit to 0 when Capabilities registers are valid and the Host Driver can read them. Additional use of Software Reset For All may not affect the value of the Capabilities registers. If this bit is set to 1, the host driver should issue reset command and reinitialize the SD card.</p> <p>1 – Reset 0 – Work</p> |

* - automatic clear bit

450 . Software Reset Register Description

SMIH Normal Interrupt Status Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------|-----------|---|
| 15 | R | Error Interrupt | 0 | <p>If any of the bits in the Error Interrupt Status register are set, then this bit is set. Therefore the Host Driver can efficiently test for an error by checking this bit first.</p> <p>1 – Error 0 – No Error</p> |
| 14 | R/W* | Boot Done Interrupt | 0 | <p>1 – Boot Mode / Alternate Boot Mode Operation is Done 0 – Boot Mode / Alternate Boot Mode Operation is in Progress</p> |
| 13 | R/W* | Boot Ack Complete Interrupt | 0 | <p>1 – Boot Ack is received successfully from the card 0 – Boot Ack is not received</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|---|
| 12 | R | Re-Tuning Event | 0 | <p>This status is set if Re-Tuning Request in the Present State register changes from 0 to 1.</p> <p>SMIHC Controller requests Host Driver to perform re-tuning for next data transfer.</p> <p>Current data transfer (not large block count) can be completed without re-tuning.</p> <p>1 – Re-Tuning should be performed 0 – Re-Tuning is not required</p> |
| 11 | R | INT_C | 0 | <p>This status is set if INT_C is enabled and INT_C# pin is in low level.</p> <p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the INT_C interrupt factor.</p> <p>Refer to the Shared Bus Control register.</p> <p>1 – INT_C is detected 0 – No interrupt is detected</p> |
| 10 | R | INT_B | 0 | <p>This status is set if INT_B is enabled and INT_B# pin is in low level.</p> <p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the INT_B interrupt factor.</p> <p>Refer to the Shared Bus Control register.</p> <p>1 – INT_B is detected 0 – No interrupt is detected</p> |
| 9 | R | INT_A | 0 | <p>This status is set if INT_A is enabled and INT_A# pin is in low level.</p> <p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the INT_A interrupt factor.</p> <p>Refer to the Shared Bus Control register.</p> <p>1 – INT_A is detected 0 – No interrupt is detected</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------|-----------|--|
| 8 | R | Card Interrupt | 0 | <p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor.</p> <p>In 1-bit mode, the SMIHC Controller shall detect the Card Interrupt without SD Clock to support wakeup.</p> <p>In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so there are some sample delays between the interrupt signal from the SD card and the interrupt to the Host System.</p> <p>When this status has been set and the Host Driver needs to start this interrupt service, Card Interrupt Status Enable in the Normal Interrupt Status Enable register may be set to 0 in order to clear the card interrupt statuses latched in the SMIHC Controller and to stop driving the interrupt signal to the Host System.</p> <p>After completion of the card interrupt service (It should reset interrupt factors in the SD card and the interrupt signal may not be asserted), set Card Interrupt Status Enable to 1 and start sampling the interrupt signal again.</p> <p>Interrupt detected by DAT[1] is supported when there is a card per slot. In case of shared bus, interrupt pins are used to detect interrupts. If 000b is set to Interrupt Pin Select in the Shared Bus Control register, this status is effective. Non-zero value is set to Interrupt Pin Select, INT_A, INT_B or INT_C is then used to device interrupts.</p> <p>1 – Generate Card Interrupt 0 – No Card Interrupt</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|---|
| 7 | R/W* | Card Removal | 0 | <p>This status is set if the Card Inserted in the Present State register changes from 1 to 0. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect state may possibly be changed when the Host Driver clear this bit and interrupt event may not be generated.</p> <p>1 – Card removed 0 – Card state stable or Debouncing</p> |
| 6 | R/W* | Card Insertion | 0 | <p>This status is set if the Card Inserted in the Present State register changes from 0 to 1. When the Host Driver writes this bit to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card detect state may possibly be changed when the Host Driver clear this bit and interrupt event may not be generated.</p> <p>1 – Card inserted 0 – Card state stable or Debouncing</p> |
| 5 | R/W* | Buffer Read Ready | 0 | <p>This status is set if the Buffer Read Enable changes from 0 to 1. Refer to the Buffer Read Enable in the Present State register.</p> <p>While performing tuning procedure (Execute Tuning is set to 1), Buffer Read Ready is set to 1 for every CMD19 execution.</p> <p>1 – Ready to read buffer 0 – Not ready to read buffer</p> |
| 4 | R/W* | Buffer Write Ready | 0 | <p>This status is set if the Buffer Write Enable changes from 0 to 1. Refer to the Buffer Write Enable in the Present State register.</p> <p>1 – Ready to write buffer 0 – Not ready to write buffer</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|---|
| 3 | R/W* | DMA Interrupt | 0 | <p>This status is set if the SMIHC Controller detects the Host SDMA Buffer boundary during transfer.</p> <p>Refer to the Host SDMA Buffer Boundary in the Block Size register.</p> <p>Other DMA interrupt factors may be added in the future. In case of ADMA, by setting Int field in the descriptor table,</p> <p>SMIHC Controller generates this interrupt. Suppose that it is used for debugging. This interrupt shall not be generated after the Transfer Complete.</p> <p>1 – DMA Interrupt is generated 0 – No DMA Interrupt</p> |
| 2 | R/W* | Block Gap Event | 0 | <p>If the Stop At Block Gap Request in the Block Gap Control register is set, this bit is set when both a read / write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1.</p> <p>(1) In the case of a Read Transaction This bit is set at the falling edge of the DAT Line Active Status (When the transaction is stopped at SD Bus timing. The Read Wait shall be supported in order to use this function.</p> <p>(2) In the case of Write Transaction This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing).</p> <p>1 – Transaction stopped at block gap 0 – No Block Gap Event</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------|-----------|--|
| 1 | R/W* | Transfer Complete | 0 | <p>This bit is set when a read / write transfer and a command with busy is completed.</p> <p>(1) In the case of a Read Transaction</p> <p>This bit is set at the falling edge of Read Transfer Active Status. This interrupt is generated in two cases. The first is when a data transfer is completed as specified by data length (After the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request in the Block Gap Control register (After valid data has been read to the Host System).</p> <p>(2) In the case of a Write Transaction</p> <p>This bit is set at the falling edge of the DAT Line Active Status. This interrupt is generated in two cases. The first is when the last data is written to the SD card as specified by data length and the busy signal released. The second is when data transfers are stopped at the block gap by setting Stop At Block Gap Request in the Block Gap Control register and data transfers completed.</p> <p>(After valid data is written to the SD card and the busy signal released).</p> <p>(3) In the case of a command with busy</p> <p>This bit is set when busy is de-asserted. Refer to DAT Line Active and Command Inhibit (DAT) in the Present State register.</p> <p>The table below shows that Transfer Complete has higher priority than Data Timeout Error. If both bits are set to 1, execution of a command can be considered to be completed.</p> <ul style="list-style-type: none"> • Relation between Transfer Complete and Data Timeout Error |

| Bit | Access | Function | POR Value | Description | | |
|-----|--------|----------|-----------|---|--------------|--|
| | | | | Transfer Complete | Data Timeout | Meaning of the Status |
| | | | | 0 | 0 | Interrupted by another factor |
| | | | | 0 | 1 | Timeout occur during transfer |
| | | | | 1 | Don't Care | Command Execution complete |
| | | | | 1 – Command execution is completed 0 – Not completed | | While performing tuning procedure (Execute Tuning is set to 1), Transfer Complete is not set to 1. |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | |
|-------------------------|-----------------------|---|-----------|---|------------------|-----------------------|-----------------------|---|---|-------------------------------|------------|---|---|---|---|-------------------|----------------------|--|----------------------|-------------------------|--|-------------------------|
| 0 | R/W* | Command Complete | 0 | <p>This bit is set when get the end bit of the command response. Auto CMD12 and Auto CMD23 consist of two responses.</p> <p>Command Complete is not generated by the response of CMD12 or CMD23 but generated by the response of a read/write command. Refer to Command Inhibit (CMD) in the Present State register for how to control this bit.</p> <p>The table below shows that Command Timeout Error has higher priority than Command Complete. If both bits are set to 1, it can be considered that the response was not received correctly.</p> | | | | | | | | | | | | | | | | | | |
| | | | | <table border="1"> <thead> <tr> <th>Command Complete</th> <th>Command Timeout Error</th> <th>Meaning of the Status</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Interrupted by another factor</td> </tr> <tr> <td>Don't Care</td> <td>1</td> <td>Response not received within 64 SDCLK cycles.</td> </tr> <tr> <td>1</td> <td>0</td> <td>Response received</td> </tr> <tr> <td colspan="2">1 – Command complete</td><td>1 – Command complete</td></tr> <tr> <td colspan="2">0 – No Command complete</td><td>0 – No Command complete</td></tr> </tbody> </table> | Command Complete | Command Timeout Error | Meaning of the Status | 0 | 0 | Interrupted by another factor | Don't Care | 1 | Response not received within 64 SDCLK cycles. | 1 | 0 | Response received | 1 – Command complete | | 1 – Command complete | 0 – No Command complete | | 0 – No Command complete |
| Command Complete | Command Timeout Error | Meaning of the Status | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | Interrupted by another factor | | | | | | | | | | | | | | | | | | | | |
| Don't Care | 1 | Response not received within 64 SDCLK cycles. | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | Response received | | | | | | | | | | | | | | | | | | | | |
| 1 – Command complete | | 1 – Command complete | | | | | | | | | | | | | | | | | | | | |
| 0 – No Command complete | | 0 – No Command complete | | | | | | | | | | | | | | | | | | | | |

* - Write-1-to-clear status

451 . Normal Interrupt Status Register Description

SMIH Error Interrupt Status Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 15:10 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 9 | R/W* | ADMA Error | 0 | <p>This bit is set when the SMIHC Controller detects errors during ADMA based data transfer. The state of the ADMA at an error occurrence is saved in the ADMA Error Status Register, In addition, the SMIHC Controller generates this Interrupt when it detects invalid descriptor data (Valid=0) at the ST_FDS state.</p> <p>ADMA Error State in the ADMA Error Status indicates that an error occurs in ST_FDS state. The Host Driver may find that Valid bit is not set at the error descriptor.</p> <p>1 – Error 0 – No Error</p> |
| 8 | R/W* | Auto CMD Error | 0 | <p>Auto CMD12 and Auto CMD23 use this error status. This bit is set when detecting that one of the bits D00-D04 in Auto CMD Error Status register has changed from 0 to 1.</p> <p>In case of Auto CMD12, this bit is set to 1, not only when the errors in Auto CMD12 occur but also when Auto CMD12 is not executed due to the previous command error.</p> <p>1 – Error 0 – No Error</p> |
| 7 | R/W* | Current Limit Error | 0 | <p>By setting the SD Bus Power bit in the Power Control register, the SMIHC Controller is requested to supply power for the SD Bus. If the SMIHC Controller supports the Current Limit function, it can be protected from an illegal card by stopping power supply to the card in which case this bit indicates a failure status. Reading 1 means the SMIHC Controller is not supplying power to SD card due to some failure. Reading 0 means that the SMIHC Controller is supplying power and no error has occurred.</p> <p>The SMIHC Controller may require some sampling time to detect the current limit. If the SMIHC Controller does not support this function, this bit shall always be set to 0.</p> <p>1 – Power fail 0 – No Error</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|---|
| 6 | R/W* | Data End Bit Error | 0 | <p>Occurs either when detecting 0 at the end bit position of read data which uses the DAT line or at the end bit position of the CRC Status.</p> <p>1 – Error 0 – No Error</p> |
| 5 | R/W* | Data CRC Error | 0 | <p>Occurs when detecting CRC error when transferring read data which uses the DAT line or when detecting the Write CRC status having a value of other than "010".</p> <p>1 – Error 0 – No Error</p> <p>Note: SPI data error token will result in data CRC error.</p> |
| 4 | R/W* | Data Timeout Error | 0 | <p>This bit is set when detecting one of following timeout conditions:</p> <ul style="list-style-type: none"> • Busy timeout for R1b,R5b type • Busy timeout after Write CRC status • Write CRC Status timeout • Read Data timeout. <p>1 – Timed out 0 – No Error</p> |
| 3 | R/W* | Command Index Error | 0 | <p>This bit is set if a Command Index error occurs in the command response.</p> <p>1 – Error 0 – No Error</p> |
| 2 | R/W* | Command End Bit Error | 0 | <p>This bit is set when detecting that the end bit of a command response is 0.</p> <p>1 – End Bit Error Generated 0 – No Error</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|--|
| 1 | R/W* | Command CRC Error | 0 | <p>Command CRC Error is generated in two cases.</p> <p>If a response is returned and the Command Timeout Error is set to 0 (indicating no timeout), this bit is set to 1 when detecting a CRC error in the command response.</p> <p>The SMIHC Controller detects a CMD line conflict by monitoring the CMD line when a command is issued. If the SMIHC Controller drives the CMD line to 1 level, but</p> <p>detects 0 level on the CMD line at the next SD clock edge, then the SMIHC Controller shall abort the command (Stop driving CMD line) and set this bit to 1.</p> <p>The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict.</p> <p>1 – CRC Error Generated 0 – No Error</p> |
| 0 | R/W* | Command Timeout Error | 0 | <p>This bit is set only if no response is returned within 64 SD clock cycles from the end bit of the command. If the SMIHC Controller detects a CMD line conflict, in which case</p> <p>Command CRC Error shall also be set as shown above, this bit shall be set without waiting for 64 SD clock cycles because the command will be aborted by the SMIHC Controller.</p> <p>1 – Timed out 0 – No Error</p> |

*- Write-1-to-clear status

452 . Error Interrupt Status Register Description

SMIH Normal Interrupt Status Enable Register

| Bit | Access | Function | POR Value | Description |
|------|--------|---|-----------|--|
| 15 | R | Fixed to 0 | 0 | The Host Driver shall control error interrupts using the Error Interrupt Status Enable register. |
| 14 | R/W | Boot Done Interrupt Status Enable | 0 | 1 – Enabled 0 – Masked |
| 13 | R/W | Boot Ack Complete Interrupt Status Enable | 0 | 1 – Enabled 0 – Masked |
| 12:9 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------------|-----------|---|
| 8 | R/W | Card Interrupt Status Enable | 0 | If this bit is set to 0, the SMIHC Controller shall clear interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver may clear the Card Interrupt Status Enable before servicing the Card Interrupt and may set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 1 – Enabled 0 – Masked |
| 7 | R/W | Card Removal Status Enable | 0 | 1 – Enabled 0 – Masked |
| 6 | R/W | Card Insertion Status Enable | 0 | 1 – Enabled 0 – Masked |
| 5 | R/W | Buffer Read Ready Status Enable | 0 | 1 – Enabled 0 – Masked |
| 4 | R/W | Buffer Write Ready Status Enable | 0 | 1 – Enabled 0 – Masked |
| 3 | R/W | DMA Interrupt Status Enable | 0 | 1 – Enabled 0 – Masked |
| 2 | R/W | Block Gap Event Status Enable | 0 | 1 – Enabled 0 – Masked |
| 1 | R/W | Transfer Complete Status Enable | 0 | 1 – Enabled 0 – Masked |
| 0 | R/W | Command Complete Status Enable | 0 | 1 – Enabled 0 – Masked |

453 . Normal Interrupt Status Enable Register Description

SMIH Error Interrupt Status Enable Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------|-----------|---------------------------|
| 15:11 | R/W | Reserved | 0 | Reserved |
| 10 | R/W | Tuning Error Status Enable | 0 | 1 – Enabled 0 – Masked |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|-------------|
| 9 | R/W | ADMA Error | 0 | 1 – Enabled |
| | | Status Enable | | 0 – Masked |
| 8 | R/W | Auto CMD Error | 0 | 1 – Enabled |
| | | Status Enable | | 0 – Masked |
| 7 | R/W | Current Limit | 0 | 1 – Enabled |
| | | Error Status Enable | | 0 – Masked |
| 6 | R/W | Data End Bit Error | 0 | 1 – Enabled |
| | | Status Enable | | 0 – Masked |
| 5 | R/W | Data CRC Error | 0 | 1 – Enabled |
| | | Status Enable | | 0 – Masked |
| 4 | R/W | Data Timeout | 0 | 1 – Enabled |
| | | Error Status Enable | | 0 – Masked |
| 3 | R/W | Command Index | 0 | 1 – Enabled |
| | | Error Status Enable | | 0 – Masked |
| 2 | R/W | Command End Bit | 0 | 1 – Enabled |
| | | Error Status Enable | | 0 – Masked |
| 1 | R/W | Command CRC | 0 | 1 – Enabled |
| | | Error Status Enable | | 0 – Masked |
| 0 | R/W | Command Timeout | 0 | 1 – Enabled |
| | | Error Status Enable | | 0 – Masked |

454 . Error Interrupt Status Register Description

SMIH Normal Interrupt Signal Enable Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|---|-----------|--|
| 15 | R | Fixed to 0 | 0 | The Host Driver shall control error interrupts using the Error Interrupt Signal Enable register. |
| 14 | R | Boot Done Interrupt Signal Enable | 1 | 1 – Enabled |
| 13 | R | Boot Ack Complete Interrupt 1 Signal Enable | | 1 – Enabled |
| 12 | R/W | Re-Tuning Event | 0 | 1 – Enabled |
| | | Signal Enable | | 0 – Masked |
| 11 | R/W | INT_C Signal Enable | 0 | 1 – Enabled |
| | | | | 0 – Masked |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------------------------|-----------|---------------------------|
| 10 | R/W | INT_B Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 9 | R/W | INT_A Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 8 | R/W | Card Interrupt Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 7 | R/W | Card Removal Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 6 | R/W | Card Insertion Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 5 | R/W | Buffer Read Ready Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 4 | R/W | Buffer Write Ready Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 3 | R/W | DMA Interrupt Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 2 | R/W | Block Gap Event Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 1 | R/W | Transfer Complete Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 0 | R/W | Command Complete Signal Enable | 0 | 1 – Enabled 0 – Masked |

455 . Normal Interrupt Signal Enable Register Description

SMIH Error Interrupt Signal Enable Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------------|-----------|---------------------------|
| 15:11 | R/W | Reserved | 0 | Reserved |
| 10 | R/W | Tuning Error Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 9 | R/W | ADMA Error Signal Enable | 0 | 1 – Enabled 0 – Masked |
| 8 | R/W | Auto CMD Error Signal Enable | 0 | 1 – Enabled 0 – Masked |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|---------------------|------------------|--------------------|
| 7 | R/W | Current Limit | 0 | 1 – Enabled |
| | | Error Signal Enable | | 0 – Masked |
| 6 | R/W | Data End Bit Error | 0 | 1 – Enabled |
| | | Signal Enable | | 0 – Masked |
| 5 | R/W | Data CRC Error | 0 | 1 – Enabled |
| | | Signal Enable | | 0 – Masked |
| 4 | R/W | Data Timeout | 0 | 1 – Enabled |
| | | Error Signal Enable | | 0 – Masked |
| 3 | R/W | Command Index | 0 | 1 – Enabled |
| | | Error Signal Enable | | 0 – Masked |
| 2 | R/W | Command End Bit | 0 | 1 – Enabled |
| | | Error Signal Enable | | 0 – Masked |
| 1 | R/W | Command CRC | 0 | 1 – Enabled |
| | | Error Signal Enable | | 0 – Masked |
| 0 | R/W | Command Timeout | 0 | 1 – Enabled |
| | | Error Signal Enable | | 0 – Masked |

456 . Error Interrupt Signal Enable Register Description

SMIH Auto CMD Error Status Register

| Bit | Access | Function | POR Value | Description |
|------------|---------------|--|------------------|--|
| 15:8 | R | Reserved | 0 | Reserved |
| 7 | R | Command Not Issued By Auto CMD12 Error | 0 | <p>Setting this bit to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register.</p> <p>This bit is set to 0 when Auto CMD Error is generated by Auto CMD23.</p> <p>1 – Not Issued</p> <p>0 – No Error</p> |
| 6:5 | R | Reserved | 0 | Reserved |
| 4 | R | Auto CMD Index Error | 0 | <p>This bit is set if the Command Index error occurs in response to a command</p> <p>1 – Error</p> <p>0 – No Error</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------------|-----------|---|
| 3 | R | Auto CMD End Bit Error | 0 | <p>This bit is set when detecting that the end bit of command response is 0.</p> <p>1 – End Bit Error Generated</p> <p>0 – No Error</p> |
| 2 | R | Auto CMD CRC Error | 0 | <p>This bit is set when detecting a CRC error in the command response</p> <p>1 – CRC Error Generated</p> <p>0 – No Error</p> |
| 1 | R | Auto CMD Timeout Error | 0 | <p>This bit is set if no response is returned within 64 SDCLK cycles from the end bit of command.</p> <p>If this bit is set to 1, the other error status bits (D04-D02) are meaningless.</p> <p>1 – Time Error</p> <p>0 – No Error</p> |
| 0 | R | Auto CMD12 Not Executed | 0 | <p>If memory multiple block data transfer is not started due to command error, this bit is not set because it is not necessary to issue Auto CMD12.</p> <p>Setting this bit to 1 means the SMIHC Controller cannot issue Auto CMD12 to stop memory multiple block data transfer due to some error.</p> <p>If this bit is set to 1, other error status bits (D04-D01) are meaningless.</p> <p>This bit is set to 0 when Auto CMD Error is generated by Auto CMD23.</p> <p>1 – Not Executed</p> <p>0 – Executed</p> |

457 . Auto CMD Error Status Register Description

SMIH Host Control 2 Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 15 | R/W | Preset Value Enable | 0 | <p>As the operating SDCLK frequency and I/O driver strength depend on the Host System implementation, it is difficult to determine these parameters in the Standard Host Driver. When Preset Value Enable is set, automatic SDCLK frequency generation and driver strength selection is performed without considering system specific conditions. This bit enables the functions defined in the Preset Value registers.</p> <p>1 – Automatic Selection by Preset Value are Enabled</p> <p>0 – SDCLK and Driver Strength are controlled by Host Driver</p> <p>If this bit is set to 0, SDCLK Frequency Select, Clock Generator Select in the Clock Control register and Driver Strength Select in Host Control 2 register are set by Host Driver.</p> <p>If this bit is set to 1, SDCLK Frequency Select, Clock Generator Select in the Clock Control register and Driver Strength Select in Host Control 2 register are set by SMIHC Controller as specified in the Preset Value registers.</p> |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-------------------------------------|------------------|---|
| 14 | R/W | Asynchronous Interrupt Enable | 0 | <p>This bit can be set to 1 if a card that supports asynchronous interrupts and Asynchronous Interrupt Support is set to 1 in the Capabilities register.</p> <p>Asynchronous interrupt is effective when DAT[1] interrupt is used in 4-bit SD mode (and zero is set to Interrupt Pin Select in the Shared Bus Control register).</p> <p>If this bit is set to 1, the Host Driver can stop the SDCLK during asynchronous interrupt period to save power. During this period, the SMIHC Controller continues to deliver the Card Interrupt to the host when it is asserted by the Card.</p> <p>1 – Enabled 0 – Disabled</p> |
| 13:9 | R/W | Reserved | 0 | Reserved |
| 8 | R/W | Upper bit of Driver Strength Select | 0 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 04-05 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 09,05,04 |
| 7 | R/W | Sampling Clock Select | 0 | <p>SMIHC Controller uses this bit to select sampling clock to receive CMD and DAT. This bit is set by tuning procedure and valid after the completion of tuning (when Execute Tuning is cleared). Setting 1 means that tuning is completed successfully and setting 0 means that tuning is failed.</p> <p>Writing 1 to this bit is meaningless and ignored. A tuning circuit is reset by writing to 0. This bit can be cleared with setting Execute Tuning.</p> <p>Once the tuning circuit is reset, it will take time to complete tuning sequence. Therefore, Host Driver should keep this bit to 1 to perform re-tuning sequence</p> <p>to complete re-tuning sequence in a short time. Change of this bit is not allowed while the SMIHC Controller is receiving response or a read data block.</p> <p>1 – Tuned clock is used to sample data 0 – Fixed clock is used to sample data</p> |

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|-------------------------------------|------------------------|-----------|--|-----------------|-------------------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 6 | R/W* | Execute Tuning | 0 | <p>This bit is set to 1 to start tuning procedure and automatically cleared when tuning procedure is completed.</p> <p>The result of tuning is indicated to Sampling Clock Select. Tuning procedure is aborted by writing 0. for more detail about tuning procedure.</p> <p>1 – Execute Tuning 0 – Not Tuned or Tuning Completed</p> | | | | | | | | |
| 5:4 | R/W | Driver Strength Select | 0 | <p>SMIHC Controller output driver in 1.8V signaling is selected by this bit. In 3.3V signaling, this field is not effective.</p> <p>This field can be set depends on Driver Type A, C and D support bits in the Capabilities register. This bit depends on setting of Preset Value Enable.</p> <p>If Preset Value Enable = 0, this field is set by Host Driver.</p> <p>If Preset Value Enable = 1, this field is automatically set by a value specified in the one of Preset Value registers.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>00_b</td><td>Driver Type B is Selected (Default)</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> </table> | 00 _b | Driver Type B is Selected (Default) | 01 _b | Driver Type A is Selected | 10 _b | Driver Type C is Selected | 11 _b | Driver Type D is Selected |
| 00 _b | Driver Type B is Selected (Default) | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|--|
| 3 | R/W | 1.8V Signaling Enable | 0 | <p>This bit controls voltage regulator for I/O cell. 3.3V is supplied to the card regardless of signaling voltage. Setting this bit from 0 to 1 starts changing signal voltage from 3.3V to 1.8V. 1.8V regulator output shall be stable within 5ms. SMIHC Controller clears this bit if switching to 1.8V signaling fails.</p> <p>Clearing this bit from 1 to 0 starts changing signal voltage from 1.8V to 3.3V. 3.3V regulator output shall be stable within 5ms. Host Driver can set this bit to 1 when SMIHC Controller supports 1.8V signaling (One of support bits is set to 1: SDR50, SDR104 or DDR50 in the Capabilities register) and the card or device supports UHS-I (S18R=1). This bit depends on setting of Preset Value Enable.</p> <p>If Preset Value Enable = 0, this field is set by Host Driver.</p> <p>If Preset Value Enable = 1, this field is automatically set by a value specified in the one of Preset Value registers.</p> <p>1 – 1.8V Signaling 0 – 3.3V Signaling</p> |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | |
|------------------|----------|-----------------|-----------|---|------------------|--------|------------------|--------|------------------|--------|------------------|---------|------------------|--------|------------------|-------|------------------|----------|------------------|----------|
| 2:0 | R/W | UHS Mode Select | 0 | <p>This field is used to select one of UHS-I modes and effective when 1.8V Signaling Enable is set to 1.</p> <p>If Preset Value Enable in the Host Control 2 register is set to 1, SMIHC Controller sets SDCLK Frequency Select, Clock</p> <p>Generator Select in the Clock Control register and Driver Strength Select according to Preset Value registers. In this case,</p> <p>one of preset value registers is selected by this field. Host Driver needs to reset SD Clock Enable before changing this field to avoid generating clock glitch. After setting this field, Host Driver sets SD Clock Enable again.</p> <table border="1" style="margin-left: 20px;"> <tr><td>000_b</td><td>SDR 12</td></tr> <tr><td>001_b</td><td>SDR 25</td></tr> <tr><td>010_b</td><td>SDR 50</td></tr> <tr><td>011_b</td><td>SDR 104</td></tr> <tr><td>100_b</td><td>DDR 50</td></tr> <tr><td>101_b</td><td>HS400</td></tr> <tr><td>110_b</td><td>Reserved</td></tr> <tr><td>111_b</td><td>Reserved</td></tr> </table> <p>When SDR50, SDR104 or DDR50 is selected for SDIO card, interrupt detection at the block gap shall not be used.</p> <p>Read Wait timing is changed for these modes.</p> | 000 _b | SDR 12 | 001 _b | SDR 25 | 010 _b | SDR 50 | 011 _b | SDR 104 | 100 _b | DDR 50 | 101 _b | HS400 | 110 _b | Reserved | 111 _b | Reserved |
| 000 _b | SDR 12 | | | | | | | | | | | | | | | | | | | |
| 001 _b | SDR 25 | | | | | | | | | | | | | | | | | | | |
| 010 _b | SDR 50 | | | | | | | | | | | | | | | | | | | |
| 011 _b | SDR 104 | | | | | | | | | | | | | | | | | | | |
| 100 _b | DDR 50 | | | | | | | | | | | | | | | | | | | |
| 101 _b | HS400 | | | | | | | | | | | | | | | | | | | |
| 110 _b | Reserved | | | | | | | | | | | | | | | | | | | |
| 111 _b | Reserved | | | | | | | | | | | | | | | | | | | |

* - automatic clear bit

458 . Host Control 2 Register Description

SMIH Capabilities Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 63:48 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | |
|-------|--------|---------------------------|-----------|--|-----------------------------------|--|------------------------|
| 47:46 | R* | Re-Tuning Modes | 0 | This field selects re-tuning method and limits the maximum data length. | | | |
| | | | | Bit4 7-46 | Re- Tuning Mode | Re-Tuning Method | Data Length |
| | | | | 00 _b | Mode1 | Timer | 4MB (Max.) |
| | | | | 01 _b | Mode2 | Timer and Re-Tuning Request | 4MB (Max.) |
| | | | | 10 _b | Mode3 | Auto Re-Tuning Any (for transfer) Timer and Re-Tuning Request | |
| | | | | 11 _b | Reserved | | |
| | | | | There are two re-tuning timings: Re-Tuning Request controlled by the SMIHC Controller and expiration of a Re-Tuning Timer controlled by the Host. | | | |
| 45 | R | Use Tuning for SDR50 | 1 | As this bit is set to 1, before using the SDR50 mode, the tuning procedure at the initialization sequence will be executed regardless of Re-Tuning Modes state in the Capabilities register. | | | |
| 44 | R | Reserved | 0 | Reserved | | | |
| 43:40 | R* | Timer Count for Re-Tuning | 1 | This field indicates an initial value of the Re-Tuning Timer for Re-Tuning Mode 1 to 3. Setting to 0 disables Re-Tuning Timer. | | | |
| | | | | 0 _h | Re-Tuning Timer disabled | | |
| | | | | 1 _h | 1 seconds | | |
| | | | | 2 _h | 2 seconds | | |
| | | | | 3 _h | 4 seconds | | |
| | | | | 4 _h | 8 seconds | | |
| | | | | | | | |
| | | | | n | $2^{(n-1)}$ seconds | | |
| | | | | | | | |
| | | | | B _h | 1024 seconds | | |
| | | | | E _h -C _h | Reserved | | |
| | | | | F _h | Get information from other source | | |
| 39 | R* | Reserved | 0 | Reserved | | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|---|
| 38 | R* | Driver Type D Support | 1 | This bit indicates support of Driver Type D for 1.8 Signaling. 1 – Driver Type D is Supported 0 – Driver Type D is Not Supported |
| 37 | R* | Driver Type C Support | 1 | This bit indicates support of Driver Type C for 1.8 Signaling. 1 – Driver Type C is Supported 0 – Driver Type C is Not Supported |
| 36 | R* | Driver Type A Support | 1 | This bit indicates support of Driver Type A for 1.8 Signaling. 1 – Driver Type A is Supported 0 – Driver Type A is Not Supported |
| 35 | R* | HS200 Support | 1 | 1 – HS200 is Supported 0 – HS200 is Not Supported |
| 34 | R* | DDR50 Support | 1 | 1 – DDR50 is Supported 0 – DDR50 is Not Supported |
| 33 | R* | SDR104 Support | 1 | SDR104 requires tuning. 1 – SDR104 is Supported 0 – SDR104 is Not Supported |
| 32 | R* | SDR50 Support | 1 | If SDR104 is supported, bit shall be set to 1. Bit 45 indicates whether SDR50 requires tuning or not. 1 – SDR50 is Supported 0 – SDR50 is Not Supported |

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------------------|-----------|--|
| 31:30 | R* | Slot Type | 0 | <p>This field indicates usage of a slot by a specific Host System. (A SMIHC Controller register set is defined per slot.)</p> <p>Embedded slot for one device (01_b) means that only one non-removable device is connected to a SD bus slot. Shared Bus Slot (10_b) can be set if SMIHC Controller supports Shared Bus Control register. The Standard Host Driver controls only a removable card or one embedded device is connected to a SD bus slot. If a slot is configured for shared bus (10_b), the Standard Host Driver does not control embedded devices connected to a shared bus. Shared bus slot is controlled by a specific host driver developed by a Host System.</p> <p>00_b – Removable Card Slot 01_b – Embedded Slot for One Device 10_b – Shared Bus Slot 11_b – Reserved</p> |
| 29 | R* | Asynchronous Interrupt Support | 1 | <p>1 – Asynchronous Interrupt Supported 0 – Asynchronous Interrupt Not Supported</p> |
| 28 | R* | System Bus Support | 0 | <p>64-bit System Bus Support 64-bit address system bus is not supported</p> |
| 27 | R | Voltage Support 1.2V | 1 | <p>Embedded system can be used 1.2V power supply. 1 – 1.2V Supported 0 – 1.2V Not Supported</p> |
| 26 | R* | Voltage Support 1.8V | 1 | <p>Embedded system can be used 1.8V power supply. 1 – 1.8V Supported 0 – 1.8V Not Supported</p> |
| 25 | R* | Voltage Support 3.0V | 1 | <p>1 – 3.0V Supported 0 – 3.0V Not Supported</p> |
| 24 | R* | Voltage Support 3.3V | 1 | <p>1 – 3.3V Supported 0 – 3.3V Not Supported</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------------|-----------|---|
| 23 | R* | Suspend/Resume Support | 0 | <p>This bit indicates whether the SMIHC Controller does not support Suspend/Resume functionality.</p> <p>If this bit is 0, the Host Driver shall not issue either Suspend or Resume commands because the Suspend and Resume mechanism is not supported.</p> <p>1 – Supported 0 – Not Supported</p> |
| 22 | R* | SDMA Support | 1 | <p>This bit indicates whether the SMIHC Controller is capable of using SDMA to transfer data between system memory and the SMIHC Controller directly.</p> <p>1 – SDMA Supported 0 – SDMA Not Supported</p> |
| 21 | R* | High Speed Support | 1 | <p>This bit indicates whether the SMIHC Controller and the Host System support High Speed mode and they can supply SD Clock frequency from 25MHz to 50MHz.</p> <p>1 – High Speed Supported 0 – High Speed Not Supported</p> |
| 20 | R* | Reserved | 0 | <p>(New assignment is not allowed)</p> <p>This bit is reserved for backward compatibility with prior specifications. If set, the SMIHC Controller is indicating that it supports legacy ADMA1 mode.</p> <p>Host drivers are not required to support this mode.</p> |
| 19 | R* | ADMA2 Support | 1 | <p>This bit indicates whether the SMIHC Controller is capable of using ADMA2.</p> <p>1 – ADMA2 Supported 0 – ADMA2 Not Supported</p> |
| 18 | R* | 8-bit Support for Embedded Device | 1 | <p>This bit indicates whether the SMIHC Controller is capable of using 8-bit bus width mode. This bit is not effective when slot Type is set to 10_b.</p> <p>In this case, refer to Bus Width Preset in the shared bus register.</p> <p>1 – 8-bit Bus Width Supported 0 – 8-bit Bus Width Not Supported</p> |

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-------|------------|------------------|-----------|---|----|------------|----|------|----|------|----|----------|
| 17:16 | R* | Max Block Length | 0 | <p>This value indicates the maximum block size that the Host Driver can read and write to the buffer in the SMIHC Controller. The buffer shall transfer as indicated below. It is noted that transfer block length shall always be 512 bytes for SD Memory Cards regardless of this field.</p> <table border="1" data-bbox="897 617 1056 752"> <tr><td>00</td><td>512(bytes)</td></tr> <tr><td>01</td><td>1024</td></tr> <tr><td>10</td><td>2048</td></tr> <tr><td>11</td><td>Reserved</td></tr> </table> | 00 | 512(bytes) | 01 | 1024 | 10 | 2048 | 11 | Reserved |
| 00 | 512(bytes) | | | | | | | | | | | |
| 01 | 1024 | | | | | | | | | | | |
| 10 | 2048 | | | | | | | | | | | |
| 11 | Reserved | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | | | | | |
|------------------------|------------------------------------|--------------------------------------|-----------|---|------------------------|---------------|------------------------|-------|-------|-------|------------------------|------|------------------------|------|------------------------|------------------------------------|-----------------|--------|-------|-------|-----------------|------|-----------------|------|-----------------|------------------------------------|
| 15:8 | R* | Base Clock Frequency For SD Clock | 0xC8 | <p>This value indicates the base (maximum) clock frequency for the SD Clock. Definition of this field depends on Host Controller Version.</p> <p>(1) 6-bit Base Clock Frequency</p> <p>This mode is supported by the Host Controller Version 1.00 and 2.00. Upper 2-bit is not effective and always 0. Unit values are 1MHz.</p> <p>The supported clock range is 10MHz to 63MHz.</p> <table border="1"> <tr><td>11xx xxxx_b</td><td>Not supported</td></tr> <tr><td>0011 1111_b</td><td>63MHz</td></tr> <tr><td>.....</td><td>.....</td></tr> <tr><td>0000 0010_b</td><td>2MHz</td></tr> <tr><td>0000 0001_b</td><td>1MHz</td></tr> <tr><td>0000 0000_b</td><td>Get information via another method</td></tr> </table> <p>(2) 8-bit Base Clock Frequency</p> <p>This mode is supported by the SMIHC Controller. Unit values are 1MHz. The supported clock range is 10MHz to 255MHz.</p> <table border="1"> <tr><td>FF_h</td><td>255MHz</td></tr> <tr><td>.....</td><td>.....</td></tr> <tr><td>02_h</td><td>2MHz</td></tr> <tr><td>01_h</td><td>1MHz</td></tr> <tr><td>00_h</td><td>Get information via another method</td></tr> </table> <p>If the real frequency is 16.5MHz, the larger value shall be set 0001 0001_b (17MHz) because the Host Driver use this value to calculate the clock divider value (Refer to the SDCLK Frequency Select in the Clock Control register) and it shall not exceed upper limit of the SD Clock frequency.</p> <p>If these bits are all 0, the Host System has to get information via another method.</p> | 11xx xxxx _b | Not supported | 0011 1111 _b | 63MHz | | | 0000 0010 _b | 2MHz | 0000 0001 _b | 1MHz | 0000 0000 _b | Get information via another method | FF _h | 255MHz | | | 02 _h | 2MHz | 01 _h | 1MHz | 00 _h | Get information via another method |
| 11xx xxxx _b | Not supported | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0011 1111 _b | 63MHz | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0000 0010 _b | 2MHz | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0000 0001 _b | 1MHz | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0000 0000 _b | Get information via another method | | | | | | | | | | | | | | | | | | | | | | | | | |
| FF _h | 255MHz | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 02 _h | 2MHz | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 _h | 1MHz | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 _h | Get information via another method | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | R* | Timeout Clock Unit | 1 | <p>This bit shows the unit of base clock frequency used to detect Data Timeout Error.</p> <p>0 – KHz</p> <p>1 – MHz</p> | | | | | | | | | | | | | | | | | | | | | | |
| 6 | R | Reserved | 1 | Reserved | | | | | | | | | | | | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------------|-----------|--|
| 5:0 | R* | Timeout Clock Frequency | 0x32 | <p>This bit shows the base clock frequency used to detect Data Timeout Error. The Timeout Clock Unit defines the unit of this field's value.</p> <p>Timeout Clock Unit =0 [KHz] unit: 1KHz to 63KHz</p> <p>Timeout Clock Unit =1 [MHz] unit: 1MHz to 63MHz</p> <p>Not 0 – 1KHz to 63KHz or 1MHz to 63MHz</p> <p>00 0000_b – Get information via another method</p> |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored

459 . Capabilities Register Description

SMIH Maximum Current Capabilities Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------------|-----------|---|
| 63:56 | R | Reserved | 0 | Reserved |
| 55:48 | R | Reserved | 0 | Reserved |
| 47:40 | R | Reserved | 0 | Reserved |
| 39:32 | R | Reserved | 0 | Reserved |
| 31:24 | R* | Maximum Current for 1.2V | 0x2 | Indicates maximum current capability for 1.2V. The value is meaningful if Voltage Support 1.2V is set in the Capabilities register. |
| 23:16 | R* | Maximum Current for 1.8V | 0x2 | Indicates maximum current capability for 1.8V. The value is meaningful if Voltage Support 1.8V is set in the Capabilities register. |
| 15:8 | R* | Maximum Current for 3.0V | 0x2 | Indicates maximum current capability for 3.0V. The value is meaningful if Voltage Support 3.0V is set in the Capabilities register. |
| 7:0 | R* | Maximum Current for 3.3V | 0x2 | Indicates maximum current capability for 3.3V. The value is meaningful if Voltage Support 3.3V is set in the Capabilities register. |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored

460 . Maximum Current Capabilities Register Description

SMIH ADMA Error Status Register

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|-------------|
| 7:3 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | |
|-----------|---|-------------------------------------|-----------|--|-----------|---|------------------------------|----|--------------------|-------------------------------------|----|---------------------------|-----------------------------|----|----------------------|------------|----|------------------------|-------------------------------------|
| 2 | R | ADMA Length Mismatch Error | 0 | <p>This error occurs in the following 2 cases:</p> <p>(1) While Block Count Enable being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length.</p> <p>(2) Total data length cannot be divided by the block length.</p> <p>1 – Error 0 – No Error</p> | | | | | | | | | | | | | | | |
| 1:0 | R | ADMA Error State | 0 | <p>This field indicates the state of ADMA when error is occurred during ADMA data transfer. This field never indicates "10" because ADMA never stops in this state.</p> <table border="1" data-bbox="809 808 1429 1251"> <thead> <tr> <th>D01 – D00</th> <th>ADMA Error State when error is occurred</th> <th>Contents of SYS_SDR register</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>ST_STOP (Stop DMA)</td> <td>Points next of the error descriptor</td> </tr> <tr> <td>01</td> <td>ST_FDS (Fetch Descriptor)</td> <td>Points the error descriptor</td> </tr> <tr> <td>10</td> <td>Never set this state</td> <td>(Not used)</td> </tr> <tr> <td>11</td> <td>ST_TFR (Transfer Data)</td> <td>Points next of the error descriptor</td> </tr> </tbody> </table> | D01 – D00 | ADMA Error State when error is occurred | Contents of SYS_SDR register | 00 | ST_STOP (Stop DMA) | Points next of the error descriptor | 01 | ST_FDS (Fetch Descriptor) | Points the error descriptor | 10 | Never set this state | (Not used) | 11 | ST_TFR (Transfer Data) | Points next of the error descriptor |
| D01 – D00 | ADMA Error State when error is occurred | Contents of SYS_SDR register | | | | | | | | | | | | | | | | | |
| 00 | ST_STOP (Stop DMA) | Points next of the error descriptor | | | | | | | | | | | | | | | | | |
| 01 | ST_FDS (Fetch Descriptor) | Points the error descriptor | | | | | | | | | | | | | | | | | |
| 10 | Never set this state | (Not used) | | | | | | | | | | | | | | | | | |
| 11 | ST_TFR (Transfer Data) | Points next of the error descriptor | | | | | | | | | | | | | | | | | |

461 . ADMA Error Status Register Description

SMIH ADMA System Address Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 63:32 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | |
|--------------------------------|-------------------------------|---------------------|-----------|--|----------------|-----------------------|--------------------------------|-----------------------|--------------------------------|-----------------------|--------------------------------|-----------------------|--------------------------------|-----------------------|------|------|-------------------------------|-------------------------------|
| 31:0 | R/W | ADMA System Address | 0 | <p>This register holds byte address of executing command of the Descriptor table. 32-bit Address Descriptor uses lower 32-bit of this register.</p> <p>At the start of ADMA, the Host Driver shall set start address of the Descriptor table. The ADMA increments this register address, which points to next line, when every fetching a Descriptor line. When the ADMA Error Interrupt is generated, this register shall hold valid Descriptor address depending on the ADMA state. The Host Driver shall program Descriptor Table on 32-bit boundary and set 32-bit boundary address to this register.</p> <p>ADMA2 ignores lower 2-bit of this register and assumes it to be 00_b.</p> <p>32-bit Address ADMA</p> <table border="1" data-bbox="857 932 1389 1224"> <thead> <tr> <th>Register Value</th><th>64-bit System Address</th></tr> </thead> <tbody> <tr><td>xxxxxxxx 00000000_h</td><td>00000000_h</td></tr> <tr><td>xxxxxxxx 00000004_h</td><td>00000004_h</td></tr> <tr><td>xxxxxxxx 00000008_h</td><td>00000008_h</td></tr> <tr><td>xxxxxxxx 0000000C_h</td><td>0000000C_h</td></tr> <tr><td>....</td><td>....</td></tr> <tr><td>FFFFFFF FFFFFFFC_h</td><td>FFFFFFF FFFFFFFC_h</td></tr> </tbody> </table> | Register Value | 64-bit System Address | xxxxxxxx 00000000 _h | 00000000 _h | xxxxxxxx 00000004 _h | 00000004 _h | xxxxxxxx 00000008 _h | 00000008 _h | xxxxxxxx 0000000C _h | 0000000C _h | | | FFFFFFF FFFFFFFC _h | FFFFFFF FFFFFFFC _h |
| Register Value | 64-bit System Address | | | | | | | | | | | | | | | | | |
| xxxxxxxx 00000000 _h | 00000000 _h | | | | | | | | | | | | | | | | | |
| xxxxxxxx 00000004 _h | 00000004 _h | | | | | | | | | | | | | | | | | |
| xxxxxxxx 00000008 _h | 00000008 _h | | | | | | | | | | | | | | | | | |
| xxxxxxxx 0000000C _h | 0000000C _h | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | |
| FFFFFFF FFFFFFFC _h | FFFFFFF FFFFFFFC _h | | | | | | | | | | | | | | | | | |

462 .ADMA System Address Register Description

SMIH Preset Value Register0

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|--|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1" data-bbox="889 1583 1214 1740"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------------|-----------|--|
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> |
| 9:0 | R* | SDCLK Frequency Select Value | 0 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

463 . Preset Value Register0 Description

SMIH Preset Value Register1

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|--|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1" style="margin-left: 20px;"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> | | | | | | | | |
| 9:0 | R* | SDCLK Frequency Select Value | 0x4 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. | | | | | | | | |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

464 . Preset Value Register1 Description

SMIH Preset Value Register2

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|---|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> | | | | | | | | |
| 9:0 | R* | SDCLK Frequency Select Value | 0x2 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. | | | | | | | | |

* - Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

465 . Preset Value Register2 Description

SMIH Preset Value Register3

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|---|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------------|-----------|--|
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> |
| 9:0 | R* | SDCLK Frequency Select Value | 0x4 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

466 . Preset Value Register3 Description

SMIH Preset Value Register4

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|---|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> | | | | | | | | |
| 9:0 | R* | SDCLK Frequency Select Value | 0x2 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. | | | | | | | | |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

467 . Preset Value Register4 Description

SMIH Preset Value Register5

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|---|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> | | | | | | | | |
| 9:0 | R* | SDCLK Frequency Select Value | 0x1 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. | | | | | | | | |

* - Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

468 . Preset Value Register5 Description

SMIH Preset Value Register6

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|---|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------------|-----------|--|
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> |
| 9:0 | R* | SDCLK Frequency Select Value | 0x0 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

469 . Preset Value Register6 Description

SMIH Preset Value Register7

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|-----------------|---------------------------|-------------------------------------|-----------|--|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|-----------------|---------------------------|
| 15:14 | R* | Driver Strength Select Value | 0x3 | <p>Driver Strength is supported by 1.8V signaling bus speed modes.</p> <p>This field is meaningless for 3.3V signaling.</p> <table border="1" style="margin-left: 20px;"> <tr><td>11_b</td><td>Driver Type D is Selected</td></tr> <tr><td>10_b</td><td>Driver Type C is Selected</td></tr> <tr><td>01_b</td><td>Driver Type A is Selected</td></tr> <tr><td>00_b</td><td>Driver Type B is Selected</td></tr> </table> | 11 _b | Driver Type D is Selected | 10 _b | Driver Type C is Selected | 01 _b | Driver Type A is Selected | 00 _b | Driver Type B is Selected |
| 11 _b | Driver Type D is Selected | | | | | | | | | | | |
| 10 _b | Driver Type C is Selected | | | | | | | | | | | |
| 01 _b | Driver Type A is Selected | | | | | | | | | | | |
| 00 _b | Driver Type B is Selected | | | | | | | | | | | |
| 13 | R* | Upper bit of Driver Strength Select | 1 | This bit reflects the SMIH_DRIVE_STRENGTH[2] in association with 14-15 bits of Drive Strength select SMIH_DRIVE_STRENGTH[2:0] = 13,15,14 | | | | | | | | |
| 12:11 | R* | Reserved | 0 | Reserved | | | | | | | | |
| 10 | R* | Clock Generator Select Value | 0 | <p>This bit is effective when Host Controller supports programmable clock generator.</p> <p>1 – Programmable Clock Generator</p> <p>0 – Host Controller Ver2.00 Compatible Clock Generator</p> | | | | | | | | |
| 9:0 | R* | SDCLK Frequency Select Value | 0 | 10-bit preset value to set SDCLK Frequency Select in the Clock Control Register is described by a host system. | | | | | | | | |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

470 . Preset Value Register7 Description

SMIH Tx Tune Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 31:10 | R/W | Reserved | 0 | Reserved |
| 9:7 | R/W | Increment tap point value | 1 | Increment value When Auto increment bit6 is set to 1, Host controller will increment Tap point based on the value programmed in this field. |
| 6 | R/W | Auto increment | 0 | Increment locally or not. |
| 5:0 | R/W | Tap point value | 0 | Tap point value |

471 . Tx Tune Register Description

SMIH Rx Tune Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 31:10 | R/W | Reserved | 0 | Reserved |
| 9:7 | R/W | Increment tap point value | 1 | Increment value When Auto increment bit6 is set to 1, Host controller will increment Tap point based on the value programmed in this field. |
| 6 | R/W | Auto increment | 0 | Increment locally or not. |
| 5:0 | R/W | Tap point value | 0 | Tap point value |

472 . Rx Tune Register Description

SMIH Ds Tune Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 31:10 | R/W | Reserved | 0 | Reserved |
| 9:7 | R/W | Increment tap point value | 1 | Increment value When Auto increment bit6 is set to 1, Host controller will increment Tap point based on the value programmed in this field. |
| 6 | R/W | Auto increment | 0 | Increment locally or not. |
| 5:0 | R/W | Tap point value | 0 | Tap point value |

473 . Ds Tune Register Description

SMIH AHB Burst Size Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 15:7 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | |
|--------|--------|-----------------------------------|-----------|--|--------|-------|--------|-------|--------|--------|--------|------|--------|-------|--------|-------|--------|--------|
| 6:0 | R/W | AHB Master Burst Size Register | 0x7 | <p>AHB Master performs Burst operations as per this register. The default Burst operations performed by SMIHC AHB Master is INCR4, INCR8 and INCR16.</p> <p>0 – Disabled 1 – Enabled.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>Bit 00</td><td>INCR4</td></tr> <tr><td>Bit 01</td><td>INCR8</td></tr> <tr><td>Bit 02</td><td>INCR16</td></tr> <tr><td>Bit 03</td><td>INCR</td></tr> <tr><td>Bit 04</td><td>WRAP4</td></tr> <tr><td>Bit 05</td><td>WRAP8</td></tr> <tr><td>Bit 06</td><td>WRAP16</td></tr> </table> | Bit 00 | INCR4 | Bit 01 | INCR8 | Bit 02 | INCR16 | Bit 03 | INCR | Bit 04 | WRAP4 | Bit 05 | WRAP8 | Bit 06 | WRAP16 |
| Bit 00 | INCR4 | | | | | | | | | | | | | | | | | |
| Bit 01 | INCR8 | | | | | | | | | | | | | | | | | |
| Bit 02 | INCR16 | | | | | | | | | | | | | | | | | |
| Bit 03 | INCR | | | | | | | | | | | | | | | | | |
| Bit 04 | WRAP4 | | | | | | | | | | | | | | | | | |
| Bit 05 | WRAP8 | | | | | | | | | | | | | | | | | |
| Bit 06 | WRAP16 | | | | | | | | | | | | | | | | | |

474 .AHB Burst Size Register Description

SMIH SDH Revision ID Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------|-----------|---|
| 31:24 | R | Reserved | 0 | Reserved |
| 23:16 | R | Maintenance | 0x07 | Minor BUG Fixes in HW alone |
| 15:8 | R | Minor | 0x02 | HW Changes alone No Software Changes required |
| 7:0 | R | Major | 0x03 | Both SW and HW changes required |

475 .SDH Revision ID Register Description

SMIH Slot Interrupt Status Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------|-----------|--|
| 15:1 | R/W | Reserved | 0 | Reserved |
| 0 | R | Interrupt Signal For A Slot | 0 | This status bit indicates the logical OR of Interrupt Signal and Wakeup Signal for the slot. |

476 .Slot Interrupt Status Register Description

SMIH Host Controller Version Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------|-----------|--|
| 15:8 | R* | Vendor Version Number | 0x01 | <p>This status is reserved for the vendor version number.</p> <p>The Host Driver should not use this status.</p> |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | |
|-----------------|---|------------------------------|-----------|--|-----------------|------------------------------------|-----------------|------------------------------------|--|---|-----------------|------------------------------------|-----|----------|-----|--|
| 7:0 | R* | Specification Version Number | 0x02 | <p>This status indicates the Host Controller Spec. Version.</p> <p>The upper and lower 4-bits indicate the version.</p> <table border="1"> <tr> <td>00_h</td><td>SD Host Specification Version 1.00</td></tr> <tr> <td>01_h</td><td>SD Host Specification Version 2.00</td></tr> <tr> <td></td><td>Including the feature of the ADMA and Test Register</td></tr> <tr> <td>02_h</td><td>SD Host Specification Version 3.00</td></tr> <tr> <td>oth</td><td>Reserved</td></tr> <tr> <td>ers</td><td></td></tr> </table> | 00 _h | SD Host Specification Version 1.00 | 01 _h | SD Host Specification Version 2.00 | | Including the feature of the ADMA and Test Register | 02 _h | SD Host Specification Version 3.00 | oth | Reserved | ers | |
| 00 _h | SD Host Specification Version 1.00 | | | | | | | | | | | | | | | |
| 01 _h | SD Host Specification Version 2.00 | | | | | | | | | | | | | | | |
| | Including the feature of the ADMA and Test Register | | | | | | | | | | | | | | | |
| 02 _h | SD Host Specification Version 3.00 | | | | | | | | | | | | | | | |
| oth | Reserved | | | | | | | | | | | | | | | |
| ers | | | | | | | | | | | | | | | | |

*- Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.

477 . Host Controller Version Register Description

15.6 USB 2.0

15.6.1 General Description

The Universal Serial Bus (USB) is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host scheduled, token based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation. USB is a popular standard for connecting peripherals and portable consumer electronic devices such as digital cameras and hand held computers to host PCs. The On-The-Go (OTG) Supplement to the USB Specification extends USB to peer-to-peer application. Using USB OTG technology, consumer electronics, peripherals and portable devices can connect to each other (for example, a digital camera can connect directly to a printer, or a keyboard can connect to a Personal Digital Assistant) to exchange data. With USB OTG, develop a fully USB compliant peripheral device that can also assume the role of a USB host. The OTG state machines determine the role of the device based on connector signals, and then initializes the device in the appropriate mode of operation (host or peripheral) based on how it is connected. After connecting the devices can negotiate using the OTG protocols to assume the role of host or peripheral based on the task to be accomplished.

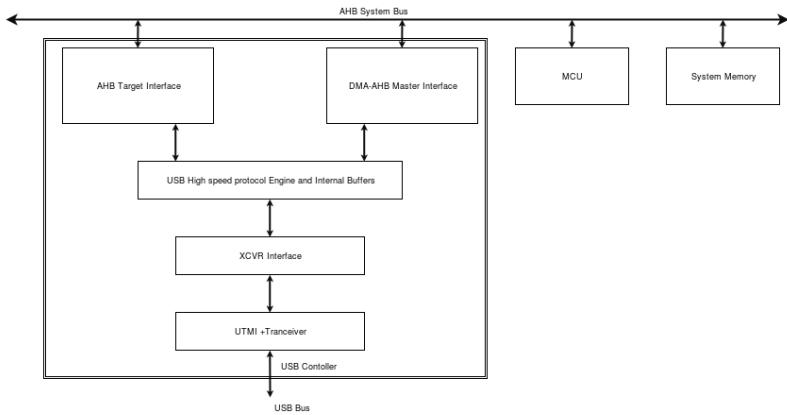
MCU supports one High-speed USB 2.0 Host/Device/OTG interface with DMA support and on-chip high-speed PHY. The USB OTG module allows to connect directly to a USB Host such as a PC (in device mode) or to a USB Device in host mode.

15.6.2 Features

- Complies with Universal Serial Bus specification 2.0.
- Complies with USB On-The-Go supplement
- Complies with Enhanced Host Controller Interface Specification
- Supports auto USB 2.0 mode discovery
- Supports all high-speed USB compliant peripherals with maximum speed of 480 Mbps
- Supports all full-speed USB compliant peripherals with maximum speed of 12Mbps
- USB-IF Certified
- Supports interrupts
- On-chip UTMI+ compliant high-speed transceiver (PHY).
- Protocol-aware DMA engine for high USB data through-put, low system CPU and system bus loading
- AMBA-AHB system bus interface for accessing from MCU

15.6.3 Functional Description

Depending on the options selected the USB core is able to act as peripheral controller, or host controller or a dual role OTG controller that is able to negotiate the host or peripheral role on the bus in compliance with the On-The-Go (OTG) supplement to the USB specification. The 32bit system bus interface contains a chaining DMA (Direct Memory Access) engine that reduces the interrupt load on the application processor, and reduces the total system bus bandwidth that must be dedicated to servicing the USB interface requirements. By transferring the data to system memory at wire rates, the buffer memory requirement within the core is minimized. The USB core also makes strategic use of the processor for tasks that do not require timing critical responses to reduce the amount of special purpose logic.



USB 2.0 Block Diagram

UTMI + Transceiver (PHY)

USB2.0 PHY is a complete transceiver that implements the USB2.0 physical layer. The transmitter uses Raw data in byte format that is converted into serial then a SYNC pattern is attached and is given to the analog front-end that drives the required pattern on the USB data lines. Meanwhile this data was bit-stuffed then NRZI encoded and an EOP pattern attached to it. The receiver hunts for a SYNC pattern, performs the NRZI decoding, bit de-stuffing and serial to parallel conversion. The clock and data are recovered from the received serial stream at FS, while at HS the receiver operates on the recovered clock. The received data is given, to the upper layer, in byte format. There are additional functions like looking for bit-stuff and alignment errors and disconnect detection.

XCVR Interface

This block interfaces with protocol engine and the UTMI+ compatible transceiver.

USB 2.0 Protocol Engine and Internal Buffers

The Protocol Engine parses all the USB tokens and generates the response packets. It is responsible for all error checking, check field generation, formatting all the handshake, ping and data response packets on the bus, and for any signals that must be generated based on a USB based time-frame. In host mode, the Protocol engine also generates all of the token packets required by the USB protocol. The Protocol engine contains several sub-functions:

- The token state machines track all of the tokens on the bus and filter the traffic based on the address and endpoint information in the token. In host mode, these state machines also generate the tokens required for data transfer and bus control.
- The CRC5 and CRC16 CRC generator/checker circuits check and generate the CRC check fields for the token and data packets.
- The data and handshake state machines generate any responses required on the USB and move the packet data through the dual port memory FIFOs to the DMA controller block.

- The Interval timers provide timing strobes that identify important bus timing events: the bus timeout interval, the micro-frame interval, the start of frame interval, and the bus reset, resume, and suspend intervals.
- Reports all transfer status to the DMA engine

DMA_AHB Master

The DMA Engine Block presents a bus initiator (master) interface to the system memory bus. It is responsible for moving all of the data to be transferred over the USB between the USB-HS core and buffers in system memory. Like the microprocessor interface block the DMA block uses a simple synchronous bus signaling protocol that eases connection to a number of different standard buses. The DMA controller must access both control information and packet data from system memory. The control information is contained in link list based queue structures. The DMA controller has state machines that are able to parse all of the data structures defined in this controller specification. In host mode, the data structures are from the EHCI specification and represent queues of transfers to be performed by the host controller, including the split transaction requests that allow an EHCI controller to direct packets to Low and Full speed devices. In device mode, the data structures designed to be similar to those in the EHCI specification are used to allow device responses to be queued for each of the active pipes in the device.

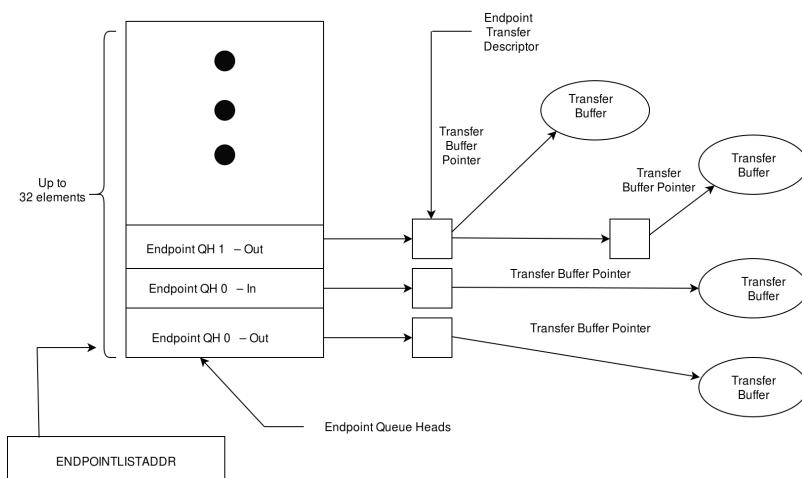
AHB Target

The AMBA-AHB slave interface supports only single accesses; the transfer size is always 32-bit long. The slave never produces Error and Retry responses. The slave transfers are simple and will always be completed.

Data Structures

Device Data Structure

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. This data structures are used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller. The data structure definitions in this support a 32bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.



End Point Queue Head Organization

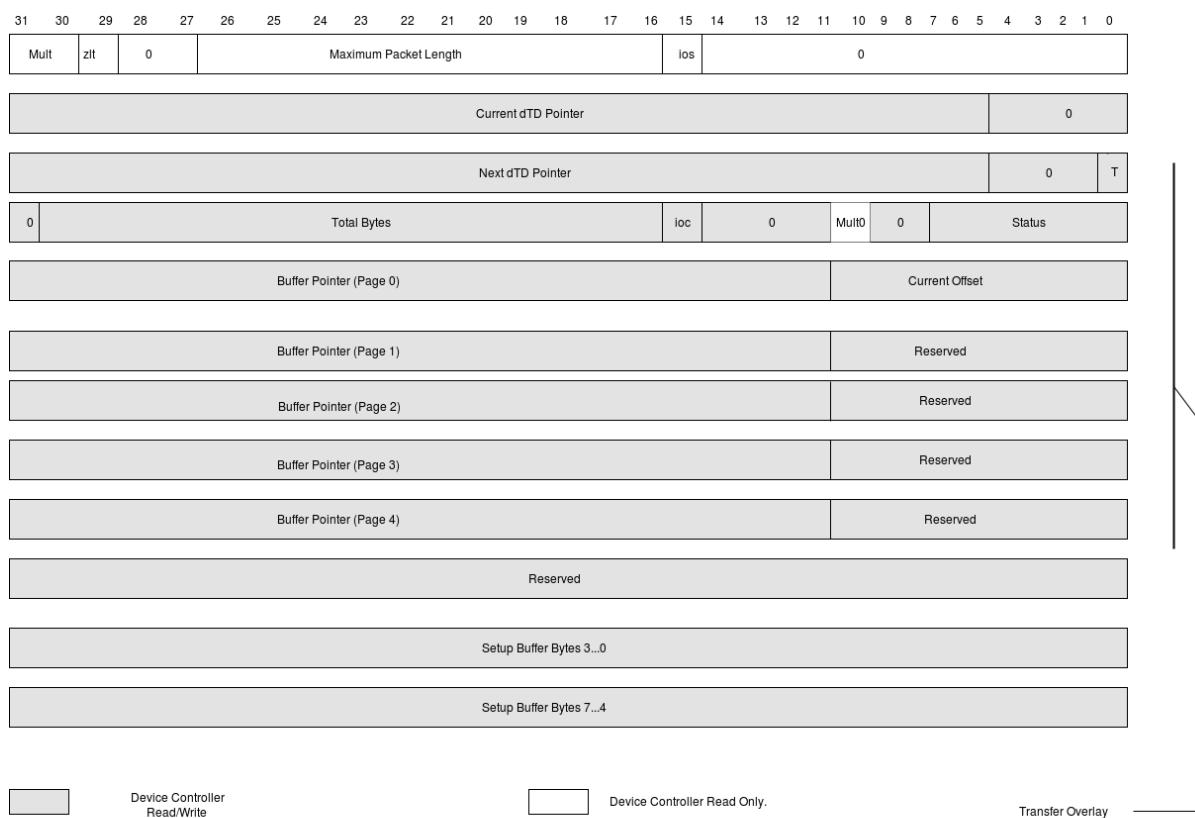
End Point Queue Head Organization

Device queue heads are arranged in an array in a continuous area of memory pointed to by the ENDPOINTLISTADDR pointer. The even numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device

controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers are managed. The dQH is a 48byte data structure, but must be aligned on 64byte boundaries. During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.



Endpoint Queue Head (dQH)

Endpoint Queue Head (dQH)

Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

| Bit | Description |
|-------|--|
| 31:30 | <p>Mult.</p> <p>This field is used to indicate the number of packets executed per transaction description as given by the following:</p> <ul style="list-style-type: none"> 00 – Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dT) 01 – Execute 1 Transaction. 10 – Execute 2 Transactions. 11 – Execute 3 Transactions. |
| 29 | <p>Zero Length Termination Select.</p> <p>This bit is used to indicate when a zero length packet is used to terminate transfers where total transfer length is a multiple . This bit is not relevant for Isochronous</p> <ul style="list-style-type: none"> 0 – Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 – Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length. |
| 28:27 | Reserved. These bits reserved for future use and should be set to zero. |
| 26:16 | <p>Maximum Packet Length.</p> <p>This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).</p> |
| 15 | <p>Interrupt On Setup (IOS).</p> <p>This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.</p> |
| 14:0 | Reserved. These bits reserved for future use and should be set to zero. |

Table 1 Endpoint Capabilities/Characteristics

Transfer Overlay

The seven DWords in the overlay area represent a transaction working space for the device controller. The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint. After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue. See dTD for a description of the overlay fields.

Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

| Bit | Description |
|------|--|
| 31:5 | <p>Current dTD.</p> <p>This field is a pointer to the dTD that is represented in the transfer overlay area.</p> <p>This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.</p> |
| 4:0 | Reserved. These bit reserved for future use and should be set to zero. |

Table 2 Next dTD Pointer

Setup Buffer

The setup buffer is dedicated storage for the 8byte data that follows a setup PID.

| DWord | Bit | Description |
|-------|------|---|
| 1 | 31:0 | <p>Setup Buffer 0.</p> <p>This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.</p> |
| 2 | 31:0 | <p>Setup Buffer 1.</p> <p>This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.</p> |

Table 3 Setup buffer bytes

Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/received for given transfer. The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------------------|-------------|----|----|----|----|-----|----|-------|----|--------|----|----|----|----|----------------|--------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|--|--|--|--|
| 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 | | | | |
| Next Link Pointer | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | T | | | | | | | | |
| 0 | Total Bytes | | | | | ioc | 0 | MultO | 0 | Status | | | | | | | | | | | | | | | | | | | | | | | | | |
| Buffer Pointer (Page 0) | | | | | | | | | | | | | | | Current Offset | | | | | | | | | | | | | | | | | | | | |
| Buffer Pointer (Page 1) | | | | | | | | | | | | | | | 0 | Frame Number | | | | | | | | | | | | | | | | | | | |
| Buffer Pointer (Page 2) | | | | | | | | | | | | | | | Reserved | | | | | | | | | | | | | | | | | | | | |
| Buffer Pointer (Page 3) | | | | | | | | | | | | | | | Reserved | | | | | | | | | | | | | | | | | | | | |
| Buffer Pointer (Page 4) | | | | | | | | | | | | | | | Reserved | | | | | | | | | | | | | | | | | | | | |

 Device Controller Read/Write

 Device Controller Read Only.

Endpoint Transfer Descriptor (dTD)

Each bit of this dTD is explained in below Tables.

| Bit | Description |
|------|---|
| 31:5 | Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively. |
| 4:1 | Reserved. Bits reserved for future use and should be set to zero. |
| 0 | Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue. |

478 . Next dTD Pointer

| Bit | Description |
|-------|---|
| 31 | Reserved. Bits reserved for future use and should be set to zero. |
| 30:16 | Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. |
| 15 | Interrupt On Complete (IOC). This bit is used to indicate if USB INT is to be set in response to device controller being finished with this dTD. |
| 14:12 | Reserved. Bits reserved for future use and should be set to zero. |
| 11:10 | Multiplier Override (MultO). This field can be used for transmit ISO's (i.e. ISOIN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit ISO. |
| 9:8 | Reserved. Bits reserved for future use and should be set to zero. |

| Bit | Description |
|-----|--|
| 7:0 | <p>Status.</p> <p>This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are:</p> <p>Bit Status Field Description</p> <ul style="list-style-type: none"> 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4,2,0 Reserved. |

479 .dTD Token

| Bit | Description |
|--------|---|
| 31:12 | <p>Buffer Pointer.</p> <p>Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.</p> |
| 0;11:0 | <p>Current Offset.</p> <p>Offset into the 4kb buffer where the packet is to begin.</p> |
| 1;11:0 | <p>Frame Number.</p> <p>Written by the device controller to indicate the frame number in which a packet finishes.</p> <p>This is typically be used to correlate relative completion times of packets on an ISO endpoint.</p> |

480 .dTD Buffer Page Pointer List

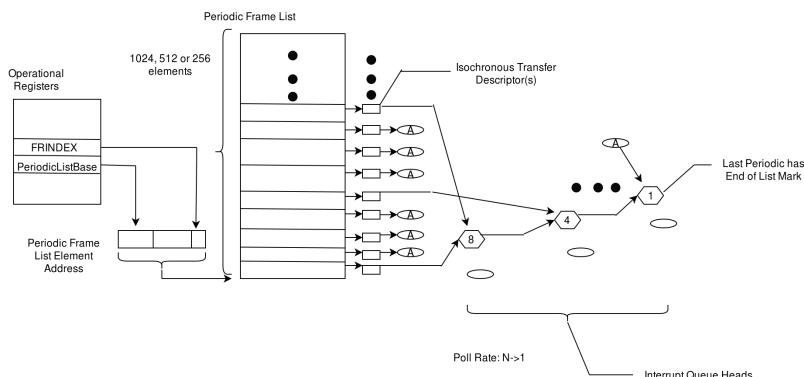
Host Data Structures

Host data structures are used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware). The data structure definitions in this chapter support a 32bit memory buffer address space. The interface consists of a Periodic Schedule, Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors. The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split transaction data streams are managed with Split transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Periodic Frame List

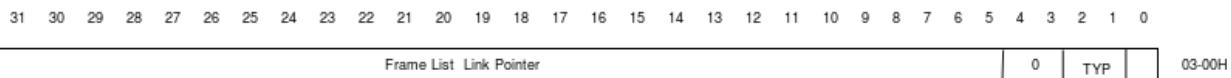
This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the PERIODICLISTBASE address register and the FRINDEX register. The periodic schedule is based on an array of pointers called the Periodic Frame List. The PERIODICLISTBASE address register is

combined with the FRINDEX register to produce a memory pointer into the frame list. The Periodic Frame List implements a sliding window of work over time.



Periodic Schedule Organization

The periodic frame list is a 4Kpage aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must support all three sizes. Programming the size (i.e. the number of elements) is accomplished by system software writing the appropriate value into Frame List Size field in the USBCMD register. Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro frame. The link pointers are aligned on DWord boundaries within the Frame List.



Format of Frame List Element Pointer

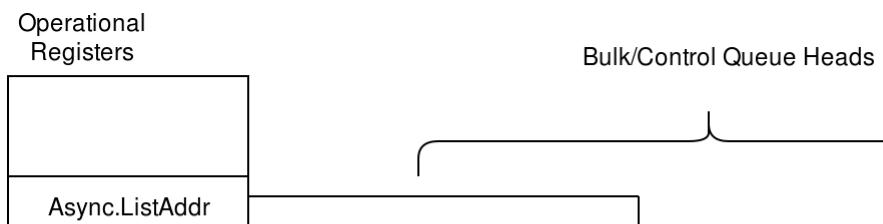
Frame List Link pointers always reference memory objects that are 32byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high, full and low speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing. The least significant bit is the TBit (bit 0). When this bit is set to a one, the host controller will never use the value of the frame list pointer as a physical memory pointer. The Typ field is used to indicate the exact type of data structure being referenced by this pointer. The value encodings are:

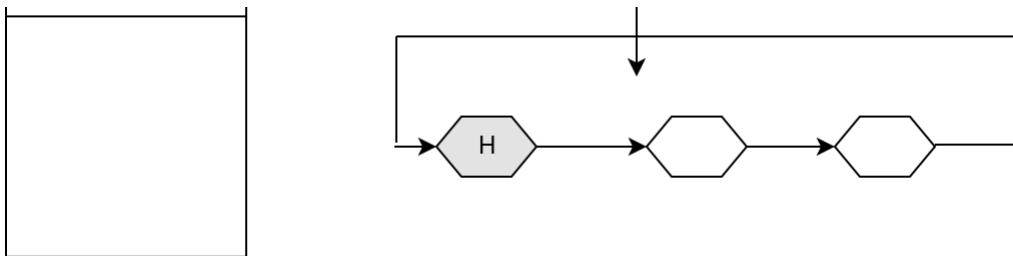
| Value | Meaning |
|-------|--|
| 00 | Isochronous Transfer Descriptor |
| 01 | Queue Head |
| 10 | Split Transaction Isochronous Transfer Descriptor. |
| 11 | Frame Span Traversal Node. |

481 . Typ Field Value Definitions

Asynchronous List Queue Head Pointer

The Asynchronous Transfer List (based at the ASYNCLISTADDR register) is where all the control and bulk transfers are managed. Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.





Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The ASYNCLISTADDR register is simply pointer to the next queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is illustrated in Figure 47. This structure is used only for high speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32byte boundary.

| 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 | Type | T | 03-00H |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|--------|---|--------|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 07-04H | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0B-08H | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0F-0CH | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 13-10H | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 17-14H | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1B-18H | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1F-1CH | | |
| Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 23-20H | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 27-24H | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2B-28H | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2F-2CH | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 33-30H | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 37-34H | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 3B-38H | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 3F-3CH | | |



Host Controller Read/Write



Host Controller Read Only. N

*Note: these fields may be modified by the host controller if the I/O field indicates an OUT.

Isochronous Transaction Descriptor (iTD)

Next Link Pointer

The first DWord of an iTD is a pointer to the next schedule data structure.

| Bit | Description | | | | | | | | | | |
|-------|--|-------|---------|-----|---------------------------------------|-----|-----------------|-----|--|-----|----------------------------------|
| 31:5 | Link Pointer (LP). These bits correspond to memory address signals [31:5], respectively. This field points to another Isochronous Transaction Descriptor (iTД/siTД) or Queue Head (QH). | | | | | | | | | | |
| 4:3 | Reserved. These bits are reserved and their value has no effect on operation. Software should initialize this field to zero. | | | | | | | | | | |
| 2:1 | QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTД, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: <table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>iTD (isochronous transfer descriptor)</td> </tr> <tr> <td>01b</td> <td>QH (queue head)</td> </tr> <tr> <td>10b</td> <td>siTD (split transaction isochronous transfer descriptor)</td> </tr> <tr> <td>11b</td> <td>FSTN (frame span traversal node)</td> </tr> </tbody> </table> | Value | Meaning | 00b | iTD (isochronous transfer descriptor) | 01b | QH (queue head) | 10b | siTD (split transaction isochronous transfer descriptor) | 11b | FSTN (frame span traversal node) |
| Value | Meaning | | | | | | | | | | |
| 00b | iTD (isochronous transfer descriptor) | | | | | | | | | | |
| 01b | QH (queue head) | | | | | | | | | | |
| 10b | siTD (split transaction isochronous transfer descriptor) | | | | | | | | | | |
| 11b | FSTN (frame span traversal node) | | | | | | | | | | |
| 0 | Terminate (T). 1= Link Pointer field is not valid. 0= Link Pointer field is valid. | | | | | | | | | | |

482 . Next Schedule Element Pointer

iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status. Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and Transaction X Offset fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB.

| Bit | Description | | | | | | |
|-------|---|-----|------------|----|--|----|--|
| 31:28 | <p>Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:</p> <table> <thead> <tr> <th>Bit</th><th>Definition</th></tr> </thead> <tbody> <tr> <td>31</td><td>Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.</td></tr> <tr> <td>30</td><td>Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.</td></tr> </tbody> </table> | Bit | Definition | 31 | Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule. | 30 | Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary. |
| Bit | Definition | | | | | | |
| 31 | Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule. | | | | | | |
| 30 | Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary. | | | | | | |
| 27:16 | Transaction X Length. For an OUT, this field is the number of data bytes the host controller will send during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (e.g. 0zero length data, 1one byte, 2two bytes,etc.). The maximum value this field may contain is 0xC00 (3072). | | | | | | |
| 15 | Interrupt On Complete (IOC). If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold | | | | | | |
| 14:12 | Page Select (PG). These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6. | | | | | | |
| 11:0 | Transaction X Offset. This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent PG field to produce the starting buffer address for this transaction. | | | | | | |

483 . iTD Transaction Status and Control

iTD Buffer Page Pointer List (Plus)

DWords 915 of an isochronous transaction descriptor are nominally page pointers (4K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non contiguous. Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page. Since each pointer is a 4K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

| Bit | Description |
|-------|---|
| 31:12 | Buffer Pointer (Page 0). This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12]. |
| 11:8 | Endpoint Number (Endpt). This 4bit field selects the particular endpoint number on the device serving as the data source or sink. |
| 7 | Reserved |

| Bit | Description |
|-----|--|
| 6:0 | Device Address. This field selects the specific device serving as the data source or sink. |

484 .iTD Buffer Pointer Page 0 (Plus)

| Bit | Description |
|-------|---|
| 31:12 | Buffer Pointer (Page 1). This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12]. |
| 11 | Direction (I/O). 0 = OUT; 1 = IN. This field encodes whether the high speed transaction should use an IN or OUT PID. |
| 10:0 | Maximum Packet Size. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). This field is used for high bandwidth endpoints where more than one transaction is issued per transaction description (e.g. per micro frame). This field is used with the Multi field to support high bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results. |

485 .iTD Buffer Pointer Page 1 (Plus)

| Bit | Description |
|-------|---|
| 31:12 | Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12]. |
| 11:2 | Reserved |
| 1:0 | Multi. This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (e.g. per micro-frame). The valid values are: |
| | Value Meaning |
| 00b | Reserved |
| 01b | One transaction to be issued for this endpoint per micro frame |
| 10b | Two transactions to be issued for this endpoint per micro frame |
| 11b | Three transactions to be issued for this endpoint per micro frame |

486 .iTD Buffer Pointer Page 2 (Plus)

| Bit | Description |
|-------|--|
| 31:12 | Buffer Pointer. This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12]. |
| 11:0 | Reserved |

487 .iTD Buffer Pointer Page 3-6 (Plus)

Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------------|----|-------------|----|----|----|----------|----|----|----|----|----|----|----|-------|----|----|----|----------------|----|----|----|---|---|---|---|---|---|---|---|---|---|--|--|
| 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 | | |
| Next Link Pointer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| IO | | Port Number | | R | | Hub Addr | | | R | | | R | | EndPt | | R | | Device Address | | | | | | | | | | | | | | | |

| | | | | |
|-------------------------|--------------------|--------------------|---|--------|
| Reserved | μ Frame C-mask | μ Frame S-mask | 0B-08H | |
| loc | P | Reserved | Total Bytes to Transfer μ Frame C-prog-mask Status | |
| Buffer Pointer (Page 0) | | Current Offset | | 13-10H |
| Buffer Pointer (Page 1) | | Reserved | TP Tcount | 17-14H |
| Back Pointer | | 0 | T | 1B-18H |

 Host Controller Read/Write  Host Controller Read Only.

Split transaction Isochronous Transaction Descriptor (siTD)

Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure.

| Bit | Description | | | | | | | | | | |
|-------|--|-------|---------|-----|---------------------------------------|-----|-----------------|-----|--|-----|----------------------------------|
| 31:5 | Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively. | | | | | | | | | | |
| 4:3 | Reserved. These bits must be written as zeros. | | | | | | | | | | |
| 2:1 | QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>iTD (isochronous transfer descriptor)</td> </tr> <tr> <td>01b</td> <td>QH (queue head)</td> </tr> <tr> <td>10b</td> <td>siTD (split transaction isochronous transfer descriptor)</td> </tr> <tr> <td>11b</td> <td>FSTN (frame span traversal node)</td> </tr> </tbody> </table> | Value | Meaning | 00b | iTD (isochronous transfer descriptor) | 01b | QH (queue head) | 10b | siTD (split transaction isochronous transfer descriptor) | 11b | FSTN (frame span traversal node) |
| Value | Meaning | | | | | | | | | | |
| 00b | iTD (isochronous transfer descriptor) | | | | | | | | | | |
| 01b | QH (queue head) | | | | | | | | | | |
| 10b | siTD (split transaction isochronous transfer descriptor) | | | | | | | | | | |
| 11b | FSTN (frame span traversal node) | | | | | | | | | | |
| 0 | Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid. | | | | | | | | | | |

488 . Next Link Pointer

siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

| Bit | Description |
|-------|---|
| 31 | Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the fullspeed transaction should be an IN or OUT |
| 30:24 | Port Number. This field is the port number of the recipient Transaction Translator. |

| Bit | Description |
|-------|--|
| 23 | Reserved. Bit reserved and should be set to zero. |
| 22:16 | Hub Address. This field holds the device address of the Companion Controllers' hub. |
| 15:12 | Reserved. Bit reserved and should be set to zero. |
| 11:8 | Endpoint Number (Endpt). This 4bit field selects the particular endpoint number on the device serving as the data source or sink. |
| 7 | Reserved. Bit reserved and should be set to zero. |
| 6:0 | Device Address. This field selects the specific device serving as the data source or sink. |

489 . Endpoint and Transaction Translator Characteristics

| Bit | Description |
|-------|---|
| 31:16 | Reserved. This field reserved for future use. It should be set to zero. |
| 15:8 | <p>Split Completion Mask (μFrame CMask). This field (along with the Active and SplitX state fields in the Status byte) is used to determine during which microframes the host controller should execute completesplit transactions. This field is a straight bit position field, so if bit zero is set then the completesplit transaction should occur in the first microframe, if bit 1 is set then it should occur in the second microframe, and so on.</p> <p>When the criteria for using this field is met, an all zeros value has undefined behavior. The host controller uses the value of the three loworder bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the μFrame CMask field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.</p> <p>The CMask can be set for multiple micro frames, as it is not known in which micro-frame the transaction will complete. So the CMask can be set for the micro frame after the SMask and all subsequent micro frames thereafter. The CMask field should not have a bit set to the same microframe as the SMask is set to.</p> |
| 7:0 | Split Start Mask (μ Frame Smask). This field (along with the Active and SplitXstate fields in the Status byte) is used to determine during which microframes the host controller should execute startsplit transactions. The host controller uses the value of the three loworder bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the μ Frame Smask field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results. This field should have only one bit set to 1 at any given time. Having more than one bit set will result in undefined results |

490 . Micro frame Schedule Control

siTD Transfer State

DWords 36 are used to manage the state of the transfer.

| Bit | Description |
|-----|--|
| 31 | Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it will assert a hardware interrupt at the next interrupt threshold. |

| Bit | Description |
|-------|---|
| 30 | Page Select (P). Used to indicate which data page pointer should be concatenated with the CurrentOffset field to construct a data buffer pointer (0 selects Page 0 pointer and 1 selects Page 1). The host controller is not required to write this field back when the siTD is retired (Active bit transitioned from a one to a zero). |
| 29:26 | Reserved. This field reserved for future use and should be set to zero |
| 25:16 | Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh) |
| 15:8 | μFrame Completesplit Progress Mask (CprogMask). This field is used by the host controller to record which splitcompletes has been executed |
| 7:0 | Status. This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding: |

491 . siTD Transfer Status and Control

siTD Buffer Pointer List (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4K (page) aligned buffer pointers. The least significant 12 bits of each DWord are used as additional transfer state.

| Bit | Description |
|-------|--|
| 31:12 | Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4K paged aligned, physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field <i>P</i> specifies the <i>current active pointer</i> |

| Bit | Description | | | | | | | | | | |
|-------|--|-------|---------|-----|--|-----|---|-----|--|-----|---|
| 11:0 | <p>Page 0:</p> <p>Current Offset. The 12 least significant bits of the Page 0 pointer is the current byte offset for the current page pointer (as selected with the page indicator bit (<i>P</i>field)). The host controller is not required to write this field back when the siTD is retired (<i>Active</i> bit transitioned from a one to a zero). The least significant bits of Page 1 pointer is split into three subfields</p> <p>Page 1:</p> <p>11:5 Reserved.</p> <p>4:3 Transaction position (TP). This field is used with Tcount to determine whether to send all, first, middle, or last with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are:</p> <table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>All. The entire full-speed transaction data payload is in this transaction (i.e. less than or equal to 188 bytes).</td> </tr> <tr> <td>01b</td> <td>Begin. This is the first data payload for a full-speed that is greater than 188 bytes.transaction</td> </tr> <tr> <td>10B</td> <td>Mid. This is the <i>middle</i> payload for a full-speed OUT transaction that is larger than 188 bytes.</td> </tr> <tr> <td>11b</td> <td>End. This is the <i>last</i> payload for a full-speed OUT transaction that was larger than 188 bytes.</td> </tr> </tbody> </table> <p>2:0 Transaction count (TCount). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.</p> | Value | Meaning | 00b | All. The entire full-speed transaction data payload is in this transaction (i.e. less than or equal to 188 bytes). | 01b | Begin. This is the first data payload for a full-speed that is greater than 188 bytes.transaction | 10B | Mid. This is the <i>middle</i> payload for a full-speed OUT transaction that is larger than 188 bytes. | 11b | End. This is the <i>last</i> payload for a full-speed OUT transaction that was larger than 188 bytes. |
| Value | Meaning | | | | | | | | | | |
| 00b | All. The entire full-speed transaction data payload is in this transaction (i.e. less than or equal to 188 bytes). | | | | | | | | | | |
| 01b | Begin. This is the first data payload for a full-speed that is greater than 188 bytes.transaction | | | | | | | | | | |
| 10B | Mid. This is the <i>middle</i> payload for a full-speed OUT transaction that is larger than 188 bytes. | | | | | | | | | | |
| 11b | End. This is the <i>last</i> payload for a full-speed OUT transaction that was larger than 188 bytes. | | | | | | | | | | |

492 .Buffer page pointer list (plus)

siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

| Bit | Description |
|------|--|
| 31:5 | siTD Back Pointer. This field is a physical memory pointer to a siTD |
| 4:1 | Reserved. This field is reserved for future use. It should be set to zero. |

| Bit | Description |
|-----|---|
| 0 | Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid. |

493 .siTD Back Link Pointer

Queue Element Transfer Descriptor (qTD)

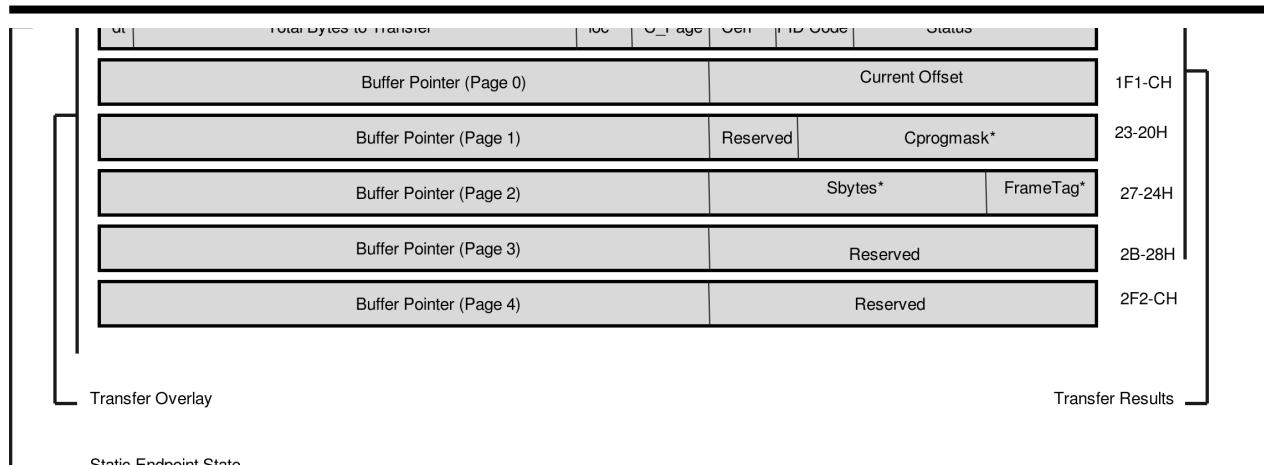
This data structure is only used with a queue head. This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20480 (5*4096) bytes. The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five element array of data buffer pointers. This structure is 32 bytes (or one 32byte cache line). This data structure must be physically contiguous. The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous. Host controller updates (host controller writes) to standalone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------------|-------------------------|----|----|----------------------------|----|----|----|----|----|----|-----|--------|------|----------|----------------|----|----|----|----|----|----|---|---|---|--------|---|---|---|--------|---|---|--------|
| 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 | |
| Next qTD Pointer | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | T | 03-00H | | | |
| Alternate Next qTD Pointer | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | T | 07-04H | | | |
| dt | Total Bytes to Transfer | | | | | | | | | | loc | C_Page | Cerr | PID Code | Status | | | | | | | | | | 0B-08H | | | | | | | |
| Buffer Pointer (Page 0) | | | | | | | | | | | | | | | Current Offset | | | | | | | | | | | | | | | | | 0F-0CH |
| Buffer Pointer (Page 0) | | | | | | | | | | | | | | | Reserved | | | | | | | | | | | | | | | | | 13-10H |
| Buffer Pointer (Page 0) | | | | | | | | | | | | | | | Reserved | | | | | | | | | | | | | | | | | 17-14H |
| Buffer Pointer (Page 0) | | | | | | | | | | | | | | | Reserved | | | | | | | | | | | | | | | | | 1B-18H |
| Buffer Pointer (Page 0) | | | | | | | | | | | | | | | Reserved | | | | | | | | | | | | | | | | | 1F-1CH |
| Host Controller Read/Write | | | | Host Controller Read Only. | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Queue Element Transfer Descriptor Block Diagram

Queue Head

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|-------------------------|----|----|----|----|----|----|----|----|----|-----|--------|------|----------|--------|----|----|----|----|----|----|---|---|---|--------|---|---|---|---|---|-----|---|--------|--|--------|--------|
| 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 | | | | | |
| Queue Head Horizontal Link Pointer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | Typ | T | 03-00H | | | |
| RL C Maximum Packet Length H dtc EP EndPt I Device Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 07-04H | |
| Mult Port Number* Hub Addr* μFrame Cmask* μFrame Smask | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0B-08H | |
| Current qTD Pointer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0F-0CH |
| Next qTD Pointer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 13-10H |
| Alternate Next qTD pointer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 17-14H |
| dt | Total Bytes to Transfer | | | | | | | | | | loc | C_Page | Cerr | PID Code | Status | | | | | | | | | | 1B-18H | | | | | | | | | | | |



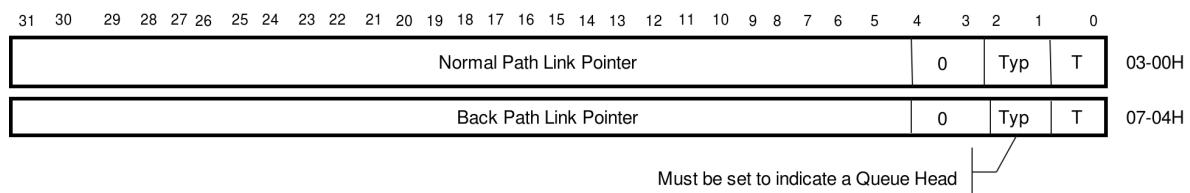
*These fields are used exclusively to support split transactions to USB 2.0 Hubs

 Host Controller Read/Write Host Controller Read Only.

Queue Head Structure Layout

Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full and Low speed transactions that span a Host frame boundary. See section Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use will yield undefined results.



Frame Span Traversal Node Structure Layout

FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

| Bit | Description |
|------|---|
| 31:5 | Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively. |
| 4:3 | Reserved. These bits must be written as 0s. |

| Bit | Description | | | | | | | | | | |
|-------|--|-------|---------|-----|---------------------------------------|-----|-----------------|-----|--|-----|----------------------------------|
| 2:1 | <p>QH/(s)iTDFSTN Select (Typ).</p> <p>This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched.</p> <p>Value encodings are:</p> <table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>iTD (isochronous transfer descriptor)</td> </tr> <tr> <td>01b</td> <td>QH (queue head)</td> </tr> <tr> <td>10b</td> <td>siTD (split transaction isochronous transfer descriptor)</td> </tr> <tr> <td>11b</td> <td>FSTN (Frame Span Traversal Node)</td> </tr> </tbody> </table> | Value | Meaning | 00b | iTD (isochronous transfer descriptor) | 01b | QH (queue head) | 10b | siTD (split transaction isochronous transfer descriptor) | 11b | FSTN (Frame Span Traversal Node) |
| Value | Meaning | | | | | | | | | | |
| 00b | iTD (isochronous transfer descriptor) | | | | | | | | | | |
| 01b | QH (queue head) | | | | | | | | | | |
| 10b | siTD (split transaction isochronous transfer descriptor) | | | | | | | | | | |
| 11b | FSTN (Frame Span Traversal Node) | | | | | | | | | | |
| 0 | Terminate (T). 1=Link Pointer field is not valid. 0=Link Pointer is valid. | | | | | | | | | | |

FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head. If the Tbit in this pointer is a zero, then this FSTN is a Save-Place indicator. Its Typ field must be set by software to indicate the target data structure is a queue head. If the Tbit in this pointer is set to a one, then this FSTN is the Restore indicator. When the Tbit is a one, the host controller ignores the Typ field.

| Bit | Description |
|------|---|
| 31:5 | <p>Back Path Link Pointer (BPLP).</p> <p>This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.</p> |
| 4:3 | Reserved. These bits must be written as 0s. |
| 2:1 | <p>Typ.</p> <p>Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.</p> |
| 0 | <p>Terminate (T).</p> <p>1=Link Pointer field is not valid (i.e. the host controller must not use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Restore indicator.</p> <p>0=Link Pointer is valid (i.e. the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator</p> |

DEVICE OPERATIONAL MODEL

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point of view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a ‘1’. In the disabled state, the pull up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

1. Set Controller Mode in the USBMODE register to device mode.

NOTE: Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

2. Allocate and Initialize device queue heads in system memory.

Minimum: Initialize device queue heads 0 Tx & 0 Rx.

NOTE: All device queue heads associated with control endpoints must be initialized before the control endpoint is enabled. NonControl device queue heads must be initialized before the endpoint is used and not necessarily before the

endpoint is enabled.

For information on device queue heads, refer to section Device Data Structures.

3. Configure ENDPOINTLISTADDR Pointer.

For additional information on ENDPOINTLISTADDR, refer to the register table.

4. Enable the microprocessor interrupt associated with the USBHS core.

Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect,USB Reset Received, DCSuspend.

For a list of available interrupts refer to the USBINTR and the USBSTS register tables.

5. Set Run/Stop bit to Run Mode.

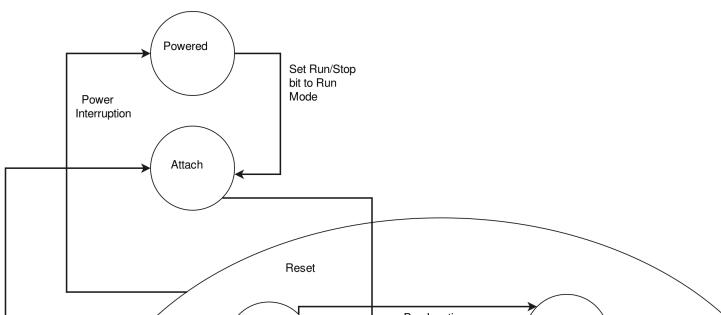
After the Run bit is set, a device reset will occur. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.

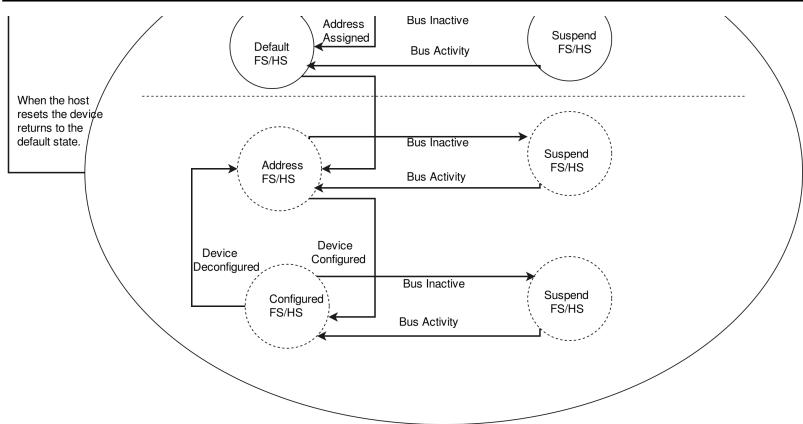
NOTE: Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

Port State and Control

From a chip or system reset, the device controller enters the powered state. A transition from the powered state to the attach state occurs when the Run/Stop bit is set to a ‘1’. After receiving a reset on the bus, the port will enter the defaultFS or defaultHS state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0. The following state diagram depicts the state of a USB 2.0 device.





Device State Diagram

States powered, attach, defaultFS/HS, suspendFS/HS are implemented in the device controller and are communicated to the DCD using the following status bits:

| Bit | Register |
|--------------------|----------|
| DCSuspend | USBSTS |
| USB Reset Received | USBSTS |
| Port Change Detect | USBSTS |
| HighSpeed Port | PORTSC |

494 . Device Controller State Information Bits

It is the responsibility of the DCD to maintain a state variable to differentiate between the DefaultFS/HS state and the Address/Configured states. Change of state from Default to Address and the Configured states is part of the enumeration process described in the device framework section of the USB 2.0 Specification. As a result of entering the Address state, the device address register (DEVICEADDR) must be programmed by the DCD. Entry into the Configured indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRLx registers and initializing the associated queue heads.

Bus Reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received: Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register. Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register. Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFFFFFF to ENDPTFLUSH. Read the reset bit in the PORTSCx register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset.



Note

A hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely reinitialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated. At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port

Change Detect is indicated. After a Port Change Detect, the device has reached the default state and the DCD can read the PORTSCx to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration.

Suspend/Resume

Suspend

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a nondefault address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming DC Suspend Interrupt is enabled). When the DCSuspend bit in the PORTSCx is set to a '1', the device controller is suspended. DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

Resume

If the device controller is suspended, its operation is resumed when any nonidle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the PORTSCx while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition

15.6.4 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction. The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a control type data channel used for device discovery and enumeration. Other types of endpoints supported by USB include bulk, interrupt, and isochronous. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification. The USBHS device controller hardware supports up to the USB 2.0 maximum of 32 endpoint specified numbers. Each additional endpoint beyond the required endpoint position adds additional hardware logic. The maximum number of endpoint numbers available to the DCD is configured at hardware synthesis timer. After synthesis, the DCD can enable, disable and configure endpoint type up to the maximum selected during synthesis. Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1IN to be a bulk endpoint and endpoint 1OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is,

for example, is always a control endpoint and uses the pair of directions. Each endpoint direction requires a queue head allocated in memory. If the maximum of 16 endpoint numbers, one for each endpoint direction are being used by the device controller, then 32 queue heads are required.

Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRLx register. Each 32bit ENDPTCTRLx is split into an upper and lower half. The lower half of ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization.

| Field | Value |
|---------------------|---|
| Data Toggle Reset | '1' |
| Data Toggle Inhibit | '0' |
| Endpoint Type | "00" – Control "01" – Isochronous "10" – Bulk "11" – Interrupt |
| Endpoint Stall | '0' |

495 . Device Controller Endpoint Initialization

Stalling

There are two occasions where the device controller may need to return to the host a STALL. The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework (chapter 9). A functional stall is only used on noncontrol endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD. A protocol stall, unlike a function stall, is used on control endpoints and is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

| USB Packet | Endpoint Stall Bit. | Effect on STALL bit. | USB Response |
|---|---------------------|----------------------|--------------|
| SETUP packet received by a noncontrol endpoint. | N/A | None. | STALL |
| IN/OUT/PING packet received by a noncontrol endpoint. | '1' | None. | STALL |

| USB Packet | Endpoint Stall Bit. | Effect on STALL bit. | USB Response |
|--|---------------------|----------------------|----------------------|
| IN/OUT/PING packet received by a noncontrol endpoint | '0' | None. | ACK/ NAK/ NYET |
| SETUP packet received by a control endpoint. | N/A | Cleared | ACK |
| IN/OUT/PING packet received by a control endpoint | '1' | None. | STALL |
| IN/OUT/PING packet received by a control endpoint. '0' | | None | ACK/ NAK/ NYET |

496 . Device Controller Stall Response Matrix

Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, refer to the USB 2.0 specification.

Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the ENDPTCTRLx register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

Data Toggle Inhibit

This feature is for test purposes only and should never be used during normal device controller operation. Setting the data toggle Inhibit bit active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state. In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from resending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

15.6.5 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were designed so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High speed turnaround time requirements by simply increasing clock rate. A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk

pipe, then we can expect the host will send IN requests to that endpoint. This device controller is designed in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as “priming” the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be designed properly use priming. Further, note that the term “flushing” is used to describe the action of clearing a packet that was queued for execution.

Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received. After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time. After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB. Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section Bandwidth and Latency Issues.

Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host. Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence. A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

$$N = \text{INT}(\text{Number Of Bytes}/\text{Max. Packet Length}) + 1$$

With Zero Length Termination (ZLT) = 1

$$N = \text{MAXINT}(\text{Number Of Bytes}/\text{Max. Packet Length})$$

| Bytes (dTD) | Max. Packet Length (dQH) | N | P1 | P2 | P3 |
|-------------|--------------------------|---|-----|-----|----|
| 511 | 256 | 2 | 256 | 255 | |
| 512 | 256 | 3 | 256 | 256 | 0 |
| 512 | 512 | 2 | 512 | 0 | |

497 . Variable Length Transfer Protocol Example (ZLT = 0)

| Bytes (dTD) | Max. Packet Length (dQH) | N | P1 | P2 | P3 |
|-------------|--------------------------|---|-----|-----|----|
| 511 | 256 | 2 | 256 | 255 | |
| 512 | 256 | 2 | 256 | 256 | |
| 512 | 512 | 1 | 512 | | |

498 . Variable Length Transfer Protocol Example (ZLT = 1)

The ZLT bit in the dTD will operate as following on BULK and control transfers:

.ZLT = 0, the default value, means that the zero length termination is active. With the ZLT option enabled, when the device is transmitting, the hardware will automatically append a zero packet length when the following conditions are true:

The packet transmitted equals maximum packet length.

The dTD has exhausted the field Total Bytes

After this the dTD will be retired. When the device is receiving, if the last packet length received equal maximum packet length and the total bytes is zero, it will wait for a zero length packet from the host to retire the current dTD. .ZLT = 1, means the zero length termination is inactive. With the ZLT option disabled, when the device is transmitting, the hardware will not append any zero length packet. When receiving, it will not require a zero length packet to retire a dTD whose last packet was equal to the maximum packet length packet. The dTD is retired as soon as Total Bytes field goes to zero, or a short packet is received. Each transfer is defined by one dTD, so the zero length termination is for each dTD. In some software application cases, the logic transfer does not fit into just one dTD, so it does not make sense to add a Zero Length Termination packet each time a dTD is consumed. On those cases we recommend to turn off this ZLT feature, and use software to generate the zero length termination.

TXdT is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RXdT is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.

• A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.

• A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the

Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active. On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction. On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to reprime the endpoint.

Interrupt/Bulk Endpoint Bus Response Matrix

| | Stall | Not Primed | Primed | | Underflow | Overflow | Not Enabled |
|-------|--------|------------|-----------------------|----------|-----------|----------|-------------|
| Setup | Ignore | Ignore | Ignore | N/A | N/A | BTO | |
| In | STALL | NAK | Transmit | BS Error | N/A | BTO | |
| Out | STALL | NAK | Receive + NYET/ACKN/A | N/A | NAK | BTO | |
| Ping | STALL | NAK | ACK | N/A | N/A | BTO | |

| | Stall | Not Primed | Primed | Underflow | Overflow | Not Enabled |
|---------|--------------|-------------------|---------------|------------------|-----------------|--------------------|
| Invalid | Ignore | Ignore | Ignore | Ignore | Ignore | BTO |

499 . Interrupt/Bulk Endpoint Bus Response Matrix

BS Error = Force Bit Stuff Error

NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

Control Endpoint Operation Model

Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged. The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet. In hardware versions 2.3 and later, the setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted by an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Setup Packet Handling (Pre2.3 hardware)

After receiving an interrupt and inspecting USBMODE to determine that a setup packet was received on a particular pipe:

1. Duplicate contents of dQH.SsetupBuffer into local software byte array.
2. Write '1' to clear corresponding ENDPTSETUPSTAT bit and thereby disabling Setup Lockout. (i.e. the Setup Lockout activates as soon as a setup arrives. By writing to the ENDPTSETUPSTAT, the device controller will accept new setup packets.)
3. Process setup packet using local software byte array copy and execute status/handshake phases.
4. Before priming for status/handshake phases ensure that ENDPTSETUPSTAT is '0'. The time from writing a '1' to ENDPTSETUPSTAT and reading back a '0' may vary according to the type of traffic on the bus up to nearly a 1ms, however it is absolutely necessary to ensure ENDPTSETUPSTAT has transitioned to '0' after step 1) and before priming for the status/handshake phases.

Setup Packet Handling (2.3 hardware and later)

Disable Setup Lockout by writing '1' to Setup Lockout Mode (SLOM) in USBMODE. (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:

1. Write '1' to clear corresponding bit ENDPTSETUPSTAT.
2. Write '1' to Setup Tripwire (SUTW) in USBCMD register.
3. Duplicate contents of dQH.SetupBuffer into local software byte array.
4. Read Setup TripWire (SUTW) in USBCMD register. (if set continue; if cleared goto 2)
5. Write '0' to clear Setup Tripwire (SUTW) in USBCMD register.

6. Process setup packet using local software byte array copy and execute status/handshake phases.

7. Before priming for status/handshake phases ensure that ENDPTSETUPSTAT is ‘0’.

A poll loop should be used to wait until ENDPTSETUPSTAT transitions to ‘0’ after step 1) above and before priming for the status/handshake phases. In core versions 3.2 and later, the time from writing a ‘1’ to ENDPTSETUPSTAT and reading back a ‘0’ is very short (~12 us) so a poll loop in the DCD will not be harmful. In core versions 3.1 and earlier, the time from writing a ‘1’ to ENDPTSETUPSTAT and reading back a ‘0’ may vary according to the type of traffic on the bus up to nearly a 1ms, however it is absolutely necessary to ensure ENDPTSETUPSTAT has transitioned to ‘0’ after step 1) and before priming for the status/handshake phases.

Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer. After priming the packet, the DCD must verify a new setup packet has not been received by reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTATUS register is a one. If a prime fails, i.e. The ENDPTPRIME bit goes to zero and the ENDPTSTATUS bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet. Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (ENDPTSTATUS) to enforce data coherency with the setup packet.

Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

| Token Type | Endpoint State | | | | | | Setup Lockout |
|------------|----------------|------------|--------------------|-----------|----------|-------------|---------------|
| | Stall | Not Primed | Primed | Underflow | Overflow | Not enabled | |
| Setup | ACK | ACK | ACK | N/A | SYSERR | BTO | |
| In | STALL | NAK | Transmit | BS Error | N/A | BTO | NA |
| Out | STALL | NAK | Receive + NYET/ACK | N/A | NAK | BTO | NA |
| Ping | STALL | NAK | ACK | N/A | N/A | BTO | NA |
| Invalid | Ignore | Ignore | Ignore | Ignore | Ignore | BTO | Ignore |

BS Error = Force Bit Stuff Error
NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.
SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive

Control Endpoint Bus Response Matrix

Isochronous Endpoint Operational Model

Isochronous endpoints are used for realtime scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes. Real time delivery by the device controller will be accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note: MULT is a twobit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISOdTD is still active after that frame, then the ISOdTD will be held ready until executed or canceled by the DCD. An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD. The first difference between bulk and ISOendpoints is that priming an ISOendpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame. Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition. The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISOdTD and move to the next ISOdTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISOdTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISOdTDs that pile up from a failure of the host to move the data. Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the Transaction Error bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired

MULT counter reaches zero.

Fulfillment Error [Transaction Error bit is set]

Packets Occurred > 0 AND # Packets Occurred < MULT

- RX Packet Retired:

MULT counter reaches zero.

NonMDATA Data PID is received**

** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.

Overflow Error:

- Packet received is > maximum packet length. [Buffer Error bit is set]

- Packet received exceeds total bytes allocated in dTD. [Buffer Error bit is set]

Fulfillment Error [Transaction Error bit is set]

Packets Occurred > 0 AND # Packets Occurred < MULT

CRC Error [Transaction Error bit is set]

Isochronous Pipe Synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro)frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (micro)frame number [N], the DCD should interrupt on SOF during frame N1. When the FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro)frame N-1 so that the device controller will execute delivery during (micro)frame N.

Isochronous Endpoint Bus Response Matrix

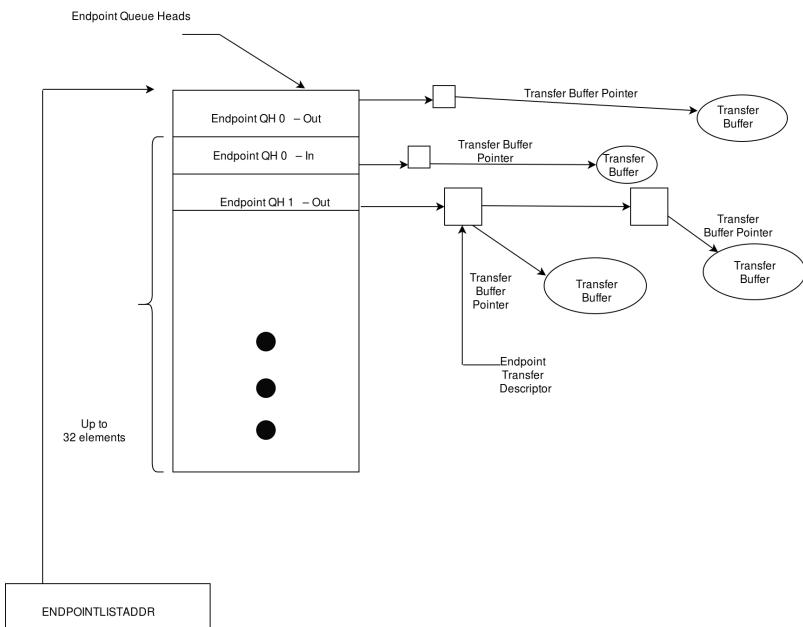
| | Stall | Not Primed | Primed | Underflow | Overflow | Not Enabled |
|---------|-------------|-------------|----------|-----------|-------------|-------------|
| Setup | STALL | STALL | STALL | N/A | N/A | BTO |
| In | NULL Packet | NULL Packet | Transmit | BS Error | N/A | BTO |
| Out | Ignore | Ignore | Receive | N/A | Drop Packet | BTO |
| Ping | Ignore | Ignore | Ignore | Ignore | Ignore | BTO |
| Invalid | Ignore | Ignore | Ignore | Ignore | Ignore | BTO |

500 . Isochronous Endpoint Bus Response Matrix

BS Error = Force Bit Stuff Error

NULL Packet = Zero Length Packet

15.6.6 Managing Queue Heads



End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTd). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 79. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers). In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multiplier, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth and in conjunction with the USB Chapter 9 protocol. Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.
- Write the next dTD Terminate bit field to “1”.
- Write the Active bit in the status field to “0”.
- Write the Halt bit in the status field to “0”.

Operational Model For Setup Transfers

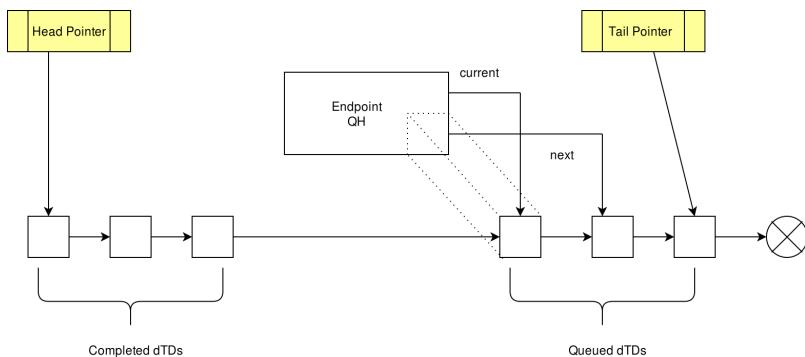
As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8byte buffer within the dQH. Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH RX to software buffer.
2. Acknowledge setup backup by writing a “1” to the corresponding bit in ENDPTSETUPSTAT.
3. Check for pending data or status dTD’s from previous control transfers and flush if any exist as discussed in section Flushing/Depriming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

15.6.7 Managing Transfers with Transfer Descriptors

Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.



Software Link Pointers

Software Link Pointers

Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs. Allocate 8DWord dTD block of memory aligned to 8DWord boundaries. Example: bit address 4:0 would be equal to “00000”

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to “1”.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to “1” and all remaining status bits set to “0”.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

Executing A Transfer Descriptor

To safely add a dTD, the DCD must follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty:

Check DCD driver to see if pipe is empty (internal representation of linkedlist should indicate if any packets are outstanding)

- Case 1: Link list is empty

1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
2. Clear active & halt bit in dQH (in case set from a previous error).
3. Prime endpoint by writing ‘1’ to correct bit position in ENDPTPRIME.

- Case 2: Link list is not empty

1. Add dTD to end of linked list.
2. Read correct prime bit in ENDPTPRIME – if ‘1’ DONE.
3. Set ATDTW bit in USBCMD register to ‘1’.
4. Read correct status bit in ENDPTSTAT. (store in tmp. variable for later)
5. Read ATDTW bit in USBCMD register.

If ‘0’ goto 3.

If ‘1’ continue to 6.

6. Write ATDTW bit in USBCMD register to ‘0’.
7. If status bit read in (4) is ‘1’ DONE.
8. If status bit read in (4) is ‘0’ then Goto Case 1: Step 1.

Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the hostinitiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure. By reading the status fields

of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix. In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

Flushing/Depriming an Endpoint

It is necessary for the DCD to flush to deprime one or more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are '0'.

Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.

3. Read ENDPTSTAT to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:

Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush by repeating

steps 1-3 until each endpoint is successfully flushed.

Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

| Error | Direction | Packet Type | Data Buffer Error | Transaction Error Bit |
|-----------------------|-----------|-------------|-------------------|-----------------------|
| Overflow ** | RX | Any | 1 | 0 |
| ISO Packet Error | RX | ISO | 0 | 1 |
| ISO Fulfillment Error | Both | ISO | 0 | 1 |

501 . Device Error Matrix

15.6.8 Servicing Interrupts

The interrupt service routine must consider that there are highfrequency, lowfrequency operations, and error operations and order accordingly.

HighFrequency Interrupts

High frequency interrupts in particular should be handled in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

| Execution Order | Interrupt | Action |
|-----------------|--------------------------------------|---|
| 1a | USB Interrupt ** ENDPTSETUPSTATUS | Copy contents of setup buffer and acknowledge setup packet (as indicated in section Managing Queue Heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol. |
| 1b | USB Interrupt ** ENDPTCOMPLETE | Handle completion of dTD as indicated in section Managing Queue Heads. |
| 2 | SOF Interrupt | Action as deemed necessary by application. This interrupt may not have a use in all applications. |

502 . High Frequency Interrupt Events

LowFrequency Interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they don't occur often in comparison to the highfrequency interrupts.

| Interrupt | Action |
|------------------------|--|
| Port Change | Change software state information. |
| Sleep Enable (Suspend) | Change software state information. Low power handling as necessary |
| Reset Received | Change software state information. Abort pending transfers. |

503 . Low Frequency Interrupt Events

Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

| Interrupt | Action |
|---------------------|--|
| USB Error Interrupt | This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packetlevel errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE). |
| System Error | Unrecoverable error. Immediate Reset of core; free transfers buffers in |

504 . Error Interrupt Events

15.6.9 HOST OPERATIONAL MODEL

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software). Each significant operational feature of the EHCI host

controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

Host Controller Initialization

When the system boots, the host controller is enumerated, assigned a base address for the register space and BIOS sets the FLADJ register to a systemspecific value. After initial poweron or HCReset (hardware or via HCReset bit in the USBCMD register), all of the operational registers will be at their default values, as illustrated in Table 26. After a hardware reset, only the operational registers not contained in the Auxiliary power well will be at their default values.

| Operational Register | Default Value (afterReset) |
|----------------------|---|
| USBCMD | 00080000h (00080B00h if Asynchronous Schedule Park Capability is a one) |
| USBSTS | 00001000h |
| USBINTR | 00001000h |
| FRINDEX | 00001000h |
| CTRLDSSEGMENT | 00001000h |
| PERIODICLISTBASE | Undefined |
| ASYNCLISTADDR | Undefined |
| CONFIGFLAG | 00000000h |
| PORSC | 00002000h (w/PPC set to one); 00003000h (w/PPC set to a zero) |

505 . Default Values of Operational Register Space

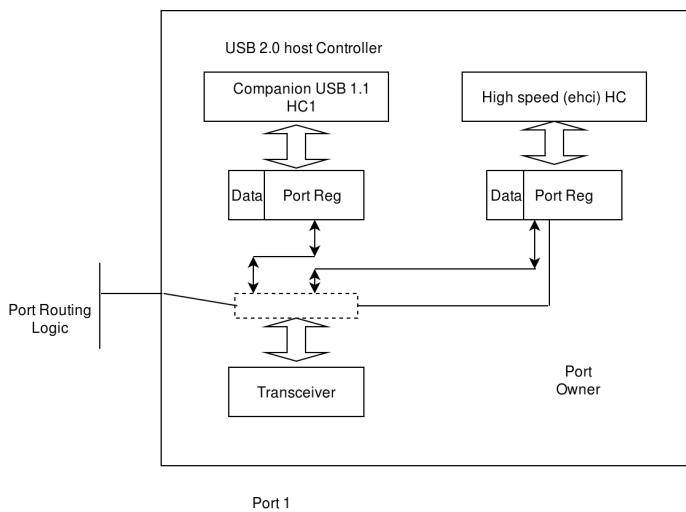
In order to initialize the host controller, software should perform the following steps

- Program the CTRLDSSEGMENT register with 4Gigabyte segment where all of the interface data structures are allocated.
- Write the appropriate value to the USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the Periodic Frame List should have their TBits set to a one.
- Write the USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller ON via setting the Run/Stop bit.
- Write a 1 to CONFIGFLAG register to route all ports to the EHCI controller.

At this point, the host controller is up and running and the port registers will begin reporting device connects, etc. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled port enabled Highspeed ports, but the schedules have not yet been enabled. The EHCI Host controller will not transmit SOFs to enabled Full or Lowspeed ports. In order to communicate with devices via the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to the Asynchronous Schedule Enable bit in the USBCMD register. In order to communicate with devices via the periodic schedule, system software must enable the periodic schedule by writing a one to the Periodic Schedule Enable bit in the USBCMD register. Note that the schedules can be turned on before the first port is reset (and enabled). Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

Port Routing and Control

A USB 2.0 Host controller is comprised of one highspeed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers. Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0 defined port capability; e.g. Low, Full, and Highspeed capability for every port. Figure 52 illustrates a simple block diagram of the port routing logic and its relationship to the highspeed and companion host controllers within a USB 2.0 host controller.



Port 1

USB 2.0 Host Controller Port Routing Block Diagram

USB 2.0

There exists one transceiver per physical port and each host controller module has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Each transceiver can be controlled by either the EHCI host controller or one companion host controller. Routing logic lies between the transceiver and the port status and control registers. The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a global routing policy control field and port ownership control fields. The Configured Flag (CF) bit (defined in section 3.6.8) is the global routing policy control. At poweron or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports will still work in Full and Lowspeed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multifunction PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (i.e. no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The N_CC field in the Structural Parameter register (HCSPARAMS) indicates whether the controller implementation includes companion host controllers. When N_CC has a nonzero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full or lowspeed devices, the ports will always fail the highspeed chirp during reset and the ports will not be enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a rootport to provide full and lowspeed device connectivity. System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers.

Port Routing Control via EHCI Configured (CF) Bit

Each port in the USB 2.0 host controller can be routed either to a single companion host controller or to the EHCI host controller. The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and port control. The Configured Flag (CF) bit (defined in Section 3.6.8), is used to globally set the policy of the routing logic. Each port register has a Port Owner control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the CF bit transitions from a zero to a one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all Port Owner bits go to zero). While the CFbit is a one, the EHCI Driver can control individual ports' routing via the Port Owner control bit. Likewise, whenever the CF bit transitions from a one to a zero (as a result of Aux power application, HCRESET, or software writing a zero to CFbit), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's CF bit (after Aux power application or HCRESET) is zero. Table 27 summarizes the default routing for all the ports, based on the value of the EHCI HC's CF bit. The view of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

| HS CF Bit | Default Port Ownership | Explanation |
|-----------|------------------------|--|
| 0B | Companion HCs | The companion host controllers own the ports and only Full and Lowspeed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full and Lowspeed ports in this configuration |
| 1B | EHCI HC | The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC. The EHCI HC has access to the additional port status and control bits defined in this specification (see Section 3.6.16). The EHCI HC can temporarily release control of the port to a companion HC by setting the PortOwner bit in the PORTSC register to a one. |

506 . Default Port Routing Depending on EHCI HC CF Bit

Port Routing Control via PortOwner and Disconnect Event

Manipulating the port routing via the CFbit is an extreme process and not intended to be used during normal operation. The normal mode of port ownership transferal is on the granularity of individual ports using the Port Owner bit in the EHCI HC's PORTSC register (for handoffs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI CFbit is set to a one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a portchange interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.
- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the LineStatus bits in the PORTSC register. If they indicate the attached device is a fullspeed device (e.g. D+ is asserted), then the EHCI Driver sets the PortReset control bit to a one (and sets the PortEnable bit to a zero) which begins the resetprocess. Software times the duration of the reset, then terminates reset signaling by writing a zero to the port reset bit. The reset process is actually complete when software reads a zero in the PortReset bit. The EHCI Driver

checks the PortEnable bit in the PORTSC register. If set to a one, the connected device is a highspeed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.

- At the time the EHCI Driver receives the port reset and enable request the LineStatus bits might indicate a low speed device. Additionally, when the port reset process is complete, the PortEnable field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the PORTSC register to a one to release port ownership to a companion host controller.
- When the EHCI Driver sets PortOwner bit to a one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI PORTSC register observes and reports a disconnect event via the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that portstate, the EHCI Driver responds with a reset complete change set to a one, a connect change set to a one and a connect status set to a zero. This information is derived directly from the EHCI port register. This will allow the hub driver to assume the device was disconnected during reset. It will acknowledge the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected. The deviceconnect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's CFbit transitions from a 1b to a 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects will be detected by the EHCI port register and the process will repeat.

Port Power

The Port Power Control (PPC) bit in the HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (See section HCSPARAMS – EHCI Compliant with extensions). When this bit is a zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the PPC bit is a one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as PortPowerOutputEnable (PPE). PPE is controlled based on the state of the combination bits PPC bit, EHCI Configured (CF)bit and individual Port Power (PP) bits. Table 28 illustrates the summary behavioral model.

| CF | CHC(PP) | EHC(PP) | Owner | PPE | Description |
|----|---------|---------|-------|-----|---|
| 0 | 0 | X | CHC | 0 | When the EHCI controller has not been configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off. |
| 0 | 1 | X | CHC | 1 | Similar to previous entry. When the companion HC's port power select is on, then the port power is on. |
| 1 | 0 | 0 | CHC | 0 | Port owner has port power turned off, the power to port is off. |
| 1 | 0 | 0 | EHC | 0 | Port owner has port power turned off, the power to port is off. |

| CF | CHC(PP) | EHC(PP) | Owner | PPE | Description |
|----|---------|---------|-------|-----|---|
| 1 | 0 | 1 | EHC | 1 | Port owner has port power on, so power to port is on. |
| 1 | 0 | 1 | CHC | 1 | If either HC has port power turned on, the power to the port is on. |
| 1 | 1 | 0 | EHC | 1 | If either HC has port power turned on, the power to the port is on. |
| 1 | 1 | 0 | CHC | 1 | Port owner has port power on, so power to port is on. |
| 1 | 1 | 1 | CHC | 1 | Port owner has port power on, so power to port is on. |
| 1 | 1 | 1 | EHC | 1 | Port owner has port power on, so power to port is on. |

507 . Port Power Enable Control Rules

PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

CHC (Companion Host Controller).

EHC (EHCI Host Controller).

Port Reporting OverCurrent

Host controllers are by definition power providers on USB. Whether the ports are considered high or low powered is a platform implementation issue. Each EHCI PORTSC register has an overcurrent status and over current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0. The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an overcurrent condition it is made available to both them companion and EHCI ports. The effect of an overcurrent status on a companion host controller port is beyond the scope of this document. The overcurrent condition effects the following bits in the PORTSC register on the EHCI port:

- Overcurrent Active bits are set to a one. When the overcurrent condition goes away, the Overcurrent Active bit will transition from a one to a zero.
- Overcurrent Change bits are set to a one. On every transition of the Overcurrent Active bit the host controller will set the Overcurrent Change bit to a one. Software sets the Overcurrent Change bit to a zero by writing a one to this bit.
- Port Enabled/Disabled bit is set to a zero. When this change bit gets set to a one, then the Port Change Detect bit in the USBSTS register is set to a one.
- Port Power (PP) bits may optionally be set to a zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to limit the current and leave power applied. When the Overcurrent Change bit transitions from a zero to a one, the host controller also sets the Port Change Detect bit in the USBSTS register to a one. In addition, if the Port Change Interrupt Enable bit in the USBINTR register is a one, then the host controller will issue an interrupt to the system. Refer to Table 29 for summary behavior for overcurrent detection when the host controller is halted (suspended from a device component point of view).

Suspend/Resume

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub. Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely via software initiation. Other control

mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, hostinitiated, or software initiated resumes are called Resume Events/Actions. Businitiated resume events are called wakeup events. The classes of wakeup events are:

- Remotewakeup enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake up the system (if necessary).
- Port connect and disconnect and overcurrent events. Sensitivity to these events can be turned on or off by using the port control bits in the PORTSC registers.

Selective suspend is a feature supported by every PORTSC register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the Run/Stop bit in the USBCMD register to a zero. The EHCI module can then be placed into a lower device state via the PCI power management interface . When a wake event occurs the system will resume operation and system software will eventually set the Run/Stop bit to a one and resume the suspended ports. Software must not set the Run/Stop bit to a one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the CPU is restarted. So, by definition, if software is running, clocks in the system are stable and the Run/Stop bit in the USBCMD register can be set to a one. There are also minimum system software delays defined in the PCI Power Management Specification. Refer to this specification for more information.

Port Suspend/Resume

System software places individual ports into suspend mode by writing a one into the appropriate PORTSC Suspend bit. Software must only set the Suspend bit when the port is in the enabled state (Port Enabled bit is a one) and the EHCI is the port owner (Port Owner bit is a zero). The host controller may evaluate the Suspend bit immediately or wait until a microframe or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several microframes of activity on the port until the host controller evaluates the Suspend bit. The host controller must evaluate the Suspend bit at least every frame boundary. System software can initiate a resume on a selectively suspended port by writing a one to the Force Port Resume bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state. If system software sets Force Port Resume bit to a one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 milliseconds after a port indicates that it is suspended (Suspend bit is a one) before initiating a port resume via the Force Port Resume bit. When Force Port Resume bit is a one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds) then sets the Force Port Resume bit to a zero. When the host controller receives the write to transition Force Port Resume to zero, it completes the resume sequence as defined in the USB specification, and sets both the Force Port Resume and Suspend bits to zero. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 μ sec. The port's Force Port Resume bit is set to a one and the Port Change Detect bit in the USBSTS register is set to a one. If the Port Change Interrupt Enable bit in the USBINTR register is a one the host controller will issue a hardware interrupt. System software observes the resume event on the port, delays a port resume time (nominally 20 msec), then terminates the resume sequence by writing zero to the Force Port Resume bit in the port. The host controller receives the write of zero to Force Port Resume, terminates the resume sequence and sets Force Port Resume and Suspend port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the PORTSC register and observing that the Suspend and Force Port Resume bits are zero. Software must ensure that the host controller is running (i.e. HCHalted bit in the USBSTS register is a zero), before terminating a resume by writing a zero to a port's Force Port Resume bit. If HCHalted is a one when Force Port Resume is set to a zero, then SOFs will not occur down the enabled port and the device will return to suspend mode in a maximum of 10 milliseconds. Table 29 summarizes the wakeup events. Whenever a resume event is detected, the Port Change Detect bit in the USBSTS register is set to a one. If the Port Change Interrupt Enable bit is a one in the USBINTR register, the host controller will also generate an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the Port Change Detect status bit in the USBSTS register.

| Port Status and Signaling Type | Signaled Port Response | Device State(D0) | Device State(D1) |
|--|---|------------------|------------------|
| Port disabled, resume K State received | No Effect | N/A | N/A |
| Port suspended, Resume KState received | Resume reflected downstream on signaled port. Force Port Resume status bit in PORTSC register is set to a one. Port Change Detect bit in USBSTS register set to a one. | [1], [2] | [2] |
| Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a one. A disconnect is detected. | Depending in the initial port state, the PORTSC Connect and Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one. | [1], [2] | [2] |
| Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is a zero. A disconnect is detected. | Depending on the initial port state, the PORTSC Connect and Enable status bits are set to zero, and the Connect Change status bit is set to a one. Port Change Detect bit in the USBSTS register is set to a one. | [1], [3] | [3] |
| Port is not connected and the port's WKCNNT_E bit is a one. A connect is detected. | PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one. | [1], [2] | [2] |
| Port is not connected and the port's WKCNNT_E bit is a zero. A connect is detected. | PORTSC Connect Status and Connect Status Change bits are set to a one. Port Change Detect bit in the USBSTS register is set to a one. | [1], [3] | [3] |
| Port is not connected and port's WKOC_E bit is a one. An overcurrent condition occurs. | PORTSC Overcurrent Active, Overcurrent Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one. | [1], [2] | [2] |
| Port is connected and the port's WKOC_E bit is a zero. An overcurrent condition occurs. | PORTSC Overcurrent Active, Overcurrent Change bits are set to a one. If Port Enable/Disable bit is a one, it is set to a zero. Port Change Detect bit in the USBSTS register is set to a one. | [1], [3] | [3] |

508 . Behavior During Wakeup Events

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USBINTR register is a one.

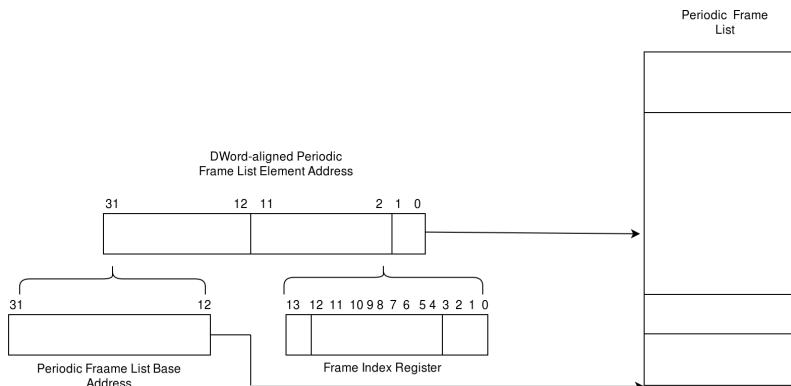
[2] PME# asserted if enabled (Note: PME Status must always be set to a one).

[3] PME# not asserted.

15.6.10 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, sharedmemory schedule. The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware / software complexity. System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The

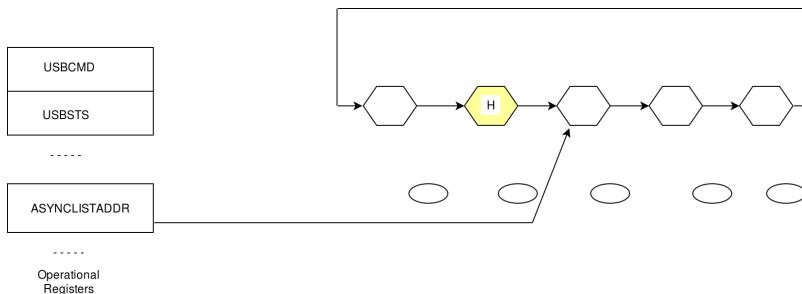
root of the periodic schedule is the PERIODICLISTBASE register (see Section PERIODICLISTBASE; DEVICEADDR). The PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in 4. In each microframe, if the periodic schedule is enabled (see Section Periodic Schedule) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It will only execute from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the PERIODICLISTBASE and the FRINDEX registers (see Figure 54). It fetches the element and begins traversing the graph of linked schedule data structures. The end of the periodic schedule is identified by a next link pointer of a schedule data structure having its Tbit set to a one. When the host controller encounters a TBit set to a one during a horizontal traversal of the periodic list, it interprets this as an EndOfPeriodicList mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. Once this transition is made, the host controller executes from the asynchronous schedule until the end of the microframe.



Derivation of Pointer into Frame List Array

Derivation of Pointer into Frame List Array

When the host controller determines that it is time to execute from the asynchronous list, it uses the operational register ASYNCLISTADDR to access the asynchronous schedule, see Figure 55.



General Format of Asynchronous Schedule List

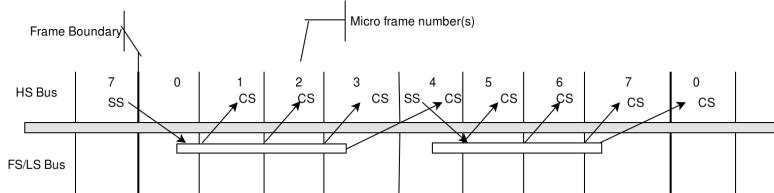
General Format of Asynchronous Schedule List

The ASYNCLISTADDR register contains a physical memory pointer to the next queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the ASYNCLISTADDR register. Software must set queue head horizontal pointer Tbits to a zero for queue heads in the asynchronous schedule. See Section Asynchronous Schedule for complete operational details.

Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full and lowspeed bus(s) below USB 2.0 Hubs be strictly aligned. Superimposed on this requirement is that USB 2.0 Hubs manage full and lowspeed transactions via a microframe pipeline (see start (SS)

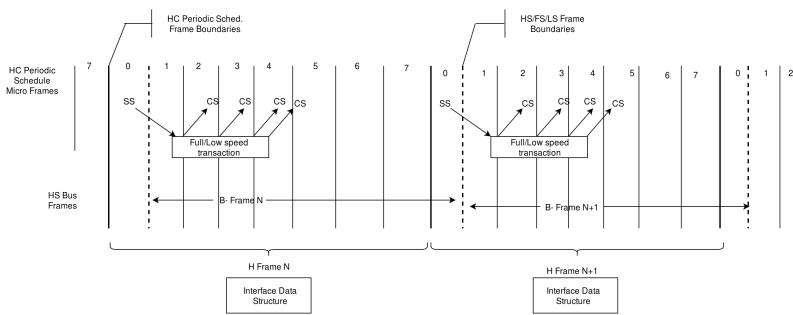
and complete (CS) splits illustrated in Figure 57). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full and lowspeed transaction translator periodic pipelines.



Frame Boundary Relationship between HS bus and FS/LS Bus

Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as Figure 57 illustrates, introduces frameboundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement a one microframe phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and framewrap scheduling boundary conditions. The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed via the Frame List Index Register (FRINDEX) documented in Section FRINDEX and initially illustrated in Section Schedule Traversal Rules. Bits FRINDEX[2:0], represent the microframe number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are incremented based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one microframe. The one microframe delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in Figure 58. This adjustment allows software to trivially\ schedule the periodic start and completesplit transactions for fulland lowspeed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phaseshift are beyond the scope of this specification. Figure 58 illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined. The host controller's view of the 1millisecond boundaries is called HFrames. The highspeed bus's view of the 1millisecond boundaries is called BFrames.



Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

HFrame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Microframe numbers for the HFrame are tracked by FRINDEX[2:0]. BFrame boundaries are visible on the highspeed bus via changes in the SOF token's frame number. Microframe numbers on the highspeed bus are only derived from the SOF token's frame number (i.e. the highspeed bus will see eight SOFs with the same frame number value). H Frames and BFrames have the fixed relationship (i.e. BFrames lag HFrames by one microframe time) illustrated in Figure 58. The host controller's periodic schedule is naturally aligned to HFrames. Software schedules transactions for full and low-speed periodic endpoints relative the HFrames. The result is these transactions execute on the highspeed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in Section FRINDEX, the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one microframe count. Table 31 illustrates the required relationship between the value of FRINDEX and the value of SOFV. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carryout on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0]. Software is

allowed to write to FRINDEX. Section FRINDEX provides the requirements that software should adhere when writing a new value in FRINDEX.

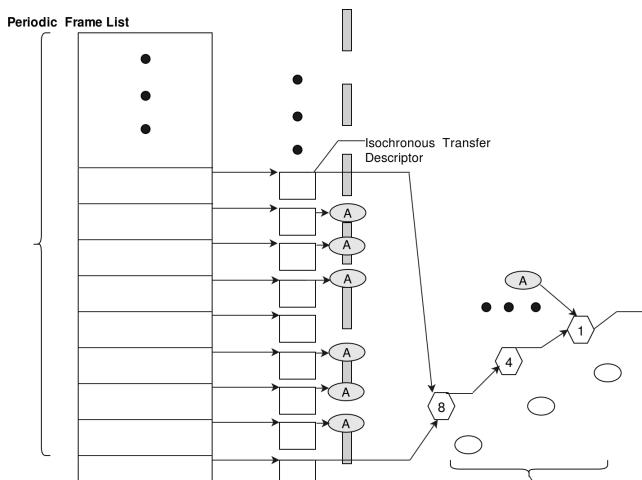
| FRINDEX[F] | SOFV | FRINDEX[F] | FRINDEX[F] | SOFV | FRINDEX[F] |
|------------|------|------------|------------|------|------------|
| N | N | 111b | N+1 | N | 000b |
| N+1 | N | 000b | N+1 | N+1 | 001b |
| N+1 | N+1 | 001b | N+1 | N+1 | 010b |
| N+1 | N+1 | 010b | N+1 | N+1 | 011b |
| N+1 | N+1 | 011b | N+1 | N+1 | 100b |
| N+1 | N+1 | 100b | N+1 | N+1 | 101b |
| N+1 | N+1 | 101b | N+1 | N+1 | 110b |
| N+1 | N+1 | 110b | N+1 | N+1 | 111b |

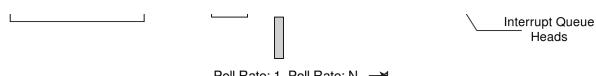
509 . Operation of FRINDEX and SOFV (SOF Value Register)

Periodic Schedule

The periodic schedule traversal is enabled or disabled via the Periodic Schedule Enable bit in the USBCMD register. If the Periodic Schedule Enable bit is set to a zero, then the host controller simply does not try to access the periodic frame list via the PERIODICLISTBASE register. Likewise, when the Periodic Schedule Enable bit is a one, then the host controller does use the PERIODICLISTBASE register to traverse the periodic schedule. The host controller will not react to modifications to the Periodic Schedule Enable immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the Periodic Schedule Enable bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b microframe. These work items must be removed from the schedule before the Periodic Schedule Enable bit is written to a zero. The Periodic Schedule Status bit in the USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing a one (or zero) to the Periodic Schedule Enable bit in the USBCMD register. Software then can poll the Periodic Schedule Status bit to determine when the periodic schedule has made the desired transition. Software must not modify the Periodic Schedule Enable bit unless the value of the Periodic Schedule Enable bit equals that of the Periodic Schedule Status bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the USB. Figure 59 illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the periodone iTD/siTDS. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (e.g. closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.





Example Periodic Schedule

Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (HighSpeed) Transfer Descriptor (iTID). There are four distinct sections to an iTD:

- The first field is the Next Link Pointer. This field is for schedule linkage purposes only;
- Transaction description array. This area is an eightelement array. Each element represents control and status information for one microframe's worth of transactions for a single highspeed isochronous endpoint.
- The buffer page pointer array is a 7element array of physical memory pointers to data buffers. These are 4K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused loworder 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and highbandwidth multiplier.

Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Each iTD can span 8 microframes worth of transactions. When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array. If the active bit in the Status field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the Next pointer to the next schedule data structure.

When the indexed active bit is a one the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, etc.). It also uses the Page Select (PG) field to index the buffer pointer array, storing the selected buffer pointer and the next sequential buffer pointer. For example, if PG field is a 0, then the host controller will store Page 0 and Page 1. The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's PG field) and the transaction description's Transaction Offset field. The host controller uses the endpoint addressing information and I/Obit to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the Status field in the currently selected transaction description. The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer(example: page 0 pointer) selected by the active transaction descriptions' PG (example value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer will cross a page boundary. When this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (example: page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the Maximum Packet Size field. An iTD supports high bandwidth pipes via the Mult (multiplier) field. When the Mult field is 1, 2, or 3, the host controller executes the specified number of Maximum Packet sized bus transactions for the endpoint in the current microframe. In other words, the Mult field represents a transaction count for the endpoint in the current microframe. If the Mult field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the Mult field.

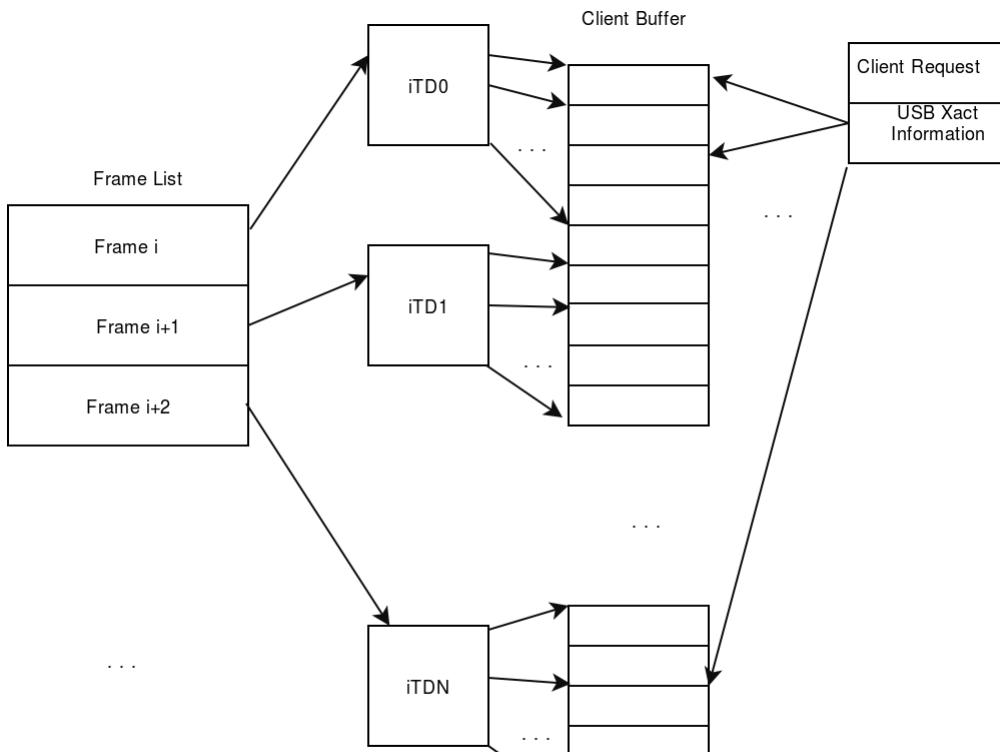
For OUT transfers, the value of the Transaction X Length field represents the total bytes to be sent during the microframe. The Mult field must be set by software to be consistent with Transaction X Length and Maximum Packet Size. The host controller will send the bytes in Maximum Packet Size'd portions. After each transaction, the host controller decrements its local copy of Transaction X Length by Maximum Packet Size. The number of bytes the host controller sends is always Maximum Packet Size or Transaction X Length, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes. mFor IN transfers, the host controller issues Mult transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use Maximum Packet Size to detect packet babble errors. The host controller keeps the sum of bytes received in the Transaction X Length

field. After all transactions for the endpoint have completed for the microframe, Transaction X Length contains the total bytes received. If the final value of Transaction X Length is less than the value of Maximum Packet Size, then less data than was allowed for was received from the associated endpoint. This short packet condition does not set the USBINT bit in the USBSTS register to a one. The host controller will not detect this condition. If the device sends more than Transaction X Length or Maximum Packet Size bytes (whichever is less) then the host controller will set the Babble Detected bit to a one and set the Active bit to a zero. Note, that the host controller is not required to update the iTD field Transaction X Length in this error scenario. If the Mult field is greater than one, then the host controller will automatically execute the value of Mult transactions. The host controller will not execute all Mult transactions if:

- The endpoint is an OUT and Transaction X Length goes to zero before all the Mult transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before Mult transactions have been executed. The end of microframe may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next microframe. Refer to Appendix D for a table summary of the host controller required behavior for all the high-bandwidth transaction cases.

Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N microframes. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD). Figure 60 illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (i.e. the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one microframe's worth of transactions. The EHCI controller does not provide pertransaction results within a microframe. It treats the permicroframe transactions as a single logical transfer. On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.





Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the Transaction Offset field is set relative to the correct buffer pointer page (e.g. the same one referenced by the PG field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current Page Buffer Pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available Page Buffer Pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer will wrap a page boundary. Doing so will yield undefined behavior. The host controller hardware is not required to 'alias' the page selector to page zero. USB 2.0 isochronous endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to a one.

Periodic Scheduling Threshold

The Isochronous Scheduling Threshold field in the HCCPARAMS capability register is an indicator to system software as to how the host controller prefetches and effectively caches schedule data structures. It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them. The iTD and siTD data structures each describe 8 microframes worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span 8 microframes. The three caching models are: no caching, microframe caching and frame caching. When software is adding new isochronous transactions to the schedule, it always performs a read of the FRINDEX register to determine the current frame and microframe the host controller is currently executing. Of course, there is no information about where in the microframe the host controller is, so a constant uncertainty factor of one microframe has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller. No caching is indicated with a value of zero in the Isochronous Scheduling Threshold field. The host controller may prefetch data structures during a periodic schedule traversal (per microframe) but will always dump any accumulated schedule state at the end of the microframe. At the appropriate time relative to the beginning of every microframe, the host controller always begins schedule traversal from the frame list. Software can use the value of the FRINDEX register (plus the constant 1 uncertainty factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 microframes in front of the current executing position of the host controller. Frame caching is indicated with a nonzero value in bit [7] of the Isochronous Scheduling Threshold field. In the framecaching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 microframes). Software uses the value of the FRINDEX register (plus the constant 1 uncertainty) to determine the current microframe/frame (assume modulo 8 arithmetic in adding the constant 1 to the microframe number). For any current frame N, if the current microframe is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current microframe is 7, then software can add isochronous transactions to Frame N + 2. Microframe caching is indicated with a nonzero value in the leastsignificant 3 bits of the Isochronous Scheduling Threshold field. System software assumes the host controller caches one or more periodic data structures for the number of microframes indicated in the Isochronous Scheduling Threshold field. For example, if the count value were 2, then the host controller keeps a window of 2 microframes worth of state (current microframe, plus the next) onchip. On each microframe boundary, the host controller releases the current microframe state and begins accumulating the next microframe state.

Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled via the Asynchronous Schedule Enable bit in the USBCMD register. If the Asynchronous Schedule Enable bit is set to a zero, then the host controller simply does not try to access the asynchronous schedule via the ASYNCLISTADDR register. Likewise, when the Asynchronous

Schedule Enable bit is a one, then the host controller does use the ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to the Asynchronous Schedule Enable bit are not necessarily immediate. Rather the new value of the bit will only be taken into consideration the next time the host controller needs to use the value of the ASYNCLISTADDR register to get the next queue head. The Asynchronous Schedule Status bit in the USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to the Asynchronous Schedule Enable bit in the USBCMD register. Software then can poll the Asynchronous Schedule Status bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the Asynchronous Schedule Enable bit unless the value of the Asynchronous Schedule Enable bit equals that of the Asynchronous Schedule Status bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the ASYNCLISTADDR register. The default value of the ASYNCLISTADDR register after reset is undefined and the schedule is disabled when the Asynchronous Schedule Enable bit is a zero. Software may only write this register with defined results when the schedule is disabled e.g. Asynchronous Schedule Enable bit in the USBCMD and the Asynchronous Schedule Status bit in the USBSTS register are zero. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the Asynchronous Schedule Enable bit is set to one. The asynchronous schedule is actually enabled when the Asynchronous Schedule Status bit is a one. When the host controller begins servicing the asynchronous schedule, it begins by using the value of the ASYNCLISTADDR register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller "completes" processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the ASYNCLISTADDR register. Next time the asynchronous schedule is accessed, this is the first data structure that will be serviced. This provides roundrobin fairness for processing the asynchronous schedule.

A host controller "completes" processing the asynchronous schedule when one of the following events occur:

- The end of a microframe occurs.
- The host controller detects an empty list condition (i.e. see Section 5.8.3)
- The schedule has been disabled via the Asynchronous Schedule Enable bit in the USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 55. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTID or siTD) in the asynchronous schedule yields undefined results. The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all nonisochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

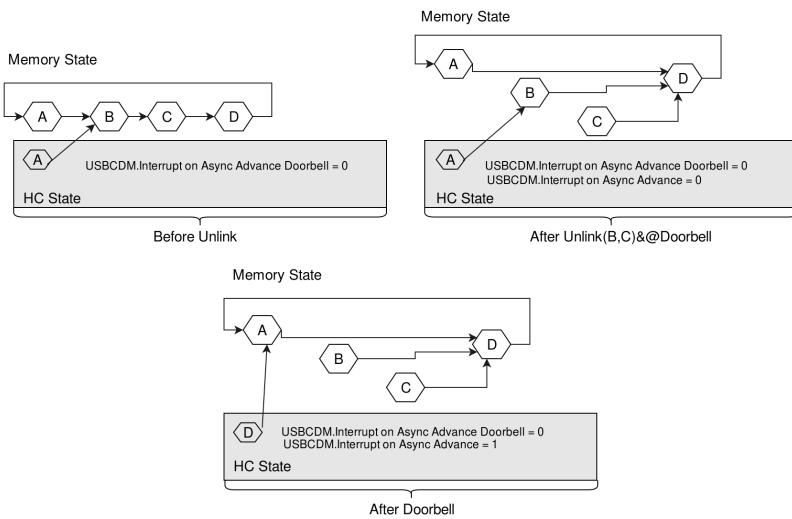
Adding Queue Heads to Asynchronous Schedule

This is a software requirement section. There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list. Activation of the list is simple. System software writes the physical memory address of a queue head into the ASYNCLISTADDR register, then enables the list by setting the Asynchronous Schedule Enable bit in the USBCMD register to a one. When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example qTD pointers have TBits set to a one or reference valid qTDs and the Horizontal Pointer references a valid queue head data structure.

Removing Queue Heads from Asynchronous Schedule

This is a software requirement section. There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list. Software deactivates the asynchronous schedule by setting the Asynchronous Schedule Enable bit in the USBCMD register to a zero. Software can determine when the list is idle when the Asynchronous Schedule Status bit in the USBSTS register is a zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list via the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

If software removes the queue head with the Hbit set to a one, it must select another queue head still linked into the schedule and set its Hbit to a one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its Hbit set to a one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, etc.). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures. The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule. The handshake is implemented with three bits in the host controller. The first bit is a command bit (Interrupt on Async Advance Doorbell bit in the USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (Interrupt on Async Advance bit in the USBSTS register) that the host controller sets after it has released all onchip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to a one, it also sets the command bit to a zero. The third bit is an interrupt enable (Interrupt on Async Advance bit in the USBINTR register) that is matched with the status bit. If the status bit is a one and the interrupt enable bit is a one, then the host controller will assert a hardware interrupt. Figure 61 illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A. The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that will remain in the asynchronous schedule. When the host controller observes that doorbell bit being set to a one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A & B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (i.e. traversed beyond queue head (B) in this example).



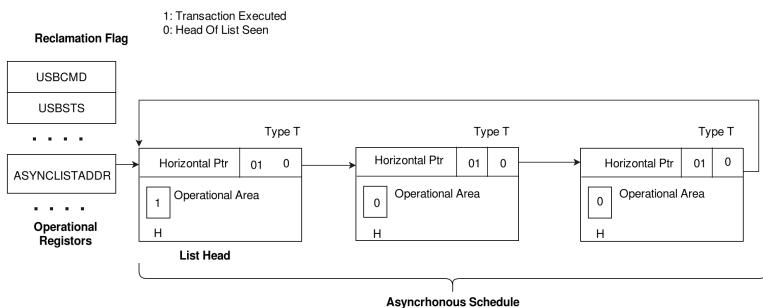
Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (e.g. observed the head of the queue (twice)) before setting the Advance on Async status bit to a one. Software may reuse the memory associated with the removed queue heads after it observes the Interrupt on Async Advance status bit is set to a one, following assertion of the doorbell. Software should acknowledge the Interrupt on Async Advance status as indicated in the USBSTS register, before using the doorbell handshake again.

Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty. The queue head data structure (see) defines an Hbit in the queue head, which allows software to mark a queue head as being the head of the reclaim list. The Enhanced Host Controller Interface also keeps a 1bit flag in the USBSTS register (Reclamation) that is set to a zero when the Enhanced Interface Host Controller observes a queue head with the Hbit set to a one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Section Asynchronous

Schedule Traversal : Start Event). If the Enhanced Host Controller Interface ever encounters an Hbit of one and a Reclamation bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule. An example illustrating the Hbit in a schedule is illustrated in Figure 62.



Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the Hbit set to a one, and that it is always coherent with respect to the schedule.

Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list long before the end of the microframe. It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints. An example of particular interest is when a startsplit for a bulk endpoint occurs early in the microframe. Given the EHCI simple traversal rules, the completesplit for that transaction may Nak/Nyet out very quickly. If it is the only item in the schedule, then the host controller will cease traversal of the Asynchronous schedule very early in the microframe. In order to provide reasonable service to this endpoint, the host controller should issue the completesplit before the end of the current microframe, instead of waiting until the next microframe. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule.

Asynchronous Schedule Traversal : Start Event

Once the HC has idled itself via the empty schedule detection (Section 5.8.3), it will naturally activate and begin processing from the Periodic Schedule at the beginning of each microframe. In addition, it may have idled itself early in a microframe. When this occurs (idles early in the microframe) the HC must occasionally reactivate during the microframe and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Section 5.8.4. Asynchronous schedule Start Events are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the microframe is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Section 5.8.4).

Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (Section Empty Asynchronous Schedule Detection) depends on the proper management of the Reclamation bit in the USBSTS register. The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (See Section Fetch Queue Head). It is required that the host controller sets the Reclamation bit to a one whenever an asynchronous schedule traversal Start Event, as documented in Section 5.8.5, occurs. The Reclamation bit is also set to a one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Section 5.10.3). The host controller sets the Reclamation bit to a zero whenever it finds a queue head with its Hbit set to a one. Software should only set a queue head's Hbit if the queue head is in the asynchronous schedule. If software sets the Hbit in an interrupt queue head to a one, the resulting behavior is undefined. The host controller may set the Reclamation bit to a zero when executing from the periodic schedule.

Operational Model for Nak Counter

This section describes the operational model for the NakCnt field defined in a queue head (see Section Queue Head). Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller. USB protocol has builtin flow control via the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High or Fullspeed) are serviced via the same asynchronous schedule. The time between the Startsplits transaction and the first Completesplit transaction could be very short (i.e. like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the Completesplit transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which will not complete until after the transaction on the classic bus completes.

There are two component fields in a queue head to support the throttling feature: a counter field (NakCnt), and a counter reload field (RL). NakCnt is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. There are two operational modes associated with this counter:

- Not Used. This mode is set when the RL field is zero. The host controller ignores the NakCnt field for any execution of transactions through a queue head with an RL field of zero. Software must use this selection for interrupt endpoints.
- Nak Throttle Mode. This mode is selected when the RL field is nonzero. In this mode, the value in the NakCnt field represents the maximum number of Nak or Nyet responses the host controller will tolerate on each endpoint. In this mode, the HC will decrement the NakCnt field based on the token/handshake criteria listed in Table 34. The host controller must reload NakCnt when the endpoint successfully moves data (e.g. policy to reward device for moving data).

| Token | Handshake NAK | Handshake NYET |
|----------------|------------------|----------------------|
| IN/PING | decrement NakCnt | N/A (protocol error) |
| OUT | decrement NakCnt | No Action1 Start |
| Split | decrement NakCnt | N/A (protocol error) |
| Complete Split | No Action | Decrement NakCnt |

510 . NakCnt Field Adjustment Rules

Nak Count Reload Control

When the host controller reaches the Execute Transaction state for a queue head (meaning that it has an active operational state), it checks to determine whether the NakCnt field should be reloaded from RL (see Section Execute Transaction). If the answer is yes, then RL is copied into NakCnt. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction. The host controller must reload nak counters (NakCnt see) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Section 5.8.5 for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 62).

Wait for List Head

This is the initial state. The state machine enters this state from Wait for Start Event when a start event as defined in Section Asynchronous Schedule Traversal : Start Event occurs. The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule. This occurs when the host controller fetches a queue head whose Hbit is set to a one.

Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the Hbit set to a one. While in this state, the host controller will perform nak counter reloads for every queue head visited that has a nonzero nak reload value (RL) field.

Wait for Start Event

This state is entered from the Do Reload state when a queue head with the Hbit set to a one is fetched. While in this state, the host controller will not perform nak counter reloads.

Managing Control/Bulk/Interrupt Transfers via Queue Heads

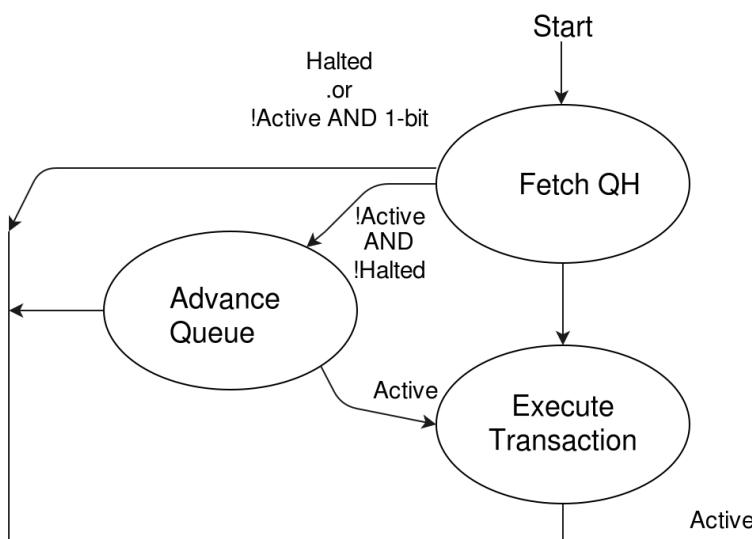
This section presents an overview of how the host controller interacts with queuing data structures. Queue heads use the Queue Element Transfer Descriptor (qTD) structure defined in Section . One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a transfer.

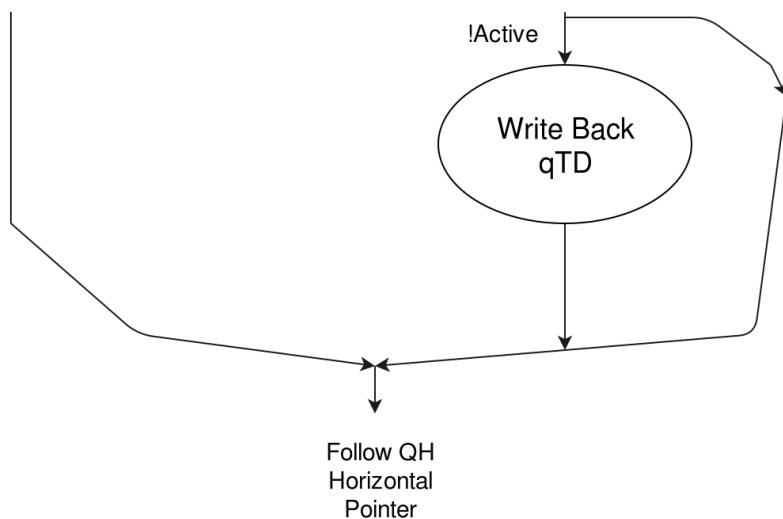
The general processing model for the host controller's use of a queue head is simple:

- read a queue head,
- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller will set one (or more) of the error reporting bits in the queue head's Status field. The Status field accumulates all errors encountered during the execution of a qTD (e.g. the error bits in the queue head Status field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (e.g. buffer or halt conditions) boundaries, the host controller must autoadvance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions will occur for the endpoint and the host controller will not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in Figure 65. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the Fetch QH state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and activate them, then the host controller will always find them if/when they are reachable.





Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (Section Execute Transaction) describes the basic requirements for all endpoints. Sections Split Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions. Note: Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section 4.6 for layout of a queue head):

- Valid static endpoint state
- For the very first use of a queue head, software may zeroout the queue head transfer overlay, then set the Next qTD Pointer field value to reference a valid qTD.

Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (Section 3.6.6.2). Additionally, it may be referenced from the Next Link Pointer field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the Typ field set to indicate a queue head, it is assumed to reference a queue head structure as defined in . While in this state, the host controller performs operations to implement empty schedule detection (Section Empty Asynchronous Schedule Detection) and Nak Counter reloads (Section Operational Model for Nak Counter). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (i.e. Smask is a zero), and
- The Hbit is a one, and
- The Reclamation bit in the USBSTS register is a zero.

When these criteria are met, the host controller will stop traversing the asynchronous list (as described in Section Empty Asynchronous Schedule Detection). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the Hbit is a one and the Reclamation bit is a one, then the host controller sets the Reclamation bit in the USBSTS register to a zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Section Operational Model for Nak Counter. This state is complete when the queue head has been read onchip.

Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head. This state is entered from the FetchQHD state if the overlay Active and Halt bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay. Note that if the lbit is a one and the Active bit is a zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal

pointer to the next schedule data structure. If the field Bytes to Transfer is not zero and the Tbit in the Alternate Next qTD Pointer is set to zero, then the host controller uses the Alternate Next qTD Pointer. Otherwise, the host controller uses the Next qTD Pointer. If Next qTD Pointer's Tbit is set to a one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure. Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its Active bit set to a one, the host controller moves the pointer value used to reach the qTD (Next or Alternate Next) to the Current qTD Pointer field, then performs the overlay. If the fetched qTD has its Active bit set to a zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure. The host controller performs the overlay based on the following rules:

- The value of the data toggle (dt) field in the overlay area depends on the value of the data toggle control (dtc) bit (see Table 22).
- If the EPS field indicates the endpoint is a highspeed endpoint, the Ping state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- Cprogmask field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- Frame Tag field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- NakCnt field in the overlay area is loaded from the RL field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

Execute Transaction

The host controller enters this state from the Fetch Queue Head state only if the Active bit in Status field of the queue head is set to a one. On entry to this state, the host controller executes a few preoperations, then checks some precondition criteria before committing to executing a transaction for the queue head. The preoperations performed and the precondition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's S mask field contains a nonzero value. It is the responsibility of software to ensure the Smask field is appropriately initialized based on the transfer type. There are other criteria that must be met if the EPS field indicates that the endpoint is a low or fullspeed endpoint, see Sections 5.12.1 and 5.12.2.

• Interrupt Transfer Precondition Criteria

If the queue head is for an interrupt endpoint (e.g. nonzero Smask field), then the FRINDEX[2:0] field must identify a bit in the Smask field that has a one in it. For example, an Smask value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

• Asynchronous Transfer Preoperations and Precondition Criteria

If the queue head is not for an interrupt endpoint (e.g. a zero Smask field), then the host controller performs one preoperation and then evaluates one precondition criteria: The preoperation is:

--Checks the Nak counter reload state (Section 5.9). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The precondition evaluated is:

-- Whether or not the NakCnt field has been reloaded, the host controller checks the value of the NakCnt field in the queue head. If NakCnt is nonzero, or if the Reload Nak Counter field is zero, then the host controller considers this queue head for a transaction.

• Transfer Type Independent Preoperations

Regardless of the transfer type, the host controller always performs at least one preoperation and evaluates one precondition. The preoperation is:

--A host controller internal transaction (down) counter qHTransactionCounter is loaded from the queue head's Mult field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the Mult field is set appropriately for the transfer type.

The preconditions evaluated are:

--The host controller determines whether there is enough time in the microframe to complete this transaction(see Section 5.4.1.1 for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.

-- If the value of qHTransactionCounter for an interrupt endpoint is zero, then the host controller exits this state.

When the preoperations are complete and preconditions are met, the host controller sets the Reclamation bit in the USBSTS register to a one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates qHTransactionCounter times in this state executing transactions. After each transaction is executed, qHTransactionCounter is decremented by one. The host controller will exit this state when one of the following events occurs:

- The qHTransactionCounter decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,4 or
- The transaction experiences a transaction error, or

The Active bit in the queue head goes to a zero, or

There is not enough time in the microframe left to execute the next transaction .

The results of each transaction is recorded in the onchip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the Total Bytes to Transfer field is decremented by the number of bytes moved in the transaction, the data toggle bit (dt) is toggled, the current page offset is advanced to the next appropriate value (e.g. advanced by the number of bytes successfully moved), and the C_Page field is updated to the appropriate value (if necessary). See Section 5.10.6. Note that the Total Bytes To Transfer field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller will execute a zerolength transaction to the endpoint. If the PID_Code field indicates an IN transaction and the device delivers data, the host controller will detect a packet babble condition, set the babble and halted bits in the Status field, set the Active bit to a zero, write back the results to the source qTD, then exit this state. In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (e.g. not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (e.g. decrement qHTransactionCounter). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of qHTransactionCounter is greater than one. If the response to the IN bus transaction is a Nak (or Nyet) and RL is nonzero, NakCnt is decremented by one. If RL is zero, then no write back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of CEErr did not change). Software should set the RL field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation. After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly NakCnt, the host controller writes back the results of the transaction to the queue head's overlay area in main memory. The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is Maximum Packet Size. The number of bytes moved during an OUT transaction is either Maximum Packet Length bytes or Total Bytes to Transfer, whichever is less. If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The CEErr field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Section Transaction Error.

The following events will cause the host controller to clear the Active bit in the queue head's overlay status field. When the Active bit transitions from a one to a zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the Active bit) determines the next state.

- CEErr field decrements to zero. When this occurs the Halted bit is set to a one and Active is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the Halted bit is set to a one and the Active bit is set to a zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The Total Bytes to Transfer field is zero after the transaction completes. Note that for a zero length transaction, it was zero before the transaction was started. When this condition occurs, the Active bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the Maximum Packet Length. When this occurs, the Active bit is set to zero and a short packet condition exists. The shortpacket condition is detected during the Advance Queue state. Refer to Section 5.12 for additional rules for managing low and full-speed transactions.

- The PID Code field indicates an IN and the device sends more than the expected number of bytes (e.g. Maximum Packet Length or Total Bytes to Transfer bytes, whichever is less) (e.g. a packet babble). This results in the host controller setting the Halted bit to a one.

With the exception of a NAK response (when RL field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high speed endpoint, the queue head information written back includes minimally the following fields:

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low or fullspeed device the queue head information written back also includes the fields:

- Cprogmask, FrameTag and Sbytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed via queue heads (control, bulk and interrupt). The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USBSTS register to a one. To halt the queue head, the Active bit is set to a zero and the Halted bit is set to a one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller will not advance the transfer state on a transaction that results in a Halt condition (e.g. no updates necessary for Total Bytes to Transfer, C_Page, Current Offset, and dt). The host controller must update CErr as appropriate. When a queue head is halted, the USB Error Interrupt bit in the USBSTS register is set to a one. If the USB Error Interrupt Enable bit in the USBCINTR register is set to a one, a hardware interrupt is generated at the next interrupt threshold.

Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a highspeed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule. This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the Mult feature that is used in the Periodic schedule. Whereas the Mult feature is a characteristic that is tunable for each endpoint; parkmode is a policy that is applied to all highspeed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Section Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the Asynchronous Schedule Park Capability bit in the HCCPARAMS register to a one. This information keys system software that the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register are modifiable. System software enables the feature by writing a one to the Asynchronous Schedule Park Mode Enable bit. When parkmode is not enabled (e.g. Asynchronous Schedule Park Mode Enable bit in the USBCMD register is a zero), the host controller must not execute more than one bus transaction per highspeed queue head, per traversal of the asynchronous schedule. When parkmode is enabled, the host controller must not apply the feature to a queue head whose EPS field indicates a Low/Fullspeed device (i.e. only one bus transaction is allowed from each Low/Fullspeed queue head per traversal of the asynchronous schedule). Parkmode may only be applied to queue heads in the Asynchronous schedule whose EPS field indicates that it is a highspeed device. The host controller must apply park mode to queue heads whose EPS field indicates a highspeed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a highspeed queue head is determined by

the value in the Asynchronous Schedule Park Mode Count field in the USBCMD register. Software must not set Asynchronous Schedule Park Mode Enable bit to a one and also set Asynchronous Schedule Park Mode Count field to a zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing parkmode. This feature does not affect how the host controller handles the bus transaction as defined in Section Execute Transaction. It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the Mult field for highbandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal downcounter PMCount from Asynchronous Schedule Park Mode Count when Asynchronous Schedule Park Mode Enable bit is a one, and a highspeed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, PMCount is decremented. The host controller may continue to execute bus transactions from the current queue head until PM Count goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flowcontrol or STALL handshake. Table 35 summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

| PID | Endpoint Response | Transfer State | after Transaction Bytes to Transfer | Action |
|------|---|------------------------------|-------------------------------------|---|
| | | PMCount | | |
| IN | DATA[0,1] w/Maximum Packet sized data | Not zero Not zero Zero | Not zero Zero Don't care | Allowed to perform another bus transaction.1, 2 Retire qTD and move to next QH Move to next QH. |
| | DATA[0,1] w/short packet | Don't care | Don't care | Retire qTD and move to next QH. |
| | NAK | Don't care | Don't care | Move to next QH. |
| | STALL, XactErr | Don't care | Don't care | Move to next QH. |
| | | | | |
| OUT | ACK | Not zero Not zero Zero | Not zero Zero Don't care | Allowed to perform another bus transaction. 2 Retire qTD and move to next QH Move to next QH. |
| | NYET, NAK | Don't care | Don't care | Move to next QH. |
| | STALL, XactErr | Don't care | Don't care | Move to next QH. |
| | | | | |
| | | | | |
| PING | ACK | Not Zero | Not Zero | Allowed to perform another bus transaction. 2 |
| | NAK | Don't care | Don't care | Move to next QH |
| | STALL, XactErr | Don't care | Don't care | Move to next QH |

511 . Actions for Park Mode, based on Endpoint Response and Residual Transfer State

Write Back qTD

This state is entered from the Execute Transaction state when the Active bit is set to a zero. The source data for the writeback is the transfer results area of the queue head overlay area (see). The host controller uses the Current qTD Pointer field as the target address for the qTD. The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back

to the source qTD include Total Bytes to Transfer, Cerr, and Status. The duration of this state depends on when the qTD writeback has been committed.

Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the Active bit is a one on exit from the Execute Transaction state, or
- When the host controller exits the Write Back qTD state, or
- If the Advance Queue state fails to advance the queue because the target qTD is not active, or
- If the Halted bit is a one on exit from the Fetch QH state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be virtually contiguous. This means: if the buffer spans more than one physical page, it must obey the following rules (Figure 66 illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4K chunk beyond the first page, each buffer portion matches to a full 4K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16Kbyte buffer with an starting buffer alignment. The host controller uses the field C_Page field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous. The host controller must detect when the current transaction will span a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing C_Page and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the Bytes to Transfer field.

For the first transaction on the qTD (assuming a 512byte transaction), the host controller uses the first buffer pointer (page 0 because C_Page is set to zero) and concatenates the Current Offset field. The 512 bytes are moved during the transaction, the Current Offset and Total Bytes to Transfer are adjusted by 512 and written back to the queue head working area. During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller will increment C_Page (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and Current Offset has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (i.e. C_Page) when necessary. There are three conditions for how the host controller handles C_Page.

- The current transaction does not span a page boundary. The value of C_Page is not adjusted by the host controller.
- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (i.e. the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment C_Page before writing back status for the transaction.

Note that the only valid adjustment the host controller may make to C_Page is to increment by one.

Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate. System software sets a bit in a queue head's S Mask to indicate which microframe within a 1 millisecond period a transaction should be executed for the queue head. Software must ensure that all queue heads in the periodic schedule have SMask set to a nonzero value. An Smask with a zero value in the context of the periodic schedule yields undefined results. If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and SMask values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible.

Managing Transfer Complete Interrupts from Queue Heads

The host controller will set an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an Interrupt on Complete (IOC) bit set to a one, or whenever a transfer (qTD) completes with a short packet. If system software needs multiple qTDs to complete a client request (i.e. like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be reused in a timely manner.

Ping Control

USB 2.0 defines an addition to the protocol for highspeed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints. Ping is not allowed for a splittransaction stream. This extension to the protocol eliminates the bad sideeffects of Naking OUT endpoints. The Status field has a Ping State bit, which the host controller uses to determine the next actual PID it will use in the next transaction to the endpoint (see Table 19).

The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- EPS field equals HighSpeed and
- PIDCode field equals OUT

Table 37 illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

| Current | Event | Device | | Next |
|---------|-------|---------|--|---------|
| | | Host | | |
| Do Ping | PING | Nak | | Do Ping |
| Do Ping | PING | Ack | | Do OUT |
| Do Ping | PING | XactErr | | Do Ping |
| Do Ping | PING | Stall | | N/C2 Do |
| OUT | OUT | Nak | | Do Ping |
| Do OUT | OUT | Nyet | | Do Ping |
| Do OUT | OUT | Ack | | Do OUT |
| Do OUT | OUT | XactErr | | Do Ping |
| Do OUT | OUT | Stall | | N/C2 |

512 . Ping Control State Transition Table

Value Meaning

- 0B Do OUT The host controller will use an OUT PID during the next bus transaction to this endpoint.
1B Do Ping The host controller will use a PING PID during the next bus transaction to this endpoint.

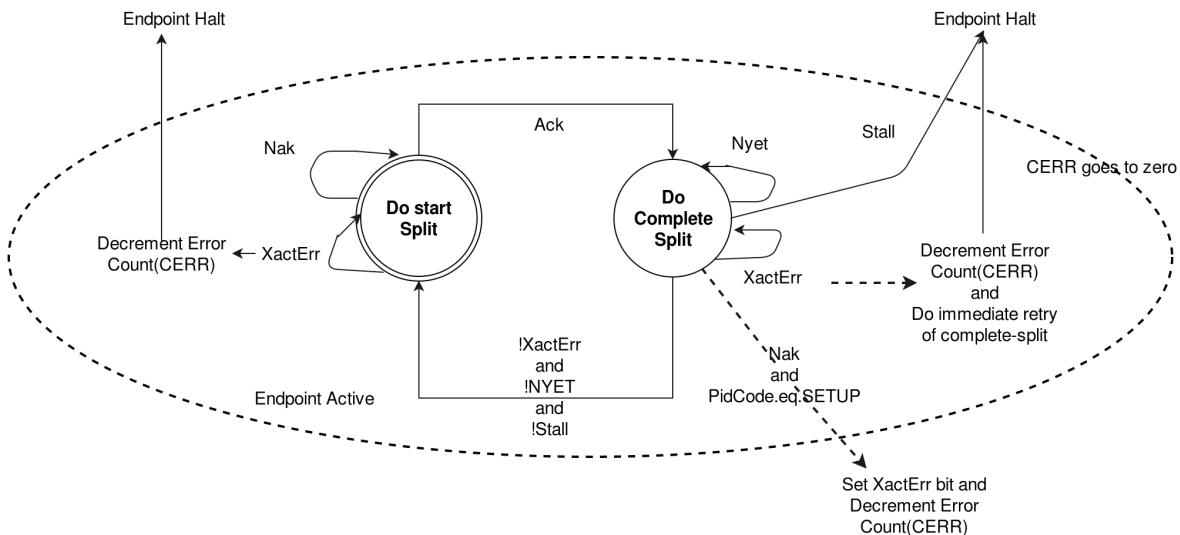
The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be imprecise on the initialization of the ping protocol (i.e. start in Do OUT when we don't know whether there is space on the device or not). The host controller manages the Ping State bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the Ping State bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the Ping State bit is preserved.

Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs. This section describes how the host controller uses the interface data structures to manage data streams with full and lowspeed devices, connected below USB 2.0 hub, utilizing the split transaction protocol. Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full and Lowspeed devices are enumerated identically as highspeed devices, but the transactions to the Full and Lowspeed endpoints use the split transaction protocol on the highspeed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full or Lowspeed transaction. The highspeed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full or Lowspeed device is attached. The EHCI interface uses dedicated data structures for managing fullspeed isochronous data streams (see Section Split Transaction Isochronous Transfer Descriptor (sITD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Sections Queue Head). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low/Fullspeed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an EPS field indicating a fullor lowspeed device indicates to the host controller that it must use split transactions to stream data for this queue head. All fullspeed bulk and full, lowspeed control are managed via queue heads in the asynchronous schedule. Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/ lowspeed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (SplitXState) to DoStartSplit. Finally, if the endpoint is a control endpoint, then system software must set the Control Transfer Type (C) bit in the queue head to a one. If this is not a control transfer type endpoint, the C bit must be initialized by software to be a zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the C bit is a zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the C bit is a one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.



Host Controller Asynchronous Schedule SplitTransaction State Machine

Asynchronous Do Start Split

This is the state which software must initialize a full or lowspeed asynchronous queue head. This state is entered from the Do Complete Split state only after a completesplit transaction receives a valid response from the transaction translator that is not a Nyet handshake. For queue heads in this state, the host controller will execute a startsplit transaction to the appropriate transaction translator. If the bus transaction completes without an error and PidCode indicates an IN or OUT transaction, then the host controller will reload the error counter (CErr). If it is a successful bus transaction and the PidCode indicates a SETUP, the host controller will not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule. If the host controller times out the transaction (no response, or bad response) the host controller decrements Cerr and proceeds to the next queue head in the asynchronous schedule.

Asynchronous Do Complete Split

This state is entered from the Do Start Split state only after a startsplit transaction receives an Ack handshake from the transaction translator. For queue heads in this state, the host controller will execute a completesplit transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's PidCode indicates an IN or OUT, the host controller will reload the error counter (CErr). When a Nyet handshake is received for a completesplit bus transaction where the queue head's PidCode indicates a SETUP, the host controller must not adjust the value of CErr.

Independent of PIDCode, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, etc. The error counter (Cerr) is decremented by one and the complete split transaction is immediately retried (if possible). If there is not enough time in the micro frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another startsplit (for some other endpoint) is sent to the transaction translator before the completesplit is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next microframe, the first transaction from the schedule will be the retry for this endpoint. If Cerr went to zero, the host controller must halt the queue.
- NAK. The target endpoint Nak'd the full or lowspeed transaction. The state of the transfer is not advanced and the state is exited. If the PidCode is a SETUP, then the Nak response is a protocol error. The XactErr status bit is set to a one and the CErr field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the halt bit in the status byte, retires the qTD but does not attempt to advance the queue.

If the PidCode indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is good, the host controller will advance the state of the transfer, e.g. move the data pointer by the number of bytes received, decrement BytesToTransfer field by the number of bytes received, and toggle the dt bit. The host controller will then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the PidCode indicates an OUT/SETUP, then any of following responses are expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The Current Offset field is incremented by Maximum Packet Length or Bytes to Transfer, whichever is less. The field Bytes To Transfer is decremented by the same amount and the data toggle bit (dt) is toggled. The host controller will then exit this state. Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section 5.10).

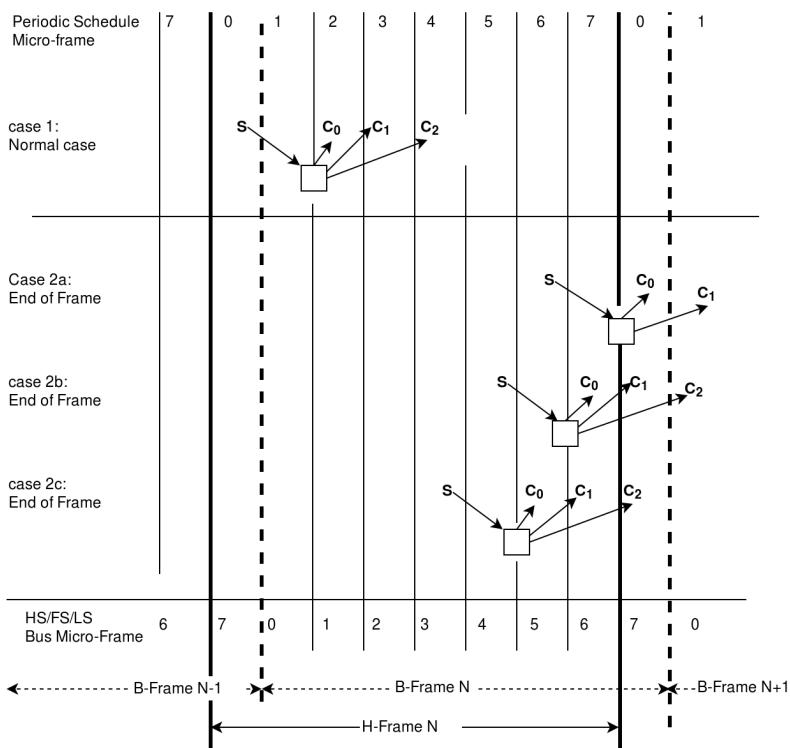
Split Transaction Interrupt

Splittransaction InterruptIN/OUT endpoints are managed via the same data structures used for highspeed interrupt endpoints. They both coexist in the periodic schedule. Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The splittransaction protocol is managed completely within this defined functional transfer framework. For example, for a highspeed endpoint, the host controller will visit a queue head, execute a highspeed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low and fullspeed endpoints, the details of the execution phase are different (i.e. takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, etc. occurs as defined in Section 5.10, but only occurs on the completion of a split transaction.

Split Transaction Scheduling Mechanisms for Interrupt

Full and lowspeed Interrupt queue heads have an EPS field indicating full or lowspeed and have a non zero Smask field. The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low and fullspeed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which microframes the startsplits and completesplits for each endpoint will occur. The characteristics of the transaction translator are such that the highspeed transaction protocol must execute during explicit microframes, or the data or response information in the pipeline is lost. Figure 68 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and X labels indicate microframes where software can schedule startsplits and complete splits (respectively).

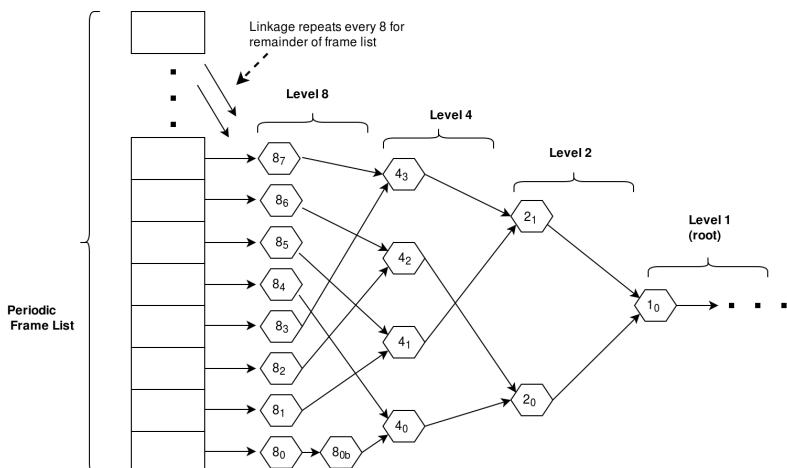


Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (H Frame in this case).

- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many completesplits must be scheduled to account for earliest to latest execution on the full/lowspeed link. The completesplits may span the HFrame boundary when the startsplit is in microframe 4 or later. When this occurs, the HFrame to BFrame alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs. Figure 69 illustrates the general layout of the periodic schedule.



General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2Npoll rate. Software can efficiently manage periodic bandwidth on the USB by spreading interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation. When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 80b where such an endpoint. Without additional support on the interface, to get 80b reachable at the correct time, software would have to link 81 to 80b. It would then have to move 41 and everything linked after into the same path as 40. This upsets the integrity of the binary tree and disallows the use of the spreading technique. FSTN data structures are used to preserve the integrity of the binarytree structure and enable the use of the spreading technique. Section Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the splittransaction protocol.

- **SplitXState.** This is a single bit residing in the Status field of a queue head (see Table 19). This bit is used to track the current state of the split transaction.
- **Frame Smask.** This is a bitfield wherein system software sets a bit corresponding to the microframe (within an H-Frame) that the host controller should execute a startsplit transaction. This is always qualified by the value of the SplitXState bit in the Status field of the queue head. For example, referring to Figure 68, case one, the S-mask would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the SplitXState indicates Do_Start, and the current microframe as indicated by FRINDEX[2:0] is 0, then execute a startsplit transaction.
- **Frame Cmask.** This is a bitfield where system software sets one or more bits corresponding to the micro frames (within an HFrame) that the host controller should execute completesplit transactions. The interpretation of this field is always qualified by the value of the SplitXState bit in the Status field of the queue head. For example, referring to Figure 68, case one, the Cmask would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the SplitXState indicates Do_Complete, and the current microframe as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a completesplit transaction. It is software's responsibility to ensure that the translation between HFrames and BFrames is correctly performed when setting bits in Smask and Cmask

Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Fullspeed interrupt queue heads that need to be reached from consecutive frame list locations (i.e. boundary cases 2a through 2c). An FSTN is essentially a back pointer, similar in intent to the back pointer field in the siTD data structure (see Section 4.4.5). This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

There are four components to the use of FSTNs:

1. FSTN data structure, defined in Section Periodic Frame Span Traversal Node (FSTN).
2. A Save Place indicator. This is always an FSTN with its Back Path Link Pointer. Tbit set to zero.
3. A Restore indicator. This is always an FSTN with its Back Path Link Pointer. Tbit set to a one.
4. Host controller FSTN traversal rules.

Host Controller Operational Model for FSTNs

When the host controller encounters an FSTN during microframes 2 through 7 it simply follows the node's Normal Path Link Pointer to access the next schedule data structure. Note that the FSTN's Normal Path Link Pointer.Tbit may set to a one, which the host controller must interpret as the end of periodic list mark. When the host controller encounters a SavePlace FSTN in microframes 0 or 1, it will save the value of the Normal Path Link Pointer and set an internal flag indicating that it is executing in Recovery Path mode. Recovery Path mode modifies the host controller's rules for how it traverses the schedule and limits which data structures will be considered for execution of bus transactions. The host controller continues executing in Recovery Path mode until it encounters a Restore FSTN or it determines that it has reached the end of the microframe (see details in the list below).

The rules for schedule traversal and limited execution while in Recovery Path mode are:

- Always follow the Normal Path Link Pointer when it encounters an FSTN that is a SavePlace indicator. The host controller must not recursively follow SavePlace FSTNs. Therefore, while executing in Recovery Path mode, it must never follow an FSTN's Back Path Link Pointer.
- Do not process an siTD or, iTD data structure. Simply follow its Next Link Pointer.
- Do not process a QH (Queue Head) whose EPS field indicates a highspeed device. Simply follow its Horizontal Link Pointer.
- When a QH's EPS field indicates a Full/Lowspeed device, the host controller will only consider it for execution if its SplitXState is DoComplete (note: this applies whether the PID Code indicates an IN or an OUT). See Sections Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. Note that the host controller must not execute a Startsplits transaction while executing in Recovery Path mode. See Section Periodic Interrupt Do Complete Split for special handling when in Recovery Path mode.
- Stop traversing the recovery path when it encounters an FSTN that is a Restore indicator. The host controller unconditionally uses the saved value of the SavePlace FSTN's Normal Path Link Pointer when returning to the normal path traversal. The host controller must clear the context of executing a Recovery Path when it restores schedule traversal to the SavePlace FSTN's Normal Path Link Pointer.

If the host controller determines that there is not enough time left in the microframe to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive microframe, the host controller starts traversal at the frame list.

In frame N (microframes 07), for this example, the host controller will traverse all of the schedule data structures utilizing the Normal Path Pointers in any FSTNs it encounters. This is because the host controller has not yet encountered a SavePlace FSTN so it is not executing in Recovery Path mode. When it encounters the Restore FSTN, (RestoreN), during microframes 0 and 1, it uses RestoreN.Normal Path Link Pointer to traverse to the next data structure (i.e. normal schedule traversal). This is because the host controller must use a Restore FSTN's Normal Path Link Pointer when not executing in a RecoveryPath mode. The nodes traversed during frame N include: {82.0,82.1, 82.2, 82.3, 42, 20, RestoreN, 10 ...}. In frame N+1 (microframes 0 and 1), when the host controller encounters SavePath FSTN (SaveN), it observes that SaveN.Back Path Link Pointer.Tbit is zero (definition of a Save-Path indicator). The host controller saves the value of SaveN.Normal Path Link Pointer and follows SaveN.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in Recovery Path mode (the

recovery path is annotated in Figure 70 with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (RestoreN). Restore N.Back Path Link Pointer.Tbit is set to a one (definition of a Restore indicator), so the host controller exits Recovery Path mode by clearing the internal Recovery Path mode flag and commences (restores) schedule traversal using the saved value of the SavePlace FSTN's Normal Path Link Pointer (e.g. SaveN.Normal Path Link Pointer). The nodes traversed during these microframes include: {83.0, 83.1, 83.2, SaveA, 82.2, 82.3, 42, 20, RestoreN, 43, 21, RestoreN, 10...}. The nodes on the recoverypath are highlighted.

In frame N+1 (microframes 27), when the host controller encounters SavePath FSTN SaveN, it will unconditionally follow SaveN.Normal Path Link Pointer. The nodes traversed during these microframes include: {83.0, 83.1, 83.2, SaveA, 43, 21, RestoreN, 10 ...}.

Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse. When using FSTNs, system software must adhere to the following rules:

- Each SavePlace indicator requires a matching Restore indicator. The SavePlace indicator is an FSTN with a valid Back Path Link Pointer and Tbit equal to zero. Note that Back Path Link Pointer.Typ field must be set to indicate the referenced data structure is a queue head. The Restore indicator is an FSTN with its Back Path Link Pointer.Tbit set to a one. A Restore FSTN may be matched to one or more SavePlace FSTNs. For example, if the schedule includes a pollrate 1 level, then system software only needs to place a Restore FSTN at the beginning of this list in order to match all possible SavePlace FSTNs.
- If the schedule does not have elements linked at a pollrate level of one, and one or more SavePlace FSTNs are used, then System Software must ensure the Restore FSTN's Normal Path Link Pointer's Tbit is set to a one, as this will be used to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a Restore FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the pollrate one list. System software must ensure that Recovery Path mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A SavePlace FSTN's Back Path Link Pointer must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list location. In other words, if the SavePlace FSTN is reachable from frame list offset N, then the FSTN's Back Path Link Pointer must reference a queue head that is reachable from frame list offset N1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one SavePlace FSTN reachable in any single frame. Note there will be times when two (or more, depending on the implementation) could exist as full/lowspeed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth rebalance causes system software to move the SavePlace FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost. For interruptIN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline. When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A dataloss condition exists whenever a startsplit is issued, accepted and successfully executed by the USB 2.0 Hub, but the completesplits get unrecoverable errors on the highspeed link, or the completesplits do not occur at the correct times. One reason completesplits might not occur at the right time would be due to hostinduced system holdoffs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interruptOUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- Cprogmask. This is an eightbit bitvector where the host controller keeps track of which completesplits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the completesplits need to be executed inorder. The host controller needs to detect when the completesplits have not been executed in order. This can

only occur due to system holdoffs where the host controller cannot get to the memorybased schedule. Cprogmask is a simple bitvector that the host controller sets one of the Cprogmask bits for each complete split executed. The bit position is determined by the microframe number in which the completesplit was executed. The host controller always checks Cprogmask before executing a completesplit transaction. If the previous completesplits have not been executed then it means one (or more) have been skipped and data has potentially been lost.

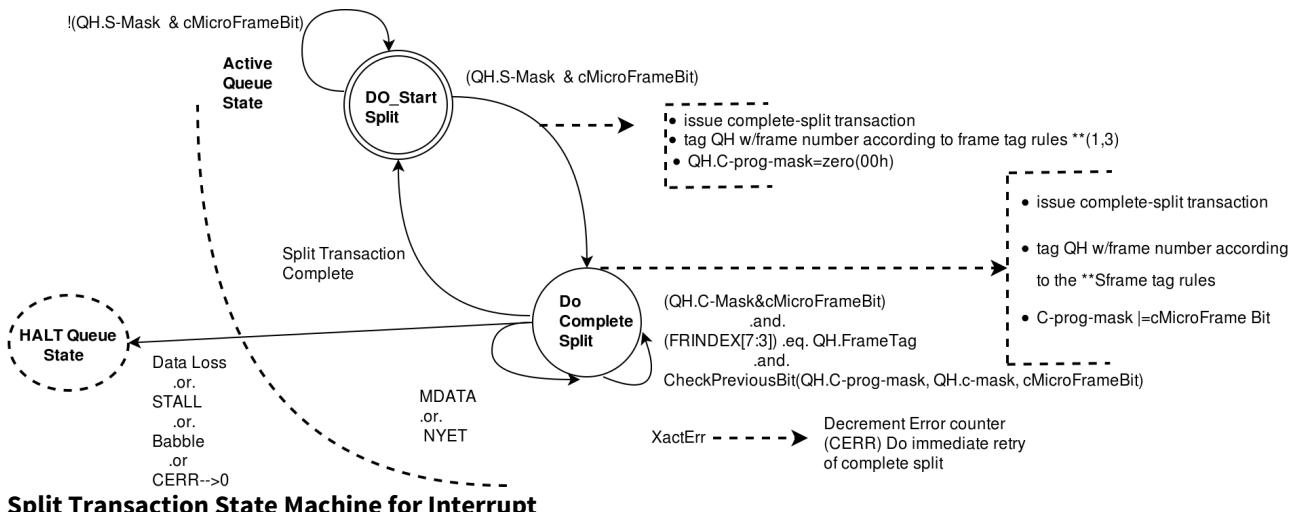
- FrameTag. This field is used by the host controller during the completesplit portion of the split transaction to tag the queue head with the frame number (HFrame number) when the next complete split must be executed.
- Sbytes. This field can be used to store the number of data payload bytes sent during the startsplit (if the transaction was an OUT). The Sbytes field must be used to accumulate the data payload bytes received during the completesplits (for an IN).

Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to microframe are in the context of a microframe within an H Frame. As with asynchronous Full and Lowspeed endpoints, a splittransaction state machine is used to manage the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- cMicroFrameBit. This is a singlebit encoding of the current microframe number. It is an eightbit value calculated by the host controller at the beginning of every microframe. It is calculated from the three least significant bits of the FRINDEX register (i.e. cMicroFrameBit = (1 shiftedleft(FRINDEX[2:0]))). The cMicroFrameBit has at most one bit asserted, which always corresponds to the current microframe number. For example, if the current microframe is 0, then cMicroFrameBit will equal 00000001b.

The variable cMicroFrameBit is used to compare against the Smask and Cmask fields to determine whether the queue head is marked for a start or completesplt transaction for the current microframe. Figure 71 illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single startsplit transaction, which occurs when the SplitXState is at Do_Start and the single bit in cMicroFrameBit has a corresponding bit active in QH.Smask. The transaction translator does not acknowledge the receipt of the periodic startsplit, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there will be more than one complete split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set to a one in the QH.Cmask field. The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the threestrikesrule (e.g. after the host tries the same transaction three times, and each encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).



Periodic Interrupt Do Start Split

This is the state software must initialize a full or lowspeed interrupt queue head StartXState bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a completesplit transaction with one of the following:

- NAK. A NAK response is a propagation of the full or lowspeed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full or lowspeed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low/fullspeed link below the transaction translator had a failure (e.g. timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last completesplit and the transaction translator responded with a NYET handshake. This means that the startsplit was not correctly received by the transaction translator, so it never executed a transaction to the full or lowspeed endpoint, see Section Periodic Interrupt Do Complete Split for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it performs the following test to determine whether to execute a startsplit.

- QH.Smask is bitwise anded with cMicroFrameBit. If the result is nonzero, then the host controller will issue a start-split transaction. If the PIDCode field indicates an IN transaction, the host controller must zeroout the QH.Sbytes field. After the splittransaction has been executed, the host controller sets up state in the queue head to track the progress of the completesplit phase of the split transaction. Specifically, it records the expected frame number into QH.FrameTag field (see Section Managing QH.FrameTag Field), set Cprogmask to zero (00h), and exits this state. Note that the host controller must not adjust the value of CErr as a result of completion of a startsplit transaction.

Periodic Interrupt Do Complete Split

This state is entered unconditionally from the Do Start Split state after a startsplit transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it checks to determine whether a completesplit transaction should be executed now. There are four tests to determine whether a completesplit transaction should be executed.

- Test A. cMicroFrameBit is bitwise anded with QH.Cmask field. A nonzero result indicates that software scheduled a completesplit for this endpoint, during this microframe.
- Test B. QH.FrameTag is compared with the current contents of FRINDEX[7:3]. An equal indicates a match.
- Test C. The completesplit progress bit vector is checked to determine whether the previous bit is set, indicating that the previous completesplit was appropriately executed.
- Test D. Check to see if a startsplit should be executed in this microframe. Note this is the same test performed in the Do Start Split state (see Section Periodic Interrupt Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a Recovery Path mode, it means a startsplit should occur in this microframe. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a completesplit transaction. When the host controller commits to executing the completesplit transaction, it updates QH.Cprogmask by bitORing with cMicroFrameBit. On completion of the completesplit transaction, the host controller records the result of the transaction in the queue head and sets QH.FrameTag to the expected HFrame number (see Section Managing QH.FrameTag Field). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the completesplit transaction. The following responses have the effects (note that any responses that result in decrementing of the CErr will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last completesplit for this split transaction. Last is defined in this context as the condition where all of the scheduled completesplits have been executed. If it is the last completesplit (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the completesplits with NYETs, meaning that the startsplit issued by the host controller was

not received. The startsplit should be retried at the next poll period. The test for whether this is the Last complete split can be performed by XOR QH.Cmask with QH.Cprog mask. If the result is all zeros then all completesplits have been executed. When this condition occurs, the XactErr status bit is set to a one and the CErr field is decremented.

- NYET (and not Last). See above description for testing for Last. The completesplit transaction received a NYET response from the transaction translator. Do not update any transfer state (except for Cprogmask and FrameTag) and stay in this state. The host controller must not adjust CErr on this response.
- Transaction Error (XactErr). Timeout, data CRC failure, etc. The CErr field is decremented and the XactErr bit in the Status field is set to a one. The complete split transaction is immediately retried (if Cerr is nonzero). If there is not enough time in the microframe to complete the retry and the endpoint is an IN, or CErr is decremented to a zero from a one, the queue is halted. If there is not enough time in the microframe to complete the retry and the endpoint is an OUT and CErr is not zero, then this state is exited (i.e. return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full and lowspeed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The Current Offset field is incremented by Maximum Packet Length or Bytes to Transfer, whichever is less. The field Bytes To Transfer is decremented by the same amount. And the data toggle bit (dt) is toggled. The host controller will then exit this state for this queue head. The host controller must reload CErr with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section 5.10).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in QH.S bytes. The host controller must not adjust CErr on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in QH.Sbytes. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head. Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers via Queue Heads). If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full or lowspeed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload CErr with maximum value on this response.
- ERR. There was an error during the full or lowspeed transaction. The ERR status bit is set to a one, Cerr is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits: Active bit is set to zero and the Halted bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. Table 38 lists the possible combinations and the appropriate action.

| Condition | Action | Description |
|---------------------------------|---|---|
| not(A) not(D) A not(C) | Ignore QHD If PIDCode = IN Halt QHD If PIDCode = OUT Retry startsplit | Neither a start nor completesplit is scheduled for the current micro-frame. Host controller should continue walking the schedule. Progress bit check failed. This means a completesplit has been missed. There is the possibility of lost data. If PIDCode is an IN, then the Queue head must be halted. If PIDCode is an OUT, then the transfer state is not advanced and the state exited (e.g. startsplit is retried). This is a hostinduced error and does not effect CERR. In either case, set the Missed Microframe bit in the status field to a one. |
| A not(B) C | If PIDCode = IN Halt QHD If PIDCode = OUT Retry startsplit | QH.FrameTag test failed. This means that exactly one or more HFrames have been skipped. This means completesplits and have missed. There is the possibility of lost data. If PIDCode is an IN, then the Queue head must be halted. If PIDCode is an OUT, then the transfer state is not advanced and the state exited (e.g. startsplit is retried). This is a hostinduced error and does not effect CERR. In either case, set the Missed Microframe bit in the status field to a one. |

| Condition | Action | Description |
|-----------|-----------------------|--|
| A | Execute completesplit | This is the nonerror case where the host controller executes a complete-split transaction. |
| B | If PIDCode = IN | This is a degenerate case where the startsplit was issued, but all of the completesplits were skipped and all possible intervening |
| C | Halt QHD | opportunities to detect the missed data failed to fire. If PIDCode is an IN, then |
| not(D) | If PIDCode = OUT | the Queue head must be halted. |
| D | Retry startsplit | If PIDCode is an OUT, then the transfer state is not advanced and the state exited (e.g. startsplit is retried). This is a host induced error and does not effect CERR. In either case, set the Missed Microframe bit in the status field to a one. Note: When executing in the context of a Recovery Path mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the normal processing mode. Regardless, the host controller must not execute a startsplit in the context of a executing in a Recovery Path mode. |

513 . Interrupt IN/OUT Do Complete Split State Execution Criteria

Managing QH.FrameTag Field

The QH.FrameTag field in a queue head is completely managed by the host controller. The rules for setting QH.FrameTag are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of FRINDEX[2:0] is 6 QH.FrameTag is set to FRINDEX[7:3] + 1. This accommodates split transactions whose startsplit and complete splits are in different HFrames (case 2a, see Figure 68).
- Rule 2: If the current value of FRINDEX[2:0] is 7, QH.FrameTag is set to FRINDEX[7:3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c (Figure 68).
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of FRINDEX[2:0] is not 6, or currently in Do Complete Split and the current value of (FRINDEX[2:0]) is not 7, FrameTag is set to FRINDEX[7:3]. This accommodates all other cases (Figure 68).

Rebalancing the Periodic Schedule

System software must occasionally adjust a periodic queue head's Smask and Cmask fields during operation. This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (i.e. new Smask and Cmask values). It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the InactivateonnextTransaction (I) bit to a one to signal the host controller that it intends to update the Smask

and Cmask on this queue head. System software will then wait for the host controller to observe the Ibit is a one and transition the Active bit to a zero. The rules for how and when the host controller sets the Active bit to zero are enumerated below:

- If the Active bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the I-bit is a one.
- If the Active bit is a one and the SplitXState is DoStart (regardless of the value of Smask), the host controller will simply set Active bit to a zero. The host controller is not required to write the transfer state back to the current qTD. Note that if the Smask indicates that a startsplit is scheduled for the current microframe, the host controller must not issue the startsplit bus transaction. It must set the Active bit to zero.

System software must save transfer state before setting the Ibit to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the Ibit was set to a one and the host controller executed its final bustransaction and set Active to a zero.

After system software has updated the Smask and Cmask, it must then reactivate the queue head. Since the Active bit and the Ibit cannot be updated with the same write, system software needs to use the following algorithm to coherently reactivate a queue head that has been stopped via the Ibit.

1. Set the Halted bit to a one, then
2. Set the Ibit to a zero, then
3. Set the Active bit to a one and the Halted bit to a zero in the same write.

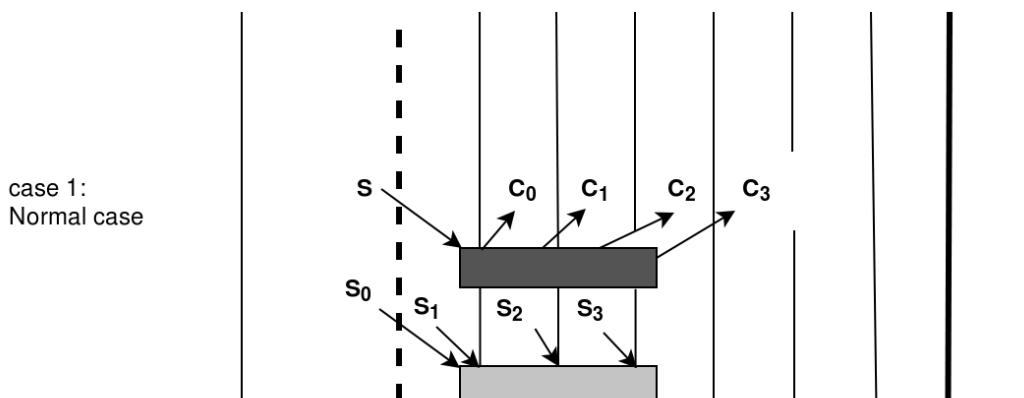
Setting the Halted bit to a one inhibits the host controller from attempting to advance the queue between the time the Ibit goes to a zero and the Active bit goes to a one.

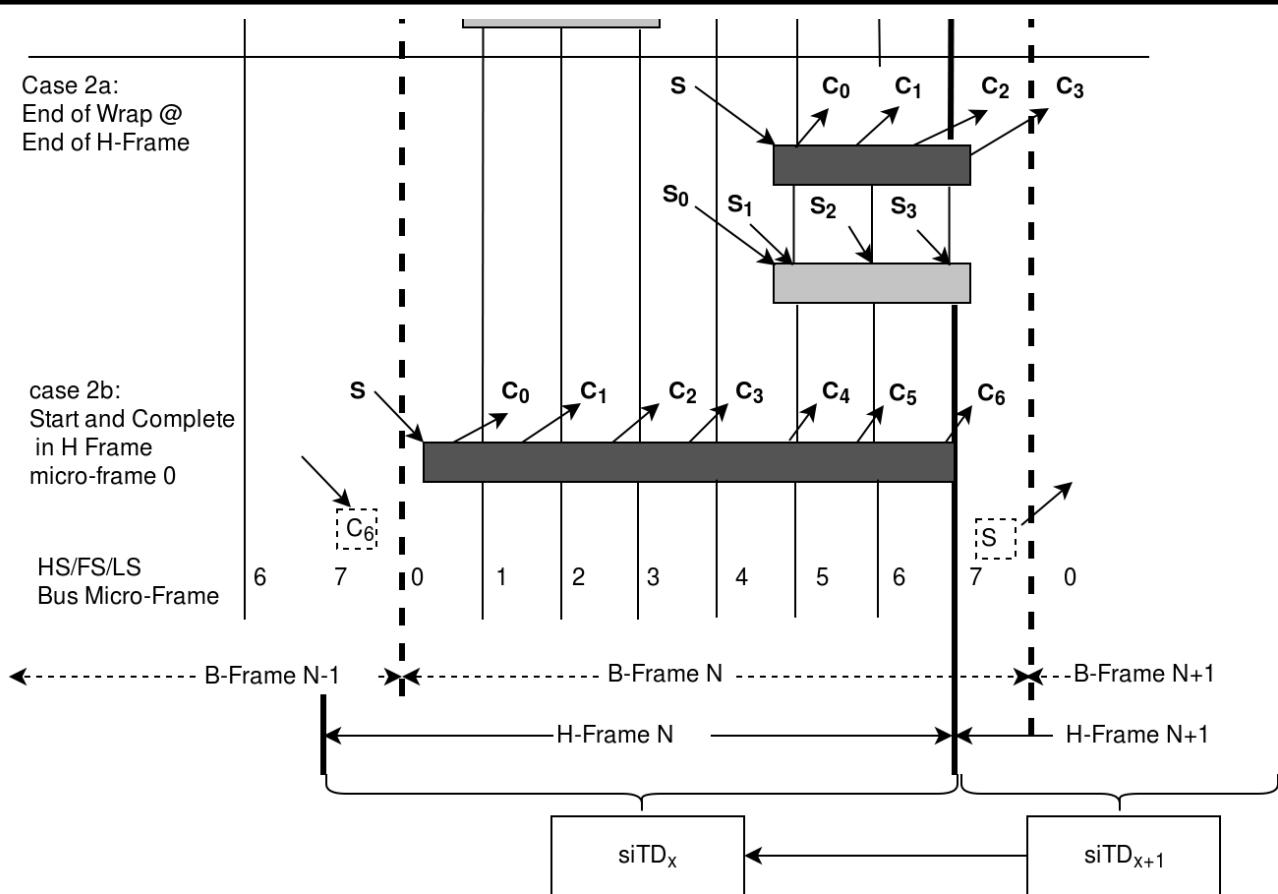
Split Transaction Isochronous

Fulldspeed isochronous transfers are managed using the splittransaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous splittransactions. This data structure uses the scheduling model of isochronous TDs (iTID, Section Isochronous (HighSpeed) Transfer Descriptor (iTID)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

Split Transaction Scheduling Mechanisms for Isochronous

Fulldspeed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full and lowspeed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which microframes the startsplits and completesplits for each fulldspeed isochronous endpoint occur. The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. Figure 72 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The X and X labels indicate microframes where software can schedule start and completesplits (respectively). The HFrame boundaries are marked with a large, solid bold vertical line. The BFrame boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the H Frame.





Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only startsplits, and no completesplits. When the endpoint is an isochronous IN, there is at most one startsplit and one to N completesplits. The scheduling boundary cases are:

- **Case 1:** The entire split transaction is completely bounded by an HFrame. For example: the startsplits and completesplits are all scheduled to occur in the same HFrame.
- **Case 2a:** This boundary case is where one or more (at most two) completesplits of a split transaction IN are scheduled across an HFrame boundary. This can only occur when the split transaction has the possibility of moving data in BFrame, microframes 6 or 7 (HFrame microframe 7 or 0). When an HFrame boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(e.g. it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).

Although the scheduling of the split transaction may take two data structures, all of the completesplits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below. Software must never schedule fullspeed isochronous OUTs across an HFrame boundary.

- **Case 2b:** This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start split and completesplit for the same endpoint can occur in the same microframe. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full and lowspeed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- **SplitXState.** This is a single bit residing in the Status field of an siTD (see Table 14). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section Split Transaction mExecution State Machine for Isochronous.

- Frame Smask. This is a bitfield wherein system software sets a bit corresponding to the microframe (within an HFrame) that the host controller should execute a startsplit transaction. This is always qualified by the value of the SplitXState bit. For example, referring to the IN example in Figure 72, case one, the Smask would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the SplitXState indicates Do Start Split, and the current microframe as indicated by FRINDEX[2:0] is 0, then execute a startsplit transaction.
- Frame Cmask. This is a bitfield where system software sets one or more bits corresponding to the micro frames (within an HFrame) that the host controller should execute completesplit transactions. The interpretation of this field is always qualified by the value of the SplitXState bit. For example, referring to the IN example in Figure 72, case one, the Cmask would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the SplitXState indicates Do Complete Split, and the current microframe as indicated by FRINDEX[2:0] is 2, 3, 4, or 5, then execute a completesplit transaction.
- Back Pointer. This field in a siTD is used to complete an IN splittransaction using the previous HFrame's siTD. This is only used when the scheduling of the completesplits span an HFrame boundary.

There exists a one-to-one relationship between a highspeed isochronous split transaction (including all start and completesplits) and one fullspeed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one fullspeed isochronous data payload. This means that for any fullspeed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN and OUTs. An siTD's scheduling information usually also maps to one highspeed isochronous split transaction. The exception to this rule is the HFrame boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of highspeed transactions and that description is strictly bounded within a frame boundary. Each case is described below:

- Case 1: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single HFrame.
- Case 2a, 2b: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction. siTDX is used to always issue the startsplit and the first N completesplits. The fullspeed transaction (for these cases) can deliver data on the fullspeed bus segment during microframe 7 of HFrameY+1, or microframe 0 of HFrameY+2. The complete splits are scheduled using siTDX+2 (not shown). The completesplits to extract this data must use the buffer pointer from siTDX+1. The only way for the host controller to reach siTDX+1 from HFrameY+2 is to use siTDX+2's back pointer. The host controller rules for when to use the back pointer are described in Section 5.12.3.3.2.

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous splittransaction is started so that it will complete before the end of the BFrame.
- Software must ensure that for a single fullspeed isochronous endpoint, there is never a startsplit and complete split in HFrame, microframe 1. This is mandated as a rule so that case 2a and case 2b can be discriminated.

According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the startsplit was in microframe 1 of HFrame N and the last completesplit would need to occur in microframe 1 of HFrame N+1. However, it is impossible to discriminate between cases 2a and case2b, which has significant impact on the complexity of the host controller.

Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report perpacket errors observed in the data stream. This includes schedule traversal problems (skipped microframes), timeouts and corrupted data received. In similar kind to interrupt splittransactions, the portions of the split transaction protocol must execute in the microframes they are scheduled. The queue head data structure used to manage full and lowspeed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact microframe in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future microframe).

Splittransaction isochronous OUTs utilize a lowlevel protocol to indicate which portions of the split transaction data have arrived. Control over the lowlevel protocol is exposed in an siTD via the fields Transaction Position (TP) and Transaction Count (Tcount). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one startsplit transaction, each of which require proper annotation. If host hold offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section 5.12.3.3.1 for a description on how these fields are used during a sequence of startsplit transactions.

The fields siTD.TCount and siTD.TP are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a splittransaction isochronous endpoint is established, Smask, TCount, and TP initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For INendpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a completesplit. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

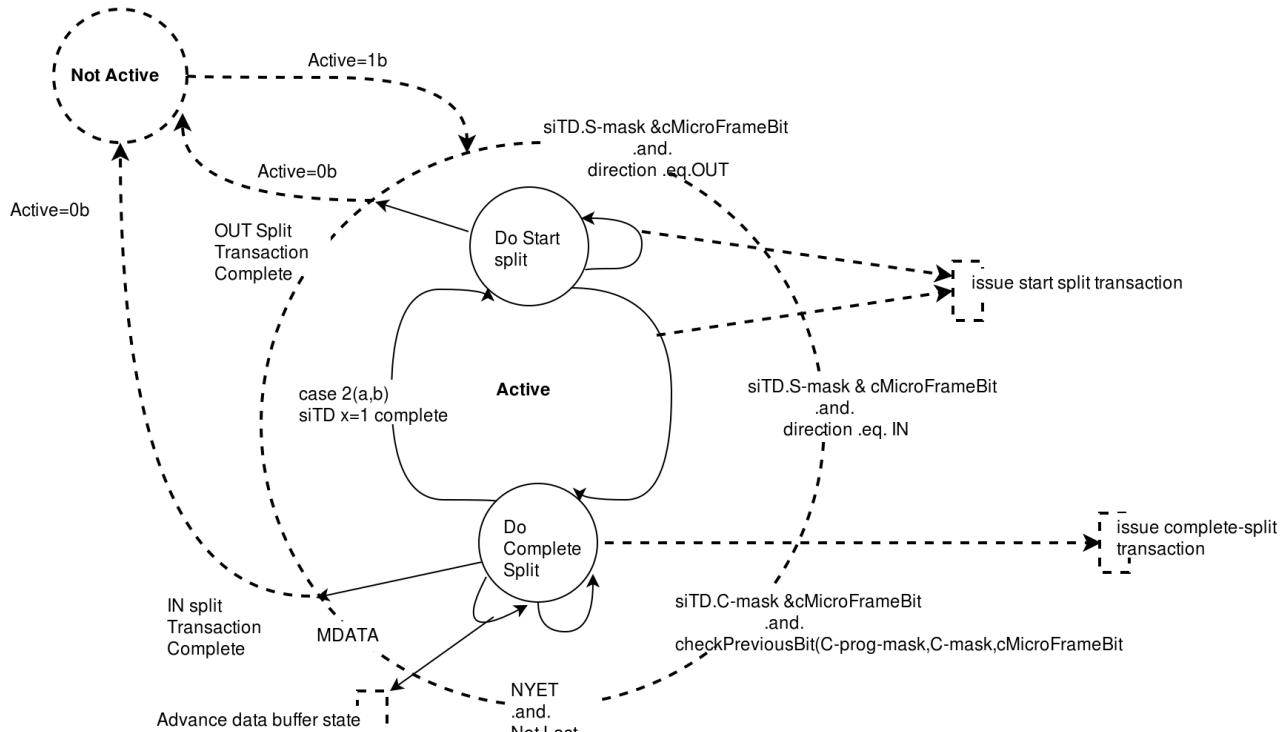
- Cprogmask. This is an eightbit bitvector where the host controller keeps track of which completesplits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the completesplits need to be executed inorder. The host controller needs to detect when the completesplits have not been executed in order. This can only occur due to system holdoffs where the host controller cannot get to the memorybased schedule. Cprogmask is a simple bitvector that the host controller sets a bit for each completesplit executed. The bit position is determined by the microframe (FRINDEX[2:0]) number in which the completesplit was executed. The host controller always checks Cprogmask before executing a completesplit transaction. If the previous completesplits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's Active bit to a one.

If a transaction translator returns with the final data before all of the completesplits have been executed, the state of the transfer is advanced so that the remaining completesplits are not executed. Refer to Section 5.12.3.3.2 for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired basedsolely on the responses from the Transaction Translator to the completesplit transactions. This means, for example, that it is possible for a transaction translator to respond to a completesplit with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD field Total Bytes to Transfer to decrement to zero. This response can occur, before all of the scheduled completesplits have been executed. In other interface, data structures (e.g. highspeed data streams through queue heads), the transition of Total Bytes to Transfer to zero signals the end of the transfer and results in setting of the Active bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next completesplit transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a microframe boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the fullspeed bus pipe stage and give the remaining bytes to the highspeed pipeline stage. At the micro frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next microframe, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the fullspeed pipeline stage). This could cause the siTD to decrement it's Total Bytes to Transfer field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) completesplit transaction in order to extract the results of the fullspeed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences holdoffs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host holdoffs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the Cprogmask is used by the host controller to detect errors. However, if the host experiences a holdoff that causes it to skip all of an siTD, or an siTD expires during a host hold off (e.g. a holdoff occurs and the siTD is no longer reachable by the host controller in order for it to report the holdoff event), then system software must detect that the siTDs have not been processed by the host controller (e.g. state not advanced) and report the appropriate error to the client driver.

Split Transaction Execution State Machine for Isochronous

In the following presentation, all references to microframe are in the context of a microframe within an H Frame. If the Active bit in the Status byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the splittransaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Isochronous Transfers, plus the variable cMicroFrameBit defined in Section 5.12.2.4 to track the progress of an isochronous split transaction. Figure 74 illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the Active bit in the Status field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.



Split Transaction State Machine for Isochronous

Periodic Isochronous -Do Start Split

Isochronous split transaction OUTs use only this state. An siTD for a splittransaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous splittransaction completes. Each time the host controller reaches an active siTD in this state, it checks the siTD.Smask against cMicroFrameBit. If there is a one in the appropriate position, the siTD will execute a startsplit transaction. By definition, the host controller cannot reach an siTD at the wrong time. If the I/O field indicates an IN, then the startsplit transaction includes only the extended token plus the fullspeed token. Software must initialize the siTD.Total Bytes To Transfer field to the number of bytes expected. This is usually the maximum packet size for the fullspeed endpoint. The host controller exits this state when the startsplit transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (i.e. the I/O field indicates an OUT). When the host controller executes a startsplit transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory buffer address for the data payload is constructed by concatenating siTD.Current Offset with the page pointer indicated by the page selector field (siTD.P). A zero in this field selects Page 0 and a 1 selects Page 1. During the startsplit for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the siTD.Pbit from a zero to a one, and begin using the siTD.Page 1 with siTD.Current Offset as the memory address pointer. The field siTD.TP is used to annotate each

startsplit transaction with the indication of which part of the splittransaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in siTD.TP to mark the startsplit with the correct transaction position code. TCount is always initialized to the number of startsplits for the current frame. TP is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 73) is used to determine the initial value of TP. The initial cases are summarized in Table 39.

Table 39 Initial Conditions for OUT siTD's TP and Tcount Fields

| TP | Tcount | next | TP next | Description |
|-------|--------|------|---------|--|
| ALL | 0 | | N/A | Transition from ALL, to done. |
| BEGIN | 1 | | END | Transition from BEGIN to END. Occurs when T count starts at 2. |
| BEGIN | BEGIN | | MID | Transition from BEGIN to MID. Occurs when T count starts at greater than 2. |
| MID | !=1 | | MID | TP stays at MID while Tcount is not equal to 1 (e.g. greater than 1). This case can occur for any of the scheduling boundary cases where the Tcount starts greater than 3. |
| MID | 1 | | END | Transition from MID to END. This case can occur for any of the scheduling boundary cases where the Tcount starts greater than 2. |

514 . Initial Conditions for OUT siTD's TP and Tcount Fields

After each startsplit transaction is complete, the host controller updates TCount and TP appropriately so that the next startsplit is correctly annotated. Table 40 illustrates all of the TP and Tcount transitions, which must be accomplished by the host controller.

| TP | Tcount | next | TP next | Description |
|-------|--------|------|---------|--|
| ALL | 0 | | N/A | Transition from ALL, to done. |
| BEGIN | 1 | | END | Transition from BEGIN to END. Occurs when T count starts at 2. |
| BEGIN | BEGIN | | MID | Transition from BEGIN to MID. Occurs when T count starts at greater than 2. |
| MID | !=1 | | MID | TP stays at MID while Tcount is not equal to 1 (e.g. greater than 1). This case can occur for any of the scheduling boundary cases where the Tcount starts greater than 3. |

| TP | Tcount | next | TP next | Description |
|-----|--------|------|------------|--|
| MID | 1 | END | | Transition from MID to END. This case can occur for any of the scheduling boundary cases where the Tcount starts greater than 2. |

515 . Transaction Position (TP)/Transaction Count (TCount) Transition

The startsplit transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The siTD.Total Bytes To Transfer and the siTD.Current Offset fields are adjusted to reflect the number of bytes transferred.
- The siTD.P (page selector) bit is updated appropriately.
- The siTD.TP and siTD.Tcount fields are updated appropriately as defined in Table 40.

These fields are then written back to the memory based siTD. The Smask is fixed for the life of the current budget. As mentioned above, TP and Tcount are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of Smask, the actual number of startsplit transactions depends on Tcount (or equivalently, Total Bytes to Transfer). The host controller must set the Active bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when TCount decrements to zero as a result of a startsplit bus transaction. Equivalently, the host controller can detect when Total Bytes to Transfer decrements to zero. Either implementation must ensure that if the initial condition is Total Bytes to Transfer equal to zero and Tcount is equal to a one, then the host controller will issue a single startsplit, with a zerolength data payload. Software must ensure that TP, Tcount and Total Bytes to Transfer are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior. If the host experiences holdoffs that cause the host controller to skip startsplit transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the startsplits for the OUT endpoint (i.e. the transaction position annotation will be incorrect as received by the transaction translator).

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the siTD.Cprogmask as it executes each scheduled startsplit. The checkPreviousBit() algorithm defined in Section Periodic Isochronous Do Complete Split can be used prior to executing each startsplit to determine whether startsplits were skipped. The host controller can use this mechanism to detect missed micro frames. It can then set the siTD's Active bit to zero and stop execution of this siTD. This saves on both memory and highspeed bus bandwidth.

Periodic Isochronous Do Complete Split

This state is only used by a splittransaction isochronous IN endpoint. This state is entered unconditionally from the Do Start State after a startsplit transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a completesplit transaction. The individual tests are listed below. The sequence they are applied depends on which microframe the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- Test A. cMicroFrameBit is bitwise anded with siTD.Cmask field. A nonzero result indicates that software scheduled a completesplit for this endpoint, during this microframe. This test is always applied to a newly fetched siTD that is in this state.
- Test B. The siTD.Cprogmask bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Section 5.12.2.4.2). The sequence in which this test is applied depends on the current value of FRINDEX[2:0]. If FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

If Test A is true and FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (seeFigure 72). See Section Periodic Isochronous Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a completesplit transaction using the transfer state of the current siTD. When the host controller commits to executing the completesplit transaction, it updates QH.Cprogmask by bitORing with cMicroFrameBit. The transfer state is advanced based on the completion status of the completesplit transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from siTD.Total Bytes To Transfer,
- Adjust siTD.Current Offset by the number of bytes received,
- Adjust siTD.P (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the siTD.Status field, depending on the results of the transaction.

Note that if the host controller encounters a condition where siTD.Total Bytes To Transfer is zero, and it receives more data, the host controller must not write the additional data to memory. The siTD.Status.Active bit must be set to zero and the siTD.Status.Babble Detected bit must be set to a one. The fields siTD.Total Bytes To Transfer, siTD.Current Offset, and siTD.P (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of siTD.Total Bytes To Transfer) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set siTD.Status.Active to a zero and siTD.Status.Babble Detected to a one when it receives an MDATA or DATA0/1 with a data payload of more than 192 bytes. The

following responses have the noted effects:

- ERR. The fullspeed transaction completed with a timeout or bad CRC and this is a reflection of that error to the host. The host controller sets the ERR bit in the siTD.Status field and sets the Active bit to a zero.
- Transaction Error (XactErr). The completesplit transaction encounters a Timeout, CRC16 failure, etc. The siTD.Status field XactErr field is set to a one and the completesplit transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the microframe occurs, the Active bit is set to zero.
- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the Active bit is set to a zero. If the Bytes To Transfer field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This short packet event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last completesplit for this split transaction. Last was defined in Section Periodic Interrupt Do Complete Split. If it is the last completesplit (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the Active bit is set to a zero. No bits are set in the Status field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled completesplits with NYETs, meaning that the startsplit issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing Cmask with Cprogmask. A zero result indicates that all completesplits have been executed.
- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the fullspeed link, which delayed the completion of the fullspeed transaction, or software set up the Smask and/or Cmasks incorrectly. The host controller must set XactErr bit to a one and the Active bit is set to a zero.
- NYET (and not Last). See above description for testing for Last. The completesplit transaction received a NYET response from the transaction translator. Do not update any transfer state (except for Cprogmask) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the fullspeed transaction data payload spans from microframe X to X+1 and during microframe X, the transaction translator will respond with an MDATA and the data accumulated up to the end of microframe X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the completesplits have been skipped. The host controller sets the Missed MicroFrame status bit and sets the Active bit to a zero.

CompleteSplit for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 72) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. Table 41 enumerates the transaction state fields.

| Buffer State | Status | Execution Progress |
|----------------------------|------------------------------|--------------------|
| Total Bytes To Transfer | All bits in the status field | Cprogmask |
| P (page select) | | |
| Current Offset | | |
| TP (transaction position) | | |
| Tcount (transaction count) | | |

516 . Summary siTD Split Transaction State

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the siTD.Back Pointer field to reference a valid siTD and will have the siTD.Back Pointer.Tbit in the siTD.Back Pointer field set to a zero. Otherwise, software must set the siTD.Back Pointer.Tbit in the siTD.Back Pointer field to a one. The host controller's rules for interpreting when to use the siTD.Back Pointer field are listed below. These rules apply only when the siTD's Active bit is a one and the SplitXState is Do Complete Split.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.Tbit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.Smask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from siTDX1. In order to access siTDX1, the host controller reads onchip the siTD referenced from siTDX.Back Pointer. The host controller must save the entire state from siTDX while processing siTDX1. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of siTD.Back Pointers. If siTDX1 is active (Active bit is a one and SplitXStat is Do Complete Split), then both Test A and Test B are applied as described above. If these criteria to execute a completesplit are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 41) of siTDX1 is appropriately advanced based on the results and written back to memory. If the resultant state of siTDX1's Active bit is a one, then the host controller returns to the context of siTDX, and follows its next pointer to the next schedule item. No updates to siTDX are necessary. If siTDX1 is active (Active bit is a one and SplitXStat is Do Start Split), then the host controller must set Active bit to a zero and Missed MicroFrame status bit to a one and the resultant status written back to memory. If siTDX1's Active bit is a zero, (because it was zero when the host controller first visited siTDX1 via siTDX's back pointer, it transitioned to zero as a result of a detected error, or the results of siTDX1's completesplit transaction transitioned it to zero), then the host controller returns to the context of siTDX and transitions its SplitXState to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (i.e. if cMicroframeBit is a 1b and siTDX.Smask[0] is a 1b). If this criterion is met the host controller immediately executes a startsplit transaction and appropriately advances the transaction state of siTDX, then follows siTDX.Next Pointer to the next schedule item. If the criterion is not met, the host controller simply follows siTDX.Next Pointer to the next schedule item. Note that in the case of a 2b boundary case, the splittransaction of siTDX1 will have its Active bit set to zero when the host controller returns to the context of siTDX. Also, note that software should not initialize an siTD with Cmask bits 0 and 1 set to a one and an Smask with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

15.6.11 Host Controller Pause

When the host controller's HCHalted bit in the USBSTS register is a zero, the host controller is sending SOF (Start Of Frame) packets down all enabled ports. When the schedules are enabled, the EHCI host controller will access the schedules in main memory each microframe. This constant pinging of main memory is known to create CPU power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the CPU, based on recent history usage. In the more aggressive power saving modes, the CPU can disable its caches. Current PC architectures assume that busmaster accesses to main memory must be cache coherent. So, when bus masters are busy touching memory, the CPU power management software can detect this activity over time and inhibit the transition of the CPU into its lowest power savings mode. USB controllers are busmasters and the frequency at which they access their memorybased schedules keeps the CPU power management software from placing the CPU

into its lowest power savings state. USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend. EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the CPU power management to get the CPU into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the CPU to reach it's low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the CPU will be required to process the data streams. It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing it's shared memory based data structures (schedule lists or otherwise).

15.6.12 Port Test Modes

EHCI host controllers must implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SEO_NAK as described in the USB Specification Revision 2.0. The system is only allowed to test ports that are owned by the EHCI controller (e.g. CFbit is a one and PortOwner bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the CFbit in the CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the Asynchronous Schedule Enable and Periodic Schedule Enable bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the Suspend bit in each appropriate PORTSC register to a one.
- Set the Run/Stop bit in the USBCMD register to a zero and wait for the HCHalted bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the mRun/Stop bit set to a one. However, all host controllers must support port testing with Run/Stop set to a zero and HCHalted set to a one.
- Set the Port Test Control field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, then the Run/Stop bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (HCHalted bit is a one) then it terminates and exits test mode by setting HCReset to a one.

15.6.13 Interrupts

The EHCI Host Controller hardware provides interrupt capability based on a number of sources. There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host Controller error events

All transactionbased sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section USBINTR). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt. During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the Interrupt Threshold Control field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro frames. This means that the host controller will not generate interrupts any more frequently than once every eight microframes. Section 5.15.2.4 details effects of a host system error. If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the

interrupt not being signaled until the next interrupt threshold. Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, CPU control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OSspecific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to deassert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register, see Section USBINTR) to a zero. The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register (Section USBSTS) from a one to a zero.

Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully. Table 43 lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the XactErr status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the XactErr status bit in thequeue head is set and the CErr field is decremented. When the PIDCode indicates a SETUP, the following responses are protocol errors and result in XactErr bit being set to a one and the CErr field being decremented.

- EPS field indicates a highspeed device and it returns a Nak handshake to a SETUP.
- EPS field indicates a highspeed device and it returns a Nyet handshake to a SETUP.
- EPS field indicates a low or fullspeed device and the completesplit receives a Nak handshake.

| Event / Result | Queue Head/qTD/iTD/siTD Sideeffects | | USB Status Register (USBSTS) USBERRINT |
|----------------------|-------------------------------------|-----------------------|---|
| | Cerr | Status Field | |
| CRC | -1 | XactErr set to a one. | 1 ¹ |
| Timeout | -1 | XactErr set to a one. | 1 ¹ |
| Bad PID ² | -1 | XactErr set to a one. | 1 ¹ |
| Babble | N/A | Section 5.15.1.1.1 | 1 |
| Buffer Error | N/A | Section 5.15.1.1.2 | |

517 . Summary of Transaction Errors

1. If occurs in a queue head, then USBERRINT is asserted only when CErr counts down from a one to a zero. In addition

2 the queue is halted, see Section 5.10.3.1.

The host controller received a response from the device, but it could not recognize the PID as a valid PID.

Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a Packet Babble. When a device sends more data than the Maximum Length number of bytes, the host controller sets the Babble Detected bit to a one and halts the endpoint if it is using a queue head (see Section 5.10.3.1). Maximum Length is defined as the minimum of Total Bytes to Transfer and Maximum Packet Size. The CErr field is not decremented for a packet babbled condition (only applies to queue heads). A babbled condition also exists if IN transaction is in progress at Highspeed EOF2 point. This is called a frame babbled. A frame babbled condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babbled is detected. The USBERRINT bit in the USBSTS register is set to a one and if the USB Error Interrupt Enable bit in the USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babbled across a microframe EOF.

Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction. This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the Data Buffer Error bit in the queue head, iTD or siTD. If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable.

An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDS, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USBSTS register to be set to a one. In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the USBERRINT bit in the USBSTS register is also set to a one.

Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the USBINT bit in the USBSTS register is set to a one. If the USB Interrupt Enable bit is set in the USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section 5.15.2.3).

Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the Port Change Detect bit in the USBSTS register to a one. If the Port Change Interrupt Enable bit in

the USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Overcurrent Change
- Force Port Resume

Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs. If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the Frame List Rollover bit in the USBSTS register to a one. If the Frame List Rollover Enable bit in the USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the onchip context of the asynchronous schedule, it evaluates the value of the Interrupt on Async Advance Doorbell bit in the USBCMD register. If it is a one, it sets the Interrupt on Async Advance bit in the USBSTS register to a one. If the Interrupt on Async Advance Enable bit in the USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule.

Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors. The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion. In the presence of noncatastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Hostbased error must result in the following actions:

- The Run/Stop bit in the USBCMD register is set to a zero.
- The following bits in the USBSTS register are set:
 - Host System Error bit is to a one.
 - HCHalted bit is set to a one.
- If the Host System Error Enable bit in the USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. Table 44 summarizes the required actions taken on the various host errors.

Table 44 Summary Behavior of EHCI Host Controller on Host System Errors

| Cycle Type | Master Abort | Target Abort | Data Phase Parity |
|--------------------|--------------|--------------|-------------------|
| Frame list pointer | Fatal | Fatal | Fatal [o] |
| fetch (read) | Fatal | Fatal | Fatal [o] |
| siTD fetch (read) | | | |

| Cycle Type | Master Abort | Target Abort | Data Phase Parity |
|----------------------------------|--------------|--------------|-------------------|
| siTD status writeback (write) | Fatal [o] | Fatal [o] | Fatal [o] |
| iTD fetch (read) | Fatal | Fatal | Fatal [o] |
| iTD status writeback (write) | Fatal [o] | Fatal [o] | Fatal [o] |
| qTD fetch (read) | Fatal | Fatal | Fatal [o] |
| qHD status writeback (write) | Fatal [o] | Fatal [o] | Fatal [o] |
| Data write | Fatal [o] | Fatal [o] | Fatal [o] |
| Data read | Fatal | Fatal | Fatal [o] |

518 . Summary Behavior of EHCI Host Controller on Host System Errors

[o] Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

15.6.14 Register Summary

Base Address:

| Register Name | Offset | Description |
|------------------|--------|---|
| USB ID | 0x0 | Identification Register |
| USB HWGENERAL | 0x4 | General Hardware Parameters Register |
| USB HWHOST | 0x8 | Host Hardware Parameters Register |
| USB HWDEVICE | 0xC | Device Hardware Parameters Register |
| USB HWTXBUF | 0x10 | TX Buffer Hardware Parameters Register |
| USB HWRXBUF | 0x14 | RX Buffer Hardware Parameters Register |
| USB SBUSCFG | 0x90 | Control for the System Bus interface Register |
| USB CAPLENGTH | 0x100 | Capability Length Register |
| USB HCIVERSION | 0x102 | Host Interface Version Number Register |
| USB HCSPARAMS | 0x104 | Host Ctrl. Structural Parameters Register |
| USB HCCPARAMS | 0x108 | Host Ctrl. Capability Parameters Register |
| USB DCIVERSION | 0x120 | Dev. Interface Version Number Register |
| USB DCCPARAMS | 0x124 | Device Ctrl. Capability Parameters Register |
| USB GPTIMER0LD | 0x80 | General Purpose Timer #0 Load Register |
| USB GPTIMER0CTRL | 0x84 | General Purpose Timer #0 Control Register |

| Register Name | Offset | Description |
|----------------------|--------|---|
| USB GPTIMER1LD | 0x88 | General Purpose Timer #1 Load Register |
| USB GPTIMER1CTRL | 0x8C | General Purpose Timer #1 Control Register |
| USBCMD | 0x140 | USB Command Register |
| USBSTS | 0x144 | USB Status Register |
| USBINTR | 0x148 | USB Interrupt Enable Register |
| USB FRINDEX | 0x14C | USB Frame Index Register |
| USB CTRLDSSEGMENT | 0x150 | Control Data Structure Segment Register |
| USB PERIODICLISTBASE | 0x154 | Frame List Base Address Register. Used only in Host Mode. |
| USB DEVICEADDR | 0x154 | USB Device Address Register. Used only in Device Mode. |
| USB ASYNCLISTADDR | 0x158 | Next Asynchronous List Address Register. Used only in Host Mode. |
| USB ENDPOINTLISTADDR | 0x158 | Address at Endpoint list in memory Register. Used only in Device Mode. |
| USB TTCTRL | 0x15C | TT Status and Control Register |
| USB BURSTSIZE | 0x160 | Programmable Burst Size Register |
| USB TXFILLTUNING | 0x164 | Host Transmit PreBuffer Packet Tuning Register |
| IC_USB | 0x16C | IC_USB enable and voltage negotiation Register |
| USB ULPI VIEWPORT | 0x170 | ULPI Viewport Register |
| USB ENDPTNAK | 0x178 | Endpoint NAK Register |
| USB ENDPTNAKEN | 0x17C | Endpoint NAK Enable Register |
| USB CONFIGFLAG | 0x180 | Configured Flag Register |
| USB PORTSC | 0x184 | Multi Port Status/Control Register |
| USB MODE | 0x1A8 | USB Device Mode Register |
| USB ENDPTSETUPSTAT | 0x1AC | Endpoint Setup Status Register |
| USB ENDPTPRIME | 0x1B0 | Endpoint Initialization Register |
| USB ENDPTFLUSH | 0x1B4 | Endpoint Deinitialize Register |
| USB ENDPTSTAT | 0x1B8 | Endpoint Status Register |
| USB ENDPTCOMPLETE | 0x1BC | Endpoint Complete Register |
| USB ENDPTCTRL0 | 0x1C0 | Endpoint Control Address 0 Register |
| USB ENDPTCTRL1 | 0x1C4 | Endpoint Control Address 1 Register |
| USB ENDPTCTRL2 | 0x1C8 | Endpoint Control Address 2 Register |

| Register Name | Offset | Description |
|-----------------|--------|--------------------------------------|
| USB ENDPTCTRL3 | 0x1CC | Endpoint Control Address 3 Register |
| USB ENDPTCTRL4 | 0x1D0 | Endpoint Control Address 4 Register |
| USB ENDPTCTRL5 | 0x1D4 | Endpoint Control Address 5 Register |
| USB ENDPTCTRL6 | 0x1D8 | Endpoint Control Address 6 Register |
| USB ENDPTCTRL7 | 0x1DC | Endpoint Control Address 7 Register |
| USB ENDPTCTRL8 | 0x1E0 | Endpoint Control Address 8 Register |
| USB ENDPTCTRL9 | 0x1E4 | Endpoint Control Address 9 Register |
| USB ENDPTCTRL10 | 0x1E8 | Endpoint Control Address 10 Register |
| USB ENDPTCTRL11 | 0x1EC | Endpoint Control Address 11 Register |
| USB ENDPTCTRL12 | 0x1F0 | Endpoint Control Address 12 Register |
| USB ENDPTCTRL13 | 0x1F4 | Endpoint Control Address 13 Register |
| USB ENDPTCTRL14 | 0x1F8 | Endpoint Control Address 14 Register |
| USB ENDPTCTRL15 | 0x1FC | Endpoint Control Address 15 Register |

519 . Register Summary Table

15.6.15 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, R/WC = Read/Clear on Write

USB IDENTIFICATION REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------|-----------|---|
| 31:24 | R | ID_CVERSION | 0x7 | Ones complement version of ID_VERSION. |
| 28:25 | R | ID_VERSION | 0x1 | Version number of the core |
| 24:21 | R | ID_REVISION | 0x2 | Revision number of the core |
| 20:16 | R | ID_TAG | 0x01 | Tag number of the core |
| 15:8 | R | ID_INV | 0xFA | Ones complement version of ID |
| 7:0 | R | ID | 0x05 | Configuration number. This number is set to 0x05 and indicates that the peripheral is the USBHS USB 2.0 core. |

520 . ID Register Description

USB GENERAL HARDWARE PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:12 | R | Reserved | | Reserved. These bits are reserved and should be set to zero. |
| 11 | R | IC | 0x0 | VUSB_HS_PHY_IC_USB is set to 0 as VUSB_HS_PHY_TYPE=0 |
| 10:9 | R | SM | 0x0 | VUSB_HS_PHY_SERIAL. This constant enables the IC-USB interface, but its also needed to set the VUSB_HS_PHY_SERIAL greater different from 0. |
| 8:6 | R | PHYM | 0x0 | VUSB_HS_PHY_TYPE |
| 5:4 | R | PHYW | 0x3 | VUSB_HS_PHY16_8 This constant selects which phy is being used 0 = UTMI/UMTI+ 1 = ULPI DDR 2 = ULPI 3 = Serial Only 4 = Software programmable - reset to UTMI 5 = Software programmable - reset to ULPI DDR 6 = Software programmable - reset to ULPI 7 = Software programmable - reset to Serial 8 = IC-USB 9 = Software programmable - reset to IC-USB |
| 3 | R | BWT | 0x0 | Reserved for internal testing. |
| 2:1 | R | CLKC | 0x2 | VUSB_HS_CLOCK_CONFIGURATION |
| 0 | R | RT | 0x1 | VUSB_HS_RESET_TYPE |

521 . HWGENERAL Register Description

USB HOST HARDWARE PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:24 | R | TTPER | 0x10 | VUSB_HS_TT_PERIODIC_CONTEXTS |
| 23:16 | R | TTASY | 0x2 | VUSB_HS_TT_ASYNC_CONTEXTS |
| 15:4 | R | Reserved | | Reserved. These bits are reserved and should be set to zero. |
| 3:1 | R | NPORT | 0x0 | VUSB_HS_NUM_PORT1 |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--------------|
| 0 | R | HC | 0x1 | VUSB_HS_HOST |

522 . HWHOST Register Description

USB DEVICE HARDWARE PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:6 | R | Reserved | | Reserved. These bits are reserved and should be set to zero. |
| 5:1 | R | DEVEP | 0x6 | VUSB_HS_DEV_EP |
| 0 | R | DC | 0x1 | device capable; [VUSB_HS_DEV /= 0]. |

523 . HWDEVICE Register Description

USB TX BUFFER HARDWARE PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------|-----------|---|
| 31 | R | TXCLR | 0x1 | This bit is fixed to 1'b1 so that the local context register's are included in the design. This means that the DMA context is implemented in Flip Flops. |
| 30:24 | R | Reserved | 0x00 | Reserved. These bits are reserved and should be set to zero. |
| 23:16 | R | TXCHANADD | 0x07 | VUSB_HS_TX_CHAN_ADD |
| 15:8 | R | TXADD | 0x0A | VUSB_HS_TX_ADD |
| 7:0 | R | TXBURST | 0x04 | VUSB_HS_TX_BURST |

524 . HWTXBUF Register Description

USB RX BUFFER HARDWARE PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:16 | R | Reserved | 0x0000 | Reserved. These bits are reserved and should be set to zero. |
| 15:8 | R | RXADD | 0x08 | VUSB_HS_RX_ADD |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|------------------|
| 7:0 | R | RXBURST | 0x04 | VUSB_HS_RX_BURST |

525 . HWRXBUF Register Description

USB SYSTEM BUS CONFIG REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:3 | R/W | Reserved | | Reserved |
| 2:0 | R/W | AHBBRST | 0x1 | AMBA AHB BURST. This is a r/w field that selects the following options for the m_hburst signal of the AMBA master interface: <ul style="list-style-type: none">• 3'b000: INCR burst of unspecified length• 3'b001: INCR4, non multiple transfers of INCR4 will be decomposed into singles• 3'b010: INCR8, non multiple transfers of INCR8, will be decomposed into INCR4 or singles• 3'b011: INCR16, non multiple transfers of INCR16, will be decomposed into INCR8, INCR4 or singles• 3'b100: This value is reserved and should not be used• 3'b101: INCR4, non multiple transfers of INCR4 will be decomposed into smaller unspecified length bursts• 3'b110: INCR8, non multiple transfers of INCR8 will be decomposed into smaller unspecified length bursts• 3'b111: INCR16, non multiple transfers of INCR16 will be decomposed into smaller unspecified length bursts. |

526 . SBUSCFG Register Description

USB CAPABILITY LENGTH REGISTER

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|--|
| 7:0 | R | CAPLENGTH | 0x40 | This register is used to indicate which offset to add to the register base address at the beginning of the Operational Register. |

527 . CAPLENGTH Register Description

USB HOST INTERFACE VERSION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|-------------------------------|
| 15:0 | R | HCIVERSION | 0x0100 | Host Interface version number |

528 . HCIVERSION Register Description

USB HOST CONTROL STRUCTURAL PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:28 | R | Reserved | | Reserved. These bits are reserved and should be set to zero. |
| 27:24 | R | N_TT | 1 | Number of Transaction Translators (N_TT). This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. For Multi-Port Host this field will always equal "0001". For all other implementation, N_TT = "0000". This in a nonEHCI field to support embedded TT. |
| 23:20 | R | N_PTT | 1 | Number of Ports per Transaction Translator (N_PTT). This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. For MultiPort Host this field will always equal N_PORTS. For all other implementations, N_PTT = "0000".This in a nonEHCI field to support embedded TT. |
| 19:17 | R | Reserved | | Reserved. |
| 16 | R | PI | 1 | Port Indicators (P INDICATOR). This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator. This field will always be "1". |
| 15:12 | R | N_CC | 0 | Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. A zero in this field indicates there are no internal Companion Controllers. Port ownership handoff is not supported. A value larger than zero in this field indicates there are companion USB1.1 host controller(s). Portownership handoffs are supported. High, Full and Lowspeed devices are supported on the host controller root ports. In this implementation this field will always be "0". |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 11:8 | R | N_PCC | 0 | <p>Number of Ports per Companion Controller. This field indicates the number of ports supported per internal Companion Controller.</p> <p>It is used to indicate the port routing configuration to the system software.</p> <p>For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3.</p> <p>The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc.</p> <p>In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2.</p> <p>The number in this field must be consistent with N_PORTS and N_CC. In this implementation this field will always be "0".</p> |
| 7:5 | R | Reserved | | Reserved. |
| 4 | R | PPC | 0x1 | <p>Port Power Control. This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches.</p> <p>A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register.</p> <p>This field will always be "0" for a device only implementation.</p> |
| 3:0 | R | N_PORTS | 0x1 | <p>Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller.</p> <p>The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 0x1 to 0xF.</p> <p>A zero in this field is undefined. The number of ports for a host implementation is parameterizable from 1 to 8. This field will always be 1 for device only implementation.</p> |

529 . HCSPARAMS Register Description

USB HOST CONTROL CAPABILITY PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|-------------|
| 31:8 | R | Reserve 0d | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 7:4 | R | IST | 0x0 | <p>Isochronous Scheduling Threshold.</p> <p>This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of microframes a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".</p> |
| 3 | R | Reserve | 0x0d | Reserved |
| 2 | R | ASP | 0x1 | <p>Asynchronous Schedule Park Capability.</p> <p>If this bit is set to a one, then the host controller supports the park feature for highspeed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field will always be "1".</p> |
| 1 | R | PFL | 0x1 | <p>Programmable Frame List Flag. If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller.</p> <p>The USBCMD register Frame List Size field is a readonly register and must be set to zero. PFL If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4Kpage boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".</p> |
| 0 | R | ADC | 0x0 | <p>64bit Addressing Capability.</p> <p>This field will always be "0". No 64bit addressing capability is supported.</p> |

530 . HCCPARAMS Register Description

USB DEVICE INTERFACE VERSION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|---|
| 15:0 | R | DCIVERSION | 0x0001 | The device controller interface conforms to the two byte BCD encoding of the interface version number contained in this register. |

531 . DCIVERSION Register Description

USB DEVICE CONTROL CAPABILITY PARAMETERS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:9 | R | Reserved | | Reserved. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 8 | R | HC | 0x1 | Host Capable. When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller. |
| 7 | R | DC | 0x1 | Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device. |
| 6:5 | R | Reserved | | Reserved |
| 4:0 | R | DEN | 0x6 | Device Endpoint Number. This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0– 16. |

532 .DCCPARAMS Register Description

USB GENERAL PURPOSE TIMER 0 LOAD REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:24 | R/W | Reserved | | Reserved. |
| 23:0 | R/W | GPTLD | 0 | General Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 10001=999 or 0x0003E7. Note: Max value is 0xFFFFFFF or 16.777215 seconds. |

533 .GPTIMER0LD Register Description

USB GENERAL PURPOSE TIMER 0 CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 31 | R/W | GTPRUN | 0 | General Purpose Timer Run. (0) – Timer stop; (1) – Timer Run. This bit enables the generalpurpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT counter value. |
| 30 | R/W | GPTRST | 0 | General Purpose Timer Reset. (0) – No action; (1) – Load Counter Value. Writing a one to this bit will reload the GPTCNT with the value in GPTLD. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 29:25 | R/W | Reserved | | Reserved |
| 24 | R/W | GPTMODE | 0 | <p>General Purpose Timer Mode. (0) – One Shot ; (1) – Repeat. This bit selects between a single timer countdown and a looped count down.</p> <p>In oneshot mode, the timer will count down to zero, generate and interrupt, and stop until the counter is reset by software.</p> <p>In repeat mode, the timer will count down to zero, generate and interrupt, and automatically reload the counter to begin again.</p> |
| 23:0 | R | GPTCNT | 0 | General Purpose Timer Counter. This field is the value of the running timer. |

534 . GPTIMER0CTRL Register Description

USB GENERAL PURPOSE TIMER 1 LOAD REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:24 | R/W | Reserved | | Reserved. |
| 23:0 | R/W | GPTLD | 0 | <p>General Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action.</p> <p>This value in this register represents the time in microseconds minus 1 for the timer duration.</p> <p>Example: for a one millisecond timer, load 10001=999 or 0x0003E7.</p> <p>Note: Max value is 0xFFFFFFF or 16.777215 seconds.</p> |

535 . GPTIMER1LD Register Description

USB GENERAL PURPOSE TIMER 1 CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 31 | R/W | GTPRUN | 0 | <p>General Purpose Timer Run. (0) – Timer stop; (1) – Timer Run. This bit enables the generalpurpose timer to run.</p> <p>Setting or clearing this bit will not have an effect on the GPTCNT counter value.</p> |
| 30 | W | GPTRST | 0 | General Purpose Timer Reset. (0) – No action; (1) – Load Counter Value. Writing a one to this bit will reload the GPTCNT with the value in GPTLD. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 29:25 | R/W | Reserved | | Reserved |
| 24 | R/W | GPTMODE | 0 | <p>General Purpose Timer Mode. (0) – One Shot ; (1) – Repeat. This bit selects between a single timer countdown and a looped count down.</p> <p>In oneshot mode, the timer will count down to zero, generate and interrupt, and stop until the counter is reset by software.</p> <p>In repeat mode, the timer will count down to zero, generate and interrupt, and automatically reload the counter to begin again.</p> |
| 23:0 | R | GPTCNT | 0 | General Purpose Timer Counter. This field is the value of the running timer. |

536 . GPTIMER1CTRL Register Description

USB COMMAND REGISTER

| B it | A cc es s | F un ctio n | POR Value | Description | | | | | | | | | | | | | | | | | | |
|-----------|----------------------------|----------------------|--------------|---|-------|----------------------------|------|--------------------------|------|--------------|------|---------------|------|---------------|------|---------------|------|----------------|------|----------------|------|----------------|
| 31:2 4 | R/W | R | 0 | Reserved | | | | | | | | | | | | | | | | | | |
| 23:1 6 | R/W | ITC | 0x08 | <p>Interrupt Threshold Control</p> <p>The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in microframes. Valid values are shown below.</p> <table> <thead> <tr> <th>Value</th> <th>Maximum Interrupt Interval</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>Immediate (no threshold)</td> </tr> <tr> <td>0x01</td> <td>1 microframe</td> </tr> <tr> <td>0x02</td> <td>2 microframes</td> </tr> <tr> <td>0x04</td> <td>4 microframes</td> </tr> <tr> <td>0x08</td> <td>8 microframes</td> </tr> <tr> <td>0x10</td> <td>16 microframes</td> </tr> <tr> <td>0x20</td> <td>32 microframes</td> </tr> <tr> <td>0x40</td> <td>64 microframes</td> </tr> </tbody> </table> | Value | Maximum Interrupt Interval | 0x00 | Immediate (no threshold) | 0x01 | 1 microframe | 0x02 | 2 microframes | 0x04 | 4 microframes | 0x08 | 8 microframes | 0x10 | 16 microframes | 0x20 | 32 microframes | 0x40 | 64 microframes |
| Value | Maximum Interrupt Interval | | | | | | | | | | | | | | | | | | | | | |
| 0x00 | Immediate (no threshold) | | | | | | | | | | | | | | | | | | | | | |
| 0x01 | 1 microframe | | | | | | | | | | | | | | | | | | | | | |
| 0x02 | 2 microframes | | | | | | | | | | | | | | | | | | | | | |
| 0x04 | 4 microframes | | | | | | | | | | | | | | | | | | | | | |
| 0x08 | 8 microframes | | | | | | | | | | | | | | | | | | | | | |
| 0x10 | 16 microframes | | | | | | | | | | | | | | | | | | | | | |
| 0x20 | 32 microframes | | | | | | | | | | | | | | | | | | | | | |
| 0x40 | 64 microframes | | | | | | | | | | | | | | | | | | | | | |
| 15 | R/W | FS2 | 0 | | | | | | | | | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|--|--------------------------------------|--|
| 14 | R/W | ATDTW | 0 | Add dTD TripWire [device mode only] This bit is used as a semaphore to ensure the proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit shall also be cleared by hardware when state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized. For more information on the use of this bit, see the Device Operational Model section of the USBHS DEV reference manual. |
| 13 | R/W | SUTW | 0 | Setup TripWire [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (See USBMODE) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. For more information on the use of this bit, see the Device Operational Model section of the USBHS DEV reference manual. |
| 12 | R/W | R | 0 | Reserved |
| 11 | R/W | ASPE | 0x0 --device mode 0x1 -- hostmode | Asynchronous Schedule Park Mode Enable Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. |
| 10 | R/W | R | 0 | Reserved |
| 9:8 | R/W | ASP[1:0] --device mode 3 --hostmode | 0 | Asynchronous Schedule Park Mode Count It contains a count of the number of successive transactions the host controller is allowed to execute from a highspeed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 0x1 to 0x3. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. |
| 7 | R | LR | 0 | Light Host/Device Controller Reset (OPTIONAL) Not Implemented. This field will always be "0". |
| 6 | R/W | IAA | 0 | Interrupt on Async Advance Doorbell This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results. |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 5 | R/W | ASE | 0 | <p>Asynchronous Schedule Enable</p> <p>This bit controls whether the host controller skips processing the Asynchronous Schedule.</p> <p>Values meaning</p> <ul style="list-style-type: none"> 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. |
| 4 | R/W | PSE | 0 | <p>Periodic Schedule Enable</p> <p>This bit controls whether the host controller skips processing the Periodic Schedule.</p> <p>Values meaning</p> <ul style="list-style-type: none"> 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. |
| 3 | R/W | FS1 | 0 | |
| 2 | R/W | FS0 | 0 | |
| 1 | R/W | RST | 0 | <p>Controller Reset (RESET)</p> <p>Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register.</p> <p>Host Controller:</p> <p>When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior.</p> <p>Device Controller:</p> <p>When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 0 | R/W | RS | 0 | <p>Run/Stop (RS)</p> <p>1=Run. 0=Stop.</p> <p>Host Controller:</p> <p>When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e. HCHalted in the USBSTS register is a one).</p> <p>Device Controller:</p> <p>Writing a one to this bit will cause the device controller to enable a pullup on D+ and initiate an attach event. This control bit is not directly connected to the pullup enable, as the pullup will become disabled upon transitioning into highspeed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event.</p> |

537 . USBCMD Register Description

USB STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:26 | R/W | Reserved | 0 | |
| 25 | R/WC | TI1 | 0 | <p>General Purpose Timer Interrupt 1 (GPTINT1)</p> <p>This bit is set when the counter in the GPTIMER1CTRL (NonEHCI) register transitions to zero. Writing a one to this bit will clear it.</p> |
| 24 | R/WC | TIO | 0 | <p>General Purpose Timer Interrupt 0 (GPTINT0)</p> <p>This bit is set when the counter in the GPTIMER0CTRL (NonEHCI) register transitions to zero. Writing a one to this bit will clear it.</p> |
| 23:20 | R/W | Reserved | 0 | |
| 19 | R/WC | UPI | 0 | <p>USB Host Periodic Interrupt (USBHSTPERINT)</p> <p>This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|----------------------------------|--|
| 18 | R/WC | UAI | 0 | <p>USB Host Asynchronous Interrupt (USBHSTASYNCINT)</p> <p>This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero.</p> |
| 17 | R/W | Reserve0d | 0 | |
| 16 | R | NAKI | 1 -- hostmode 0 – device mode | <p>NAK Interrupt Bit</p> <p>It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set.</p> <p>This bit is automatically cleared by hardware when all the enabled TX/RX Endpoint NAK bits are cleared.</p> |
| 15 | R | AS | 0 | <p>Asynchronous Schedule Status</p> <p>This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled.</p> <p>The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register.</p> <p>When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used by the host controller.</p> |
| 14 | R | PS | 0 | <p>Periodic Schedule Status</p> <p>This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled.</p> <p>The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register.</p> <p>When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used by the host controller.</p> |
| 13 | R | RCL | 0 | <p>Reclamation</p> <p>This is a readonly status bit used to detect an empty asynchronous schedule. Only used by the host controller.</p> |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 12 | R | HCH | 1 | <p>HCHalted</p> <p>This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). Only used by the host controller</p> |
| 11:9 | R/W | Reserved | 0 | |
| 8 | R/WC | SLI | 0 | <p>DCSuspend</p> <p>When a device controller enters a suspend state from an active state, this bit will be set to a one. This bit is only cleared by software writing a 1 to it. Only used by the device controller.</p> |
| 7 | R/WC | SRI | 0 | <p>SOF Received</p> <p>When the device controller detects a Start Of (micro)Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125us in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a nonEHCI status bit.</p> |
| 6 | R/WC | URI | 0 | <p>USB Reset Received</p> <p>When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller.</p> |
| 5 | R/WC | AAI | 0 | <p>Interrupt on Async Advance</p> <p>System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller.</p> |
| 4 | R/WC | SEI | 0 | <p>System Error</p> <p>In the BVCI implementation of the USBHS core, this bit is not used, and will always be cleared to "0". In the AMBA implementation, this bit will be set to "1" when an Error response is seen by the master interface (HRESP[1:0]=ERROR).</p> |
| 3 | R/WC | FRI | 0 | <p>Frame List Rollover</p> <p>The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1:3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used by the host controller.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 2 | R/WC | PCI | 0 | <p>Port Change Detect</p> <p>The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a JK transition on the suspended port. The Device Controller sets this bit to a one when it detects resume signaling or the port controller enters the full or highspeed operational state. When the port controller exits the full or highspeed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible.</p> |
| 1 | R/WC | UEI | 0 | <p>USB Error Interrupt (USBERRINT)</p> <p>When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set</p> <p>UEI See Section (Reference Host Operation Model: Transfer/Transaction Based Interrupt – i.e.4.15.1 in EHCI 13) for a complete list of host error interrupt conditions.</p> |
| 0 | R/WC | UI | 0 | <p>USB Interrupt (USBINT)</p> <p>This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.</p> |

538 . USBSTS Register Description

USB INTERRUPT ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:26 | R/W | R | 0 | Reserved |
| 25 | R/W | TIE1 | 0 | <p>General Purpose Timer Interrupt Enable 1 :</p> <p>When this bit is a one, and the GPTINT1 bit in the USBSTS register is a one, the controller will issue an interrupt.</p> <p>The interrupt is acknowledged by software clearing the GPTINT1 bit.</p> |
| 24 | R/W | TIE0 | 0 | <p>General Purpose Timer Interrupt Enable 0 :</p> <p>When this bit is a one, and the GPTINT0 bit in the USBSTS register is a one, the controller will issue an interrupt.</p> <p>The interrupt is acknowledged by software clearing the GPTINT0 bit.</p> |
| 23:20 | R/W | R | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 19 | R/W | UPIA | 0 | |
| 18 | R/W | UAIE | 0 | USB Host Async. Interrupt enable : When this bit is a one, and the USBHSTASYNCINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTASYNCINT bit. |
| 17 | R/W | R | 0 | Reserved |
| 16 | R/W | NAKE | 0 | NAK Interrupt Enable : This bit is set by software if it wants to enable the hardware interrupt for the NAK Interrupt bit. If both this bit and the corresponding NAK Interrupt bit are set, a hardware interrupt is generated. |
| 15:9 | R/W | Reserved | 0 | Reserved |
| 8 | R/W | SLE | 0 | Sleep Enable: When this bit is a one, and the DCSuspend bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the DCSuspend bit. Only used by the device controller. |
| 7 | R/W | SRE | 0 | SOF Receive Enable: When this bit is a one, and the SOF Received bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the SOF Received bit. |
| 6 | R/W | URE | 0 | USB Reset Enable: When this bit is a one, and the USB Reset Received bit in the USBSTS register is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 5 | R/W | AAE | 0 | Interrupt on Async. Advance Enable : When this bit is a one, and the Interrupt on Async Advance bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. |
| 4 | R/W | SEE | 0 | System Error Enable: When this bit is a one, and the System Error bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the System Error bit. |
| 3 | R/W | FRE | 0 | Framelist Roll over Enable: When this bit is a one, and the Frame List Rollover bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. |
| 2 | R/W | PCE | 0 | Port Change Detect Enable: When this bit is a one, and the Port Change Detect bit in the USBSTS register is a one, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit. |
| 1 | R/W | UEE | 0 | USB Error Interrupt Enable: When this bit is a one, and the USBERRINT bit in the USBSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register |
| 0 | R/W | UE | 0 | USB Interrupt Enable: When this bit is a one, and the USBINT bit in the USBSTS register is a one, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit. |

539 . USB INTR Register Description

USB FRAME INDEX REGISTER

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | |
|----------------|---------------------------------|----------|-----------|---|---------------|---------------------------------|----------------|----|---------------|----|---------------|----|---------------|---|--------------|---|--------------|---|--------------|---|-------------|---|
| 31:14 | R/W | Reserved | | Reserved | | | | | | | | | | | | | | | | | | |
| 13:0 | R/W | FRINDEX | 0x0000 | <p>Frame Index.</p> <p>The value, in this register, increments at the end of each time frame (e.g. microframe). Bits [N: 3] are used for the Frame List current index.</p> <p>This means that each location of the frame list is accessed 8 times (frames or microframes) before moving to the next index.</p> <p>The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode.</p> <table> <thead> <tr> <th>USBCMD Number</th> <th>[Frame List Size] Elements N</th> </tr> </thead> <tbody> <tr> <td>000b (1024)</td> <td>12</td> </tr> <tr> <td>001b (512)</td> <td>11</td> </tr> <tr> <td>010b (256)</td> <td>10</td> </tr> <tr> <td>011b (128)</td> <td>9</td> </tr> <tr> <td>100b (64)</td> <td>8</td> </tr> <tr> <td>101b (32)</td> <td>7</td> </tr> <tr> <td>110b (16)</td> <td>6</td> </tr> <tr> <td>111b (8)</td> <td>5</td> </tr> </tbody> </table> <p>In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe.</p> | USBCMD Number | [Frame List Size] Elements N | 000b (1024) | 12 | 001b (512) | 11 | 010b (256) | 10 | 011b (128) | 9 | 100b (64) | 8 | 101b (32) | 7 | 110b (16) | 6 | 111b (8) | 5 |
| USBCMD Number | [Frame List Size] Elements N | | | | | | | | | | | | | | | | | | | | | |
| 000b (1024) | 12 | | | | | | | | | | | | | | | | | | | | | |
| 001b (512) | 11 | | | | | | | | | | | | | | | | | | | | | |
| 010b (256) | 10 | | | | | | | | | | | | | | | | | | | | | |
| 011b (128) | 9 | | | | | | | | | | | | | | | | | | | | | |
| 100b (64) | 8 | | | | | | | | | | | | | | | | | | | | | |
| 101b (32) | 7 | | | | | | | | | | | | | | | | | | | | | |
| 110b (16) | 6 | | | | | | | | | | | | | | | | | | | | | |
| 111b (8) | 5 | | | | | | | | | | | | | | | | | | | | | |

540 .FRINDEX Register Description

USB CONTROL DATASTRUCTURE SEGMENT REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-------------|---|
| 31:0 | R | DSSEGMENT | 0x0000_0000 | This register is not used in this implementation. This register is programmed with 4Gigabyte segment where all of the interface data structures are allocated. |

541 .CTRLDSSEGMENT Register Description

USB FRAME LIST BASE ADDRESS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:12 | R/W | PERBASE | 0 | Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller. |
| 11:0 | R/W | Reserved | 0 | Reserved |

542 .PERIODICLISTBASE Register Description

USB DEVICE ADDRESS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:25 | R/W | USBADR | 0 | Device Address. These bits correspond to the USB device address. |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 24 | R/W | USBADRA | 0 | <p>Device Address Advance.</p> <p>When this bit is ‘0’, any writes to USBADR are instantaneous.</p> <p>When this bit is written to a ‘1’ at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register.</p> <p>After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions:</p> <ul style="list-style-type: none"> 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). <p>Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field.</p> <p>This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase.</p> <p>If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.</p> |
| 23:0 | R/W | Reserved | | Reserved |

543 . DEVICE ADDR Register Description

USB ASYNC LIST ADDRESS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:5 | R/W | ASYBASE | 0 | <p>Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively.</p> <p>This field may only reference a Queue Head (OH). Only used by the host controller.</p> |
| 4:0 | R/W | Reserved | | Reserved. |

544 . ASYNCLISTADDR Register Description

USB ENDPOINT LIST ADDRESS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:11 | R/W | EPBASE | 0 | <p>Endpoint List Pointer (Low). These bits correspond to memory address signals [31:11], respectively.</p> <p>This field will reference a list of up to 32 Queue Heads (QH). (i.e. one queue head per endpoint & direction.)</p> |
| 10:0 | R/W | Reserved | | Reserved |

545 . ENDPOINTLISTADDR Register Description

USB TT CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:0 | R/W | | | <p>This register contains parameters needed for internal TT operations. This Register is not used in the device controller operation.</p> |

546 . TTCTRL Register Description

USB BURSTSIZE REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:16 | R/W | Reserved | | Reserved. |
| 15:8 | R/W | TXPBURST | 0x04 | <p>Programmable TX Burst Length.</p> <p>This register represents the maximum length of a the burst in 32 bit words while moving data from system memory to the USB bus.</p> <p>If field AHBBRST of register SBUSCFG(0x090) is different from zero, this field TXPBURST will return the value of the INCRx length.</p> <p>Supported values are integer values from 4 to 128. It is recommended to set this value to a integer submultiple of VUSB_HS_TX_CHAN.</p> <p>Different values will not use all the available buffer space, preventing proper TX endpoint priming in stream disable mode (SDIS bit set to '1').</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 7:0 | R/W | RXPBURST | 0x04 | <p>Programmable RX Burst Length.</p> <p>This register represents the maximum length of a the burst in 32 bit words while moving data from the USB bus to system memory.</p> <p>If field AHBBRST of register SBUSCFG(0x090) is different from zero, this field RXPBRUST will return the value of the INCRx length.</p> <p>The supported values are integer values from 4 to 128. It is recommended to set this value to a integer sub-multiple of VUSB_HS_RX_DEPTH.</p> |

547 .BURSTSIZE ADDR Register Description

USB HOST TX PREBUFFER PACKET TUNING REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------|-----------|---|
| 31:22 | R/W | Reserved | | Reserved |
| 21:16 | R/W | TXFIFOTHRES | 2 | <p>FIFO Burst Threshold. This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus.</p> <p>The minimum value is 2 and this value should be a low as possible to maximize USB performance.</p> <p>A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.</p> |
| 15:13 | R/W | Reserved | | Reserved |
| 12:8 | R/W | TXSCHEALTH | 0 | <p>Scheduler Health Counter.</p> <p>This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next StartOfFrame .</p> <p>This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.</p> |
| 7 | R/W | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 6:0 | R/W | TXSCHOH | 0 | <p>Scheduler Overhead. This register adds an additional fixed offset to the schedule time estimator described above as Tff.</p> <p>As an approximation, the value chosen for this register should limit the number of backoff events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus.</p> <p>Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization.</p> <p>The time unit represented in this register is 1.267us when a device is connected in High Speed Mode for OTG & SPH.</p> <p>The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode for OTG & SPH. The time unit represented in this register is always 1.267 the MPH product.</p> |

548 . TXFILLTUNING Register Description

IC USB ENABLE REGISTER

| B it | A cc e ss | F un ctio n | P OR V al u e | D e sc ri p t i o n |
|--|--------------------|-------------------------|------------------------------|---|
| 31,2 7, 23,1 9, 15,1 1 3,7 | R/W | IC8,IC70 ,... IC1 | IC8,IC7, ... IC1 | InterChip transceiver enable. These bits enables the InterChip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to "11" in the PORTSCx. Writing a '1' to each bit selects the IC_USB interface for that port. If the Controller is not MultiPort, IC8 to IC2 will be '0' and ReadOnly. |

| Bit | Access | Function | POR Value | Description |
|--|---------------|---|------------------|--|
| 30:2 8, 26:2 4, 22:2 0, 18:1 6, 14:1 2, 10:8, 6:4, 2:0 | R | IC_VDDx D8, IC_VDD7,.. IC_VDD1 | 0 | <p>IC_VDDx</p> <p>It selects which voltage is being supplied to the peripheral through each port (MPH case).</p> <p>000 > No voltage</p> <p>001 > 1.0V</p> <p>010 > 1.2V</p> <p>011 > 1.5V</p> <p>100 > 1.8</p> <p>101 > 3.0</p> <p>110 > Reserved</p> <p>111 > Reserved</p> <p>This field is set to '000' in case of a device controller. IC_VDD8 to IC_VDD2 are readonly and set to '000' in case the controller is not MultiPort.</p> <p>The Voltage negotiation should happen between enabling port power (PP) and asserting the run/stop bit in register.</p> |

549 .IC_USB Register Description

USB ULPI VIEWPORT REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 31 | R/W | ULPIWU | 0x0 | <p>ULPI Wakeup</p> <p>Writing the '1' to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver can not set it back to '0'.</p> |
| 30 | R/W | ULPIRUN | 0x0 | <p>ULPI Read/Write Run</p> <p>Writing the '1' to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver can not set it back to '0'.</p> |
| 29 | R/W | ULPIRW | 0x0 | <p>ULPI Read/Write Control</p> <p>(0) – Read; (1) – Write.</p> <p>This bit selects between running a read or write operation.</p> |
| 28 | R/W | Reserved | | Reserved. This bit is reserved and its value has no effect on operation |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 27 | R/W | ULPISS | 0x0 | <p>ULPI Sync State (1) – Normal Sync. State. (0) In another state (i.e. carkit, serial, low power)</p> <p>This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multiport host. Otherwise, this field should always remain 0</p> |
| 26:24 | R/W | ULPIPORT | 0x0 | <p>ULPI Port Number</p> <p>For the wakeup or read/write operation to be executed, this value selects the port number the ULPI PHY is attached to in the multi port host. The range is 0 to 7. This field should always be written as a 0 for the non multi port products.</p> |
| 23:16 | R/W | ULPIADDR | 0x00 | <p>ULPI Data Address</p> <p>When a read or write operation is commanded, the address of the operation is written to this field.</p> |
| 15:8 | R/W | ULPIDATRD | 0x00 | <p>ULPI Data Read</p> <p>After a read operation completes, the result is placed in this field.</p> |
| 7:0 | R/W | ULPIDATWR | 0x00 | <p>ULPI Data Write</p> <p>When a write operation is commanded, the data to be sent is written to this field.</p> |

550 . ULPVIEWPORT Register Description

USB ENDPOINT NAK REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 31:16 | R/WC | EPTN | 0 | <p>TX Endpoint NAK</p> <p>Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint.</p> <p>Bit 15 – Endpoint #15 Bit 1 – Endpoint #1 Bit 0 – Endpoint #0</p> |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 15:0 | R/WC | EPRN | 0 | <p>RX Endpoint NAK</p> <p>Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint.</p> <p>Bit 15 – Endpoint #15</p> <p>Bit 1 – Endpoint #1</p> <p>Bit 0 – Endpoint #0</p> |

551 .ENDPTNAK Register Description

USB ENDPOINT NAK ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:16 | R/W | EPTNE | 0 | <p>TX Endpoint NAK Enable</p> <p>Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set.</p> <p>Bit 15 – Endpoint #15</p> <p>Bit 1 – Endpoint #1</p> <p>Bit 0 – Endpoint #0</p> |
| 15:0 | R/W | EPRNE | 0 | <p>RX Endpoint NAK Enable</p> <p>Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set.</p> <p>Bit 15 – Endpoint #15</p> <p>Bit 1 – Endpoint #1</p> <p>Bit 0 – Endpoint #0</p> |

552 .ENDPTNAKEN Register Description

USB CONFIGURED FLAG REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|--|
| 31:1 | R | Reserved | 0x0 | |
| 0 | R | Port ctrl | 0x1 | A read from this register returns a constant of a 0x1 to indicate that all port routings default to this host controller |

553 .CONFIG FLAG Register Description

USB MULTI PORT STATUS/CONTROL Register x

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:0 | R | PTS | | <p>Parallel Transceiver Select</p> <p>This register bit pair is used in conjunction with the configuration constant VUSB_HS_PHY_TYPE to control which parallel transceiver interface is selected.</p> <p>This field is reset to: "00" as VUSB_HS_PHY_TYPE = 0 – UTMI/UTMI+</p> |
| 29 | R | STS | | <p>Serial Transceiver Select</p> <p>This register bit is used in conjunction with the configuration constant VUSB_HS_PHY_SERIAL to control whether the parallel or serial transceiver interface is selected for FS and LS operation. The Serial Interface Engine can be used in combination with the UTMI+ or ULPI physical interface to provide FS/LS signaling instead of the parallel interface. This bit has no effect unless Parallel Transceiver Select is set to UTMI+ or ULPI. The Serial/1.1 physical interface will use the Serial Interface Engine for FS/LS signaling regardless of this bit value.</p> <p>Note: This bit is reserved for future operation and is a placeholder adding dynamic use of the serial engine in accord with UMTI+ and ULPI characterization logic. This bit is not defined in the EHCI specification.</p> |
| 28 | R/W | PTW | | <p>Parallel Transceiver Width</p> <p>This register bit is used in conjunction with the configuration constant VUSB_HS_PHY16_8 to control whether the data bus width of the UTMI transceiver interface. This bit has no effect as the Serial interface is selected.</p> <p>This bit is not defined in the EHCI specification.</p> |
| 27:6 | R | PSPD | | <p>Port Speed</p> <p>This register field indicates the speed at which the port is operating. For HS mode operation in the host controller and HS/FS operation in the device controller the port routing steers data to the Protocol engine. For FS and LS mode operation in the host controller, the port routing steers data to the Protocol Engine w/ Embedded Transaction Translator.</p> <p>00 – Full Speed</p> <p>01 – Low Speed</p> <p>10 – High Speed</p> <p>This bit is not defined in the EHCI specification.</p> |
| 25 | R/W | SRT | | <p>SRT *Shorten Reset Time.* When the core is acting as a Host Controller, this bit enables a bypass of the Chirp J/K reset handshake, saving 67ms in simulation time for each reset sequence.</p> <p>This bit should only be used for initial system integration simulations, and should always be set to 0 for normal operation. This bit is not defined in the EHCI specification.</p> |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 24 | R/W | PFSC | 0 | <p>Port Force Full Speed Connect</p> <p>Writing this bit to a 1b will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device. This bit is not defined in the EHCI specification. This bit is for debugging purposes.</p> |
| 23 | R/W | PHCD | 0 | <p>PHY Low Power Suspend Clock Disable (PLPSCD)</p> <p>Writing this bit to a 1b will disable the PHY clock. Writing a 0b enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend – Clock Disable when the device is not running (USBCMD Run/Stop=0b) or the host has signaled suspend (PORTSC SUSPEND=1b). Low power suspend will be cleared automatically when the host has signaled resume if using a circuit similar to that in 10. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend – Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. See 10 System Level Issues and Core Configuration for more discussion on clock disable and power down issues.</p> <p>This bit is not defined in the EHCI specification.</p> <p>When this bit is set to 1, and the PHY clock stops in Suspend mode, register bits ATDTW and SRI , and register ENDPTFLUSH are not available for reading, as they are located in the PHY clock domain.</p> |
| 22 | R/W | WKOC | 0 | <p>Wake on Overcurrent Enable (WKOC_E)</p> <p>Writing this bit to a one enables the port to be sensitive to overcurrent conditions as wakeup events. This field is zero if Port Power(PP) is zero. This bit is output from the controller as signal pwrctl_wake_ovcurr_en (OTG/host core only) for use by an external power control circuit.</p> |
| 21 | R/W | WKDS | 0 | <p>Wake on Disconnect Enable (WKDSCNNT_E)</p> <p>Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit is output from the controller as signal pwrctl_wake_dscnnt_en (OTG/host core only) for use by an external power control circuit.</p> |
| 21 | R/W | WKCN | 0 | <p>Wake on Connect Enable (WKCNNT_E)</p> <p>Writing this bit to a one enables the port to be sensitive to device connects as wakeup events. This field is zero if Port Power(PP) is zero or in device mode. This bit is output from the controller as signal pwrctl_wake_cnnt_en (OTG/host core only) for use by an external power control circuit.</p> |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | | | |
|----------------|---------------------------|-----------------|------------------|---|-----------|---------|-------|-------------------------|-------|---------|-------|---------|-------|---------------------------|-------|--------|-------|-----------------|-------|-----------------|-------|-----------------|----------------|----------|
| 19:1 6 | R/W | PTC | 0 | <p>Port Test Control</p> <p>Any other value than zero indicates that the port is operating in test mode.</p> <table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0000b</td> <td>TEST_MODE_DISABLE</td> </tr> <tr> <td>0001b</td> <td>J_STATE</td> </tr> <tr> <td>0010b</td> <td>K_STATE</td> </tr> <tr> <td>0011b</td> <td>SE0 (host) / NAK (device)</td> </tr> <tr> <td>0100b</td> <td>Packet</td> </tr> <tr> <td>0101b</td> <td>FORCE_ENABLE_HS</td> </tr> <tr> <td>0110b</td> <td>FORCE_ENABLE_FS</td> </tr> <tr> <td>0111b</td> <td>FORCE_ENABLE_LS</td> </tr> <tr> <td>1000b to 1111b</td> <td>Reserved</td> </tr> </tbody> </table> <p>The FORCE_ENABLE_FS and FORCE_ENABLE_LS are extensions to the test mode support specified in the EHCI specification.</p> <p>Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point.</p> <p>Note: Low speed operations are not supported as a peripheral device.</p> | Value | Meaning | 0000b | TEST_MODE_DISABLE | 0001b | J_STATE | 0010b | K_STATE | 0011b | SE0 (host) / NAK (device) | 0100b | Packet | 0101b | FORCE_ENABLE_HS | 0110b | FORCE_ENABLE_FS | 0111b | FORCE_ENABLE_LS | 1000b to 1111b | Reserved |
| Value | Meaning | | | | | | | | | | | | | | | | | | | | | | | |
| 0000b | TEST_MODE_DISABLE | | | | | | | | | | | | | | | | | | | | | | | |
| 0001b | J_STATE | | | | | | | | | | | | | | | | | | | | | | | |
| 0010b | K_STATE | | | | | | | | | | | | | | | | | | | | | | | |
| 0011b | SE0 (host) / NAK (device) | | | | | | | | | | | | | | | | | | | | | | | |
| 0100b | Packet | | | | | | | | | | | | | | | | | | | | | | | |
| 0101b | FORCE_ENABLE_HS | | | | | | | | | | | | | | | | | | | | | | | |
| 0110b | FORCE_ENABLE_FS | | | | | | | | | | | | | | | | | | | | | | | |
| 0111b | FORCE_ENABLE_LS | | | | | | | | | | | | | | | | | | | | | | | |
| 1000b to 1111b | Reserved | | | | | | | | | | | | | | | | | | | | | | | |
| 15:1 4 | R/W | PIC | 0 | <p>Port Indicator Control</p> <p>Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. If P_INDICATOR bit is a one, then the bit is:</p> <table> <thead> <tr> <th>Bit Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>Port indicators are off</td> </tr> <tr> <td>01b</td> <td>Amber</td> </tr> <tr> <td>10b</td> <td>Green</td> </tr> <tr> <td>11b</td> <td>Undefined</td> </tr> </tbody> </table> <p>Refer to the USB Specification Revision 2.0 13 for a description on how these bits are to be used. This field is output from the controller as signals port_ind_ctl_1 & port_ind_ctl_0 for use by an external led driving circuit.</p> | Bit Value | Meaning | 00b | Port indicators are off | 01b | Amber | 10b | Green | 11b | Undefined | | | | | | | | | | |
| Bit Value | Meaning | | | | | | | | | | | | | | | | | | | | | | | |
| 00b | Port indicators are off | | | | | | | | | | | | | | | | | | | | | | | |
| 01b | Amber | | | | | | | | | | | | | | | | | | | | | | | |
| 10b | Green | | | | | | | | | | | | | | | | | | | | | | | |
| 11b | Undefined | | | | | | | | | | | | | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | |
|--------------|---|----------|-----------|--|--------------|--------------|-----|---|-----|---|-----|--------|-----|-----------|
| 13 | R | PO | 0 | <p>Port Owner</p> <p>Port owner hand off is not implemented in this design, therefore this bit will always read back as 0. The EHCI definition is include here for reference:</p> <p>This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a highspeed device). Software writes a one to this bit when the attached device is not a highspeed device. A one in this bit means that an internal companion controller owns and controls the port.</p> | | | | | | | | | | |
| 12 | R/W | PP | | <p>Port Power (PP)—Read/Write or Read Only. The function of this bit depends on the value of</p> <p>the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows:</p> <table> <tr> <td>PPC</td> <td>PP Operation</td> </tr> <tr> <td>0b</td> <td>0b Read Only— A device controller with no OTG capability does not have port power control switches.</td> </tr> <tr> <td>1b</td> <td>1b/0b – RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e. PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an overcurrent condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device only implementation port power control is not necessary, thus PPC and PP = 0.</td> </tr> </table> | PPC | PP Operation | 0b | 0b Read Only— A device controller with no OTG capability does not have port power control switches. | 1b | 1b/0b – RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e. PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an overcurrent condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device only implementation port power control is not necessary, thus PPC and PP = 0. | | | | |
| PPC | PP Operation | | | | | | | | | | | | | |
| 0b | 0b Read Only— A device controller with no OTG capability does not have port power control switches. | | | | | | | | | | | | | |
| 1b | 1b/0b – RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e. PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an overcurrent condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device only implementation port power control is not necessary, thus PPC and PP = 0. | | | | | | | | | | | | | |
| 11:10 | R | LS | 0 | <p>Line Status</p> <p>These bits reflect the current logical levels of the D+ (bit 11) and D (bit 10) signal lines. The encoding of the bits are:</p> <table> <tr> <td>Bits [11:10]</td> <td>]Meaning</td> </tr> <tr> <td>00b</td> <td>SE0</td> </tr> <tr> <td>10b</td> <td>Jstate</td> </tr> <tr> <td>01b</td> <td>Kstate</td> </tr> <tr> <td>11b</td> <td>Undefined</td> </tr> </table> <p>In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary.</p> | Bits [11:10] |]Meaning | 00b | SE0 | 10b | Jstate | 01b | Kstate | 11b | Undefined |
| Bits [11:10] |]Meaning | | | | | | | | | | | | | |
| 00b | SE0 | | | | | | | | | | | | | |
| 10b | Jstate | | | | | | | | | | | | | |
| 01b | Kstate | | | | | | | | | | | | | |
| 11b | Undefined | | | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description | | | | | | | | |
|------------------------------|---------------|-----------------|------------------|---|------------------------------|------------|----|---------|----|--------|----|---------|
| 9 | R | HSP | 0 | <p>High Speed Port</p> <p>When the bit is one, the host/device connected to the port is in highspeed mode and if set to zero, the host/device connected to the port is not in a highspeed mode. Note: HSP is redundant with PSPD(27:26) but will remain in the design for compatibility. This bit is not defined in the EHCI specification.</p> | | | | | | | | |
| 8 | R/W | PR | | <p>Port Reset</p> <p>This field is zero if Port Power(PP) is zero.</p> <p>In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. When software writes a one to this bit the busreset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.</p> <p>In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register.</p> | | | | | | | | |
| 7 | R/W | SUSP | | <p>Suspend</p> <p>In Host Mode: Read/Write. 1=Port in suspend state. 0=Port not in suspend state. Default=0. Port Enabled Bit and Suspend bit of this register define the port states as follows:</p> <table> <thead> <tr> <th>Bits [Port Enabled, Suspend]</th> <th>Port State</th> </tr> </thead> <tbody> <tr> <td>0x</td> <td>Disable</td> </tr> <tr> <td>10</td> <td>Enable</td> </tr> <tr> <td>11</td> <td>Suspend</td> </tr> </tbody> </table> <p>When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (i.e. Port enabled bit is a zero) the results are undefined. This field is zero if Port Power(PP) is zero in host mode.</p> <p>In Device Mode: Read Only. 1=Port in suspend state. 0=Port not in suspend state. Default=0. In device mode this bit is a read only status bit.</p> | Bits [Port Enabled, Suspend] | Port State | 0x | Disable | 10 | Enable | 11 | Suspend |
| Bits [Port Enabled, Suspend] | Port State | | | | | | | | | | | |
| 0x | Disable | | | | | | | | | | | |
| 10 | Enable | | | | | | | | | | | |
| 11 | Suspend | | | | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 6 | R/W | FPR | 0 | <p>Force Port Resume</p> <p>1= Resume detected/driven on port.</p> <p>0=No resume (K state) detected/driven on port.</p> <p>In Host Mode:</p> <p>Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a JtoK transition is detected while the port is in the Suspend state. When this bit transitions to a one because a JtoK transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver.</p> <p>Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full speed ‘K’) is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the highspeed idle. Writing a zero has no affect because the port controller will time the resume operation clear the bit the port control state switches to HS</p> <p>or FS idle. This field is zero if Port Power(PP) is zero in host mode.</p> <p>This bit is notEHCI compatible.</p> <p>In Device mode:</p> <p>After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J to K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J to K transition detected, the Port Change Detect bit in the USBSTS register is also set to one.</p> |
| 5 | R/WC | OCC | 0 | <p>Overcurrent Change</p> <p>1=This bit gets set to one when there is a change to Overcurrent Active. Software clears this bit by writing a one to this bit position. For host/OTG implementations the user can provide overcurrent detection to the vbus_pwr_fault input for this condition. For deviceonly implementations this bit shall always be 0.</p> |
| 4 | R | OCA | 0 | <p>Overcurrent Active</p> <p>1=This port currently has an overcurrent condition. 0=This port does not have an overcurrent condition. This bit will automatically transition from one to zero when the over current condition is removed. For host/OTG implementations the user can provide overcurrent detection to the vbus_pwr_fault input for this condition. For deviceonly implementations this bit shall always be 0.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 3 | R/WC | PEC | 0 | <p>Port Enable/Disable Change</p> <p>1=Port enabled/disabled status has changed. 0=No change. Default = 0.</p> <p>In Host Mode:</p> <p>For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it.</p> <p>This field is zero if Port Power(PP) is zero.</p> <p>In Device mode:</p> <p>The device port is always enabled. (This bit will be zero)</p> |
| 2 | R/W | PE | | <p>Port Enabled/Disabled</p> <p>1=Enable. 0=Disable. Default 0.</p> <p>In Host Mode:</p> <p>Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power(PP) is zero in host mode.</p> <p>In Device Mode:</p> <p>The device port is always enabled. (This bit will be one)</p> |
| 1 | R/WC | CSC | 0 | <p>Connect Status Change</p> <p>1 =Change in Current Connect Status.</p> <p>0=No change.</p> <p>In Host Mode:</p> <p>Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an alreadyset bit (i.e., the bit will remain set). Software clears this bit by writing a one to it.</p> <p>This field is zero if Port Power(PP) is zero in host mode.</p> <p>In Device Mode:</p> <p>This bit is undefined in device controller mode.</p> |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 0 | R | CCS | 0 | <p>Current Connect Status</p> <p>In Host Mode:</p> <p>1=Device is present on port. 0=No device is present. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power(PP) is zero in host mode.</p> <p>In Device Mode:</p> <p>1=Attached. 0=Not Attached. A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.</p> |

554 . PORTSCx Register Description

USB MODE REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 31:6 | R/W | Reserved. | | Reserved. |
| 5 | R/W | VBPS | 0x0 | <p>This bit is connected to the vbus_pwr_select output and can be used for any generic control</p> <p>but is named to be used by logic that selects between an onchip Vbus power source (charge pump) and an offchip source in systems when both are available.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 4 | R/W | SDIS | 0x0 | <p>Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode, when enabled, ensures that the RX and TX buffers are sufficient to contain an entire packet, so the usual double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active.</p> <p>Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB.</p> <p>Note: Time duration to prefill the FIFO becomes significant when stream disable is active.</p> <p>See TXFILLTUNING and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature.</p> |
| 3 | R/W | SLOM | 0x0 | <p>R/W Setup Lockout Mode.</p> <p>In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model.</p> <p>0 – Setup Lockouts On (default); 1 – Setup Lockouts Off (DCD requires use of Setup Data)</p> <p>Buffer Tripwire in USBCMD)</p> |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | |
|-----|--|----------|-----------|--|-----|---------|----|--|----|--|----|--|----|--|
| 2 | R/W | ES | 0x0 | <p>Endian Select</p> <p>This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32bit words.</p> <table> <thead> <tr> <th>Bit</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>Little Endian [Default] – first byte referenced in least significant byte of 32bit word.</td></tr> <tr> <td>1</td><td>Big Endian first byte referenced in most significant byte of 32bit word.</td></tr> </tbody> </table> | Bit | Meaning | 0 | Little Endian [Default] – first byte referenced in least significant byte of 32bit word. | 1 | Big Endian first byte referenced in most significant byte of 32bit word. | | | | |
| Bit | Meaning | | | | | | | | | | | | | |
| 0 | Little Endian [Default] – first byte referenced in least significant byte of 32bit word. | | | | | | | | | | | | | |
| 1 | Big Endian first byte referenced in most significant byte of 32bit word. | | | | | | | | | | | | | |
| 1:0 | R/W | CM | 0x0 | <p>Controller Mode – Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register.</p> <table> <thead> <tr> <th>Bit</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>00</td><td>Idle [Default for combination host/device]</td></tr> <tr> <td>01</td><td>Reserved</td></tr> <tr> <td>10</td><td>Device Controller [Default for device only controller]</td></tr> <tr> <td>11</td><td>Host Controller [Default for host only controller]</td></tr> </tbody> </table> | Bit | Meaning | 00 | Idle [Default for combination host/device] | 01 | Reserved | 10 | Device Controller [Default for device only controller] | 11 | Host Controller [Default for host only controller] |
| Bit | Meaning | | | | | | | | | | | | | |
| 00 | Idle [Default for combination host/device] | | | | | | | | | | | | | |
| 01 | Reserved | | | | | | | | | | | | | |
| 10 | Device Controller [Default for device only controller] | | | | | | | | | | | | | |
| 11 | Host Controller [Default for host only controller] | | | | | | | | | | | | | |

555 . USBMODE Register Description

USB ENDPOINT SETUP STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|-----------|--------|----------|-----------|-------------|
| 31:1 6 | R/W | Reserved | 0 | Reserved. |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|--|
| 15:0 | R/W | ENDPTSETUPSTAT | 0 | <p>Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one.</p> <p>Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head.</p> <p>The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock out mechanism is engaged. See Managing Endpoints in the Device Operational Model.</p> <p>This register is only used in device mode.</p> |

556 .ENDPTSETUPSTAT Register Description

USB ENDPOINT PRIME REGISTER

| Bit | Access | Function | POR Value | Description | | | | | | |
|-----------|----------------|----------|-----------|--|----------|----------------|---------|---------------|---------|---------------|
| 31:1 6 | R/W | PETB | 0 | <p>Prime Endpoint Transmit Buffer</p> <p>For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.</p> <p>Note: These bits will be momentarily set by hardware during hardware repriming operations when a dTD is retired, and the dQH is updated.</p> <table> <tr> <td>PETB[15]</td> <td>– Endpoint #15</td> </tr> <tr> <td>PETB[1]</td> <td>– Endpoint #1</td> </tr> <tr> <td>PETB[0]</td> <td>– Endpoint #0</td> </tr> </table> | PETB[15] | – Endpoint #15 | PETB[1] | – Endpoint #1 | PETB[0] | – Endpoint #0 |
| PETB[15] | – Endpoint #15 | | | | | | | | | |
| PETB[1] | – Endpoint #1 | | | | | | | | | |
| PETB[0] | – Endpoint #0 | | | | | | | | | |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 15:0 | R/W | PERB | 0 | <p>Prime Endpoint Receive Buffer</p> <p>For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.</p> <p>Note: These bits will be momentarily set by hardware during hardware repriming operations when a dTD is retired, and the dQH is updated.</p> <p>Bit 15 – Endpoint #15</p> <p>Bit 1 – Endpoint #1</p> <p>Bit 0 – Endpoint #0</p> |

557 .ENDPTPRIME Register Description

USB ENDPOINT FLUSH REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:16 | R/W | FETB | 0 | <p>Flush Endpoint Transmit Buffer</p> <p>Writing a one to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers.</p> <p>If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion.</p> <p>Hardware will clear this register after the endpoint flush operation is successful.</p> <p>FETB[15] – Endpoint #15</p> <p>FETB[1] – Endpoint #1</p> <p>FETB[0] – Endpoint #0</p> |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 15:0 | R/W | FERB | 0 | <p>Flush Endpoint Receive Buffer</p> <p>Writing a one to a bit(s) will cause the associated endpoint(s) to clear any primed buffers.</p> <p>If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion.</p> <p>Hardware will clear this register after the endpoint flush operation is successful.</p> <p>Bit 15 – Endpoint #15</p> <p>Bit 1 – Endpoint #1</p> <p>Bit 0 – Endpoint #0</p> |

558 . ENDPTFLUSH Register Description

USB ENDPOINT STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:16 | R | ETBR | 0 | <p>Endpoint Transmit Buffer Ready</p> <p>One bit for each endpoint indicates status of the respective endpoint buffer.</p> <p>This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register.</p> <p>There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready.</p> <p>This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register.</p> <p>Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.</p> <p>ETBR[15] – Endpoint #15</p> <p>ETBR[1] – Endpoint #1</p> <p>ETBR[0] – Endpoint #0</p> |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 15:0 | R | ERBR | 0 | <p>Endpoint Receive Buffer Ready</p> <p>One bit for each endpoint indicates status of the respective endpoint buffer.</p> <p>This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register.</p> <p>There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready.</p> <p>This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register.</p> <p>Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register</p> <p>ERBR[15] – Endpoint #15 ERBR[1] – Endpoint #1 ERBR[0] – Endpoint #0</p> |

559 .ENDPTSTAT Register Description

USB ENDPOINT COMPLETE REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:16 | R/WC | ETCE | 0 | <p>Endpoint Transmit Complete Event</p> <p>Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status.</p> <p>If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register.</p> <p>ETCE[15] – Endpoint #15 ETCE[1] – Endpoint #1 ETCE[0] – Endpoint #0</p> |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 15:0 | R/WC | ERCE | 0 | <p>Endpoint Receive Complete Event</p> <p>Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status.</p> <p>If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register.</p> <p>ERCE[15] – Endpoint #15 ERCE[1] – Endpoint #1 ERCE[0] – Endpoint #0</p> |

560 .ENDPTCOMPLETE Register Description

USB ENDPOINT CONTROL ADDRESS REGISTER 0

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:24 | R/W | Reserved | | Reserved |
| 23 | R/W | TXE | 0x1 | <p>TX Endpoint Enable</p> <p>1 – Enabled</p> <p>Endpoint0 is always enabled.</p> |
| 22:20 | R/W | Reserved | | Reserved |
| 19:18 | R/W | TXT | 0x0 | <p>TX Endpoint Type – Read/Write</p> <p>00 – Control</p> <p>Endpoint0 is fixed as a Control End Point.</p> |
| 17 | R/W | Reserved | | Reserved |
| 16 | R/W | TXS | 0x0 | <p>TX Endpoint Stall – Read/Write</p> <p>0 – End Point OK [Default]</p> <p>1 – End Point Stalled</p> |
| 15:8 | R/W | Reserved | | Reserved |
| 7 | R/W | RXE | 0x0 | <p>RX Endpoint Enable</p> <p>1 – Enabled</p> <p>Endpoint0 is always enabled.</p> |
| 6:4 | R/W | Reserved | | Reserved |
| 3:2 | R/W | RXT | 0x2 | <p>RX Endpoint Type – Read/Write</p> <p>00 – Control</p> <p>Endpoint0 is fixed as a Control End Point.</p> |
| 1 | R/W | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 0 | R/W | RXS | 0x0 | RX Endpoint Stall – Read/Write 0 – End Point OK. [Default] 1 – End Point Stalled |

561 . ENDPTCTRL0 Register Description

USB ENDPOINT CONTROL ADDRESS REGISTER n

There is an ENDPTCTRLx register for each endpoint in a device.x = 1 to 15.

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:24 | R/W | Reserved | | Reserved |
| 23 | R/W | TXE | 0x0 | TX Endpoint Enable 0 – Disabled 1 – Enabled An Endpoint should be enabled only after it has been configured. |
| 22 | R/W | TXR | | TX Data Toggle Reset (WS) Write 1 – Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device. |
| 21 | R/W | TXI | 0x0 | TX Data Toggle Inhibit 0 – PID Sequencing Enabled. 1 – PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. |
| 20 | R | Reserved | | Reserved.Bit reserved and should be set to zero. |
| 19:18 | R/W | TXT | 0x0 | TX Endpoint Type 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 17 | R/W | TXD | 0x0 | <p>TX Endpoint Data Source</p> <p>0 – Dual Port Memory Buffer/DMA Engine</p> <p>Should always be written as 0.</p> |
| 16 | R/W | TXS | 0x0 | <p>TX Endpoint Stall</p> <p>0 – End Point OK</p> <p>1 – End Point Stalled</p> <p>This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and</p> <p>this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.</p> <p>Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to</p> <p>STALL until this bit is either cleared by software or automatically cleared as above for control endpoints.</p> |
| 15:8 | R/W | Reserved | | Reserved |
| 7 | R/W | RXE | 0x0 | <p>RX Endpoint Enable</p> <p>0 – Disabled</p> <p>1 – Enabled</p> <p>An Endpoint should be enabled only after it has been configured.</p> |
| 6 | R/W | RXR | 0x0 | <p>RX Data Toggle Reset (WS)</p> <p>Write 1 – Reset PID Sequence</p> <p>Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data</p> <p>PID's between the host and device.</p> |
| 5 | R/W | RXI | 0x0 | <p>RX Data Toggle Inhibit</p> <p>0 – Disabled</p> <p>1 – Enabled</p> <p>This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the</p> <p>data toggle sequence and always accept data packet regardless of their data PID.</p> |
| 4 | R/W | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 3:2 | R/W | RXT | 0x0 | RX Endpoint Type 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt |
| 1 | R/W | RXD | 0x0 | RX Endpoint Data Sink 0 – Dual Port Memory Buffer/DMA Engine Should always be written as zero. |
| 0 | R/W | RXS | 0x0 | RX Endpoint Stall 0 – End Point OK. 1 – End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. |

562 . ENDPTCTRL1-15 Register Description

DWords 36 are used to manage the state of the transfer.

Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

16 MCU APB Peripherals

16.1 General Description

MCU peripheral's AHB to AHB bridge have two AHB slaves. Any Peripheral can be accessed from two AHB masters : MCU AHB master, GPDMA AHB master. Thus two different peripherals can be independently accessed by MCU and GPDMA at the same time. When both the master interfaces access the same peripheral, MCU master will be given preference and its transaction will happen first followed by the DMA's transaction.

Following peripherals are integrated in the MCU.

- UART
- Enhanced GPIO (EGPIO)
- Motor Control PWM
- MCU eFuse controller
- Config Timers
- Quadrature Encoder
- Serial GPIO (SIO)
- CAN controller
- USART
- Hardware Random Number Generator (HRNG)
- CRC accelerator
- SSI Slave
- SSL Master
- General SPI Master
- I2S/PCM Slave and Master
- I2C Master and Slave

MCU Peripherals are divided into 4 domains according to power architecture. MCU Peripherals Address Map Table is shown in the following table

| Peripheral | Base Address |
|------------------------------|--------------|
| Peripheral-1 Domain | |
| UART1 | 4400_0000 |
| USART1 | 4400_0100 |
| I2C1 Master and Slave | 4401_0000 |
| SSI Master | 4402_0000 |
| uDMA Configuration registers | 4403_0000 |
| Peripheral-2 Domain | |
| SSI Slave | 4501_0000 |
| UART2 | 4502_0000 |
| General SPI Master | 4503_0000 |
| Config Timers | 4506_0000 |
| CAN Controller | 4507_0000 |

| | |
|--|-----------|
| CRC Accelerator | 4508_0000 |
| HRNG | 4509_0000 |
| Peripheral-3 (always-on) Domain | |
| VIC | 4611_0000 |
| ROM PATCH | 4612_0200 |
| EGPIO | 4613_0000 |
| CCI Configuration registers | 4617_0000 |
| RF Register Access SPI | 4618_0000 |
| PMU Configuration registers | 4600_0000 |
| PAD Configuration registers | 4600_4000 |
| MCU Configuration registers | 4600_8000 |
| MCU eFUSE Controller | 4600_C000 |
| Peripheral-4 Domain | |
| Serial GPIO | 4700_0000 |
| I2C2 Master and Slave | 4704_0000 |
| I2S/PCM Master and Slave | 4705_0000 |
| Quadrature Encoder | 4706_0000 |
| Motor Control PWM | 4707_0000 |

563 . MCU Peripherals Address Map Table

16.2 CAN Controller

16.2.1 General Description

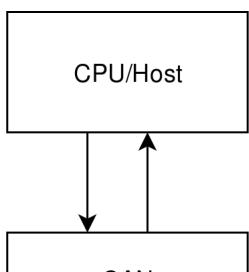
The Controller Area Network (CAN) is an advanced serial communications protocol that efficiently supports distributed control systems. This CAN controller is widely used in automotive and industrial applications. This CAN Controller conforms to Bosch CAN 2.0B specification (2.0B Active). The CAN protocol uses Data Link Layer and the Physical Layer in the ISO-OSI model.

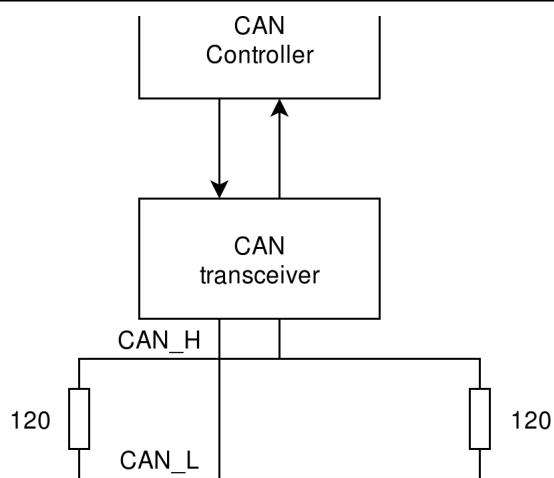
Maximum data transfer rate is 1Mbps at maximum 40m bus length when using a twisted wire pair.

Message length is short with a maximum of 8 data

bytes per message and there is a low latency between transmission request and start of transmission. The bus is handled with Carrier Sense Multiple Access / Collision Detection with Non

Destructive Arbitration. This means that collision of messages is avoided by bit-wise arbitration without loss of time. The CAN controller requires an external CAN bus transceiver that interfaces to 2-wire CAN bus as shown in the diagram below.





CAN system

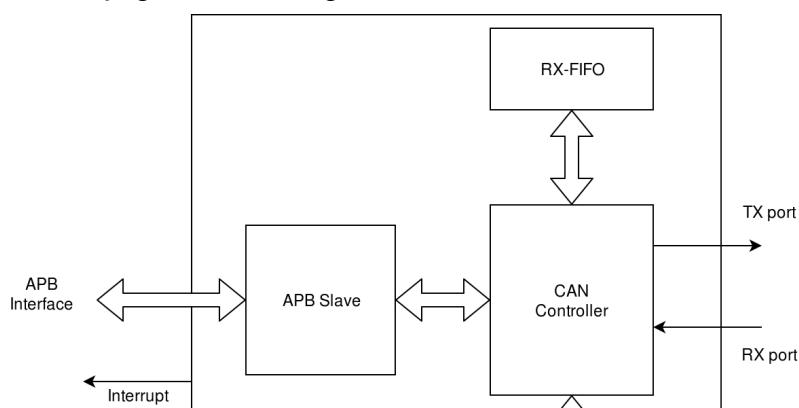
16.2.2 Features

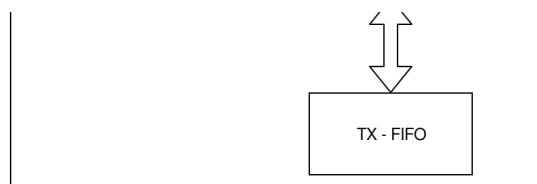
- Conforms to Bosch CAN 2.0B specification
- 11- and 29-bit wide message identifiers
- Data rate up to 1 Mbps
- Hardware message filtering (dual/single filters)
- 64-byte receive FIFO
- 16-byte transmit buffer
- Overload frame is generated on FIFO overflow
- Normal & Listen Only modes supported
- Single Shot transmission
- Ability to abort transmission
- Readable error counters
- Last Error Code
- Programmable clock frequency

16.2.3 Functional Description

The CAN controller has simple 32-bit CPU interface (APB bus). It has Hardware message filtering, 16 bytes transmit FIFO and 64 byte receive FIFO, which enables back-to-back message reception with minimum CPU load. CAN bus uses multi-master bus scheme with one logic bus line and equal nodes. The number of nodes is not limited by the protocol. Nodes do not have specific addresses. Instead, message identifiers are used, indicating the message content and priority of message. This also means that multicasting and broadcasting is supported by CAN. Number of nodes may be changed at run-time without disturbing the communication of the other nodes.

The CAN protocol uses Non-Return-To-Zero (NRZ) coding. For synchronization purposes, Bit Stuffing is used. CAN provides sophisticated error detection and error handling mechanisms and, due to differential transmission, high immunity against electromagnetic interference.



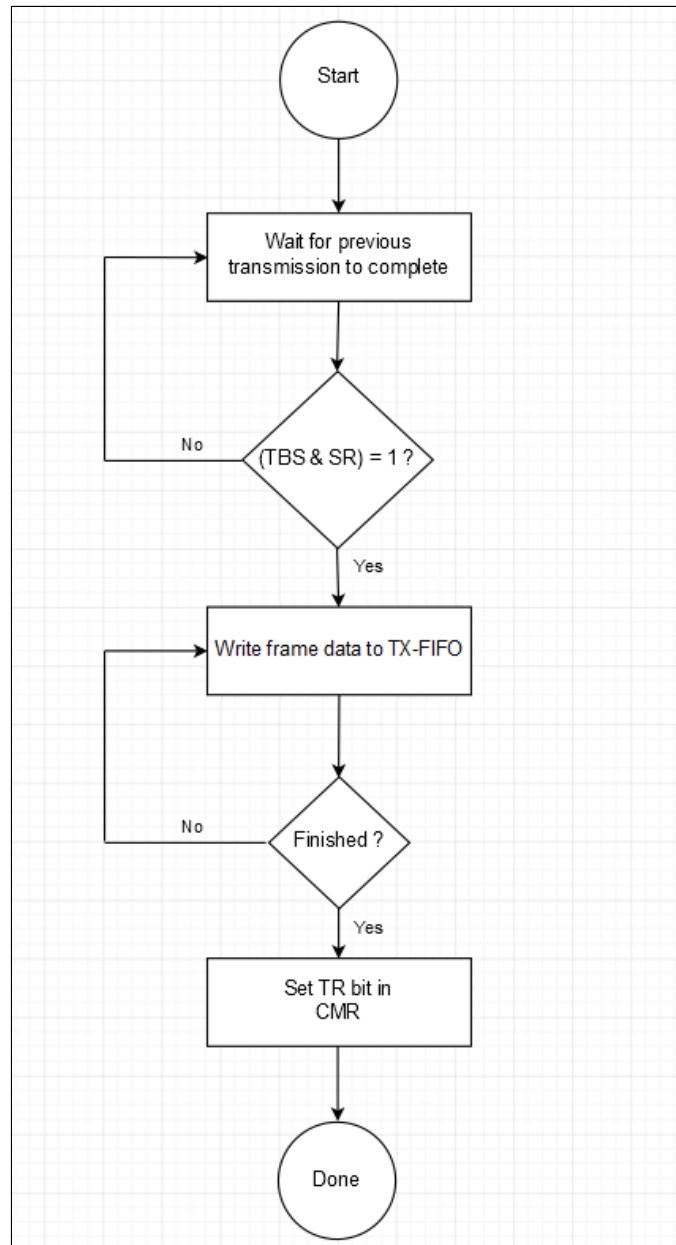


CAN Controller Block diagram

Transmit and Receive Flowcharts

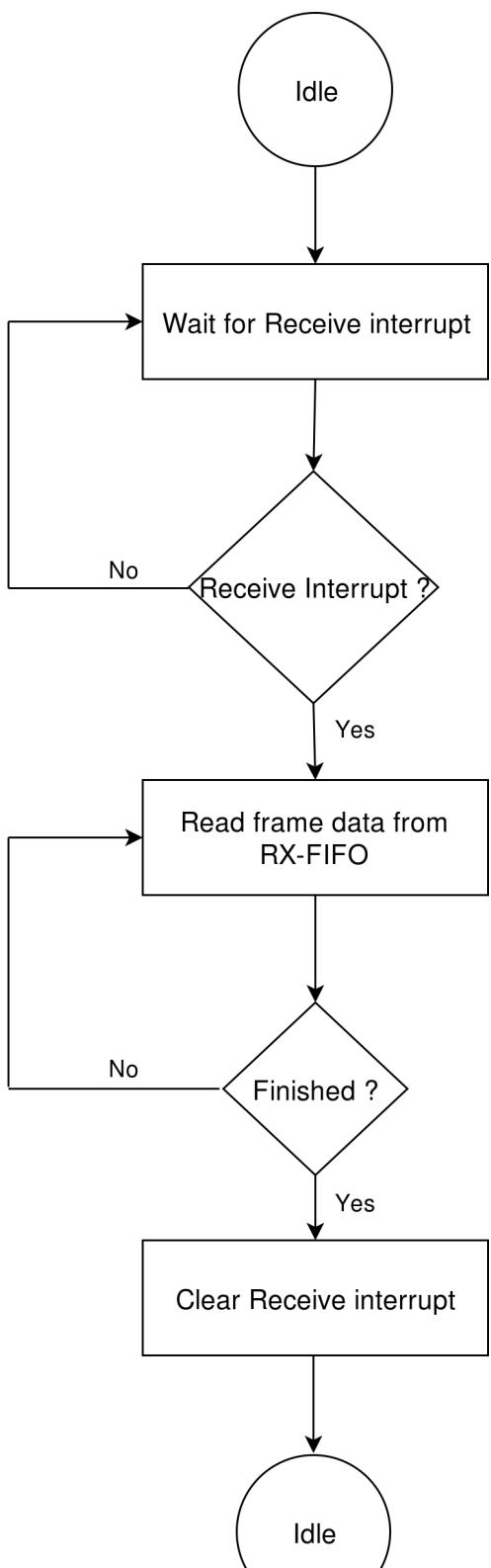
Following flowcharts describe typical flow for sending and receiving frames.

Transmit Flowchart



16 . Transmit Flowchart

Receive Flowchart

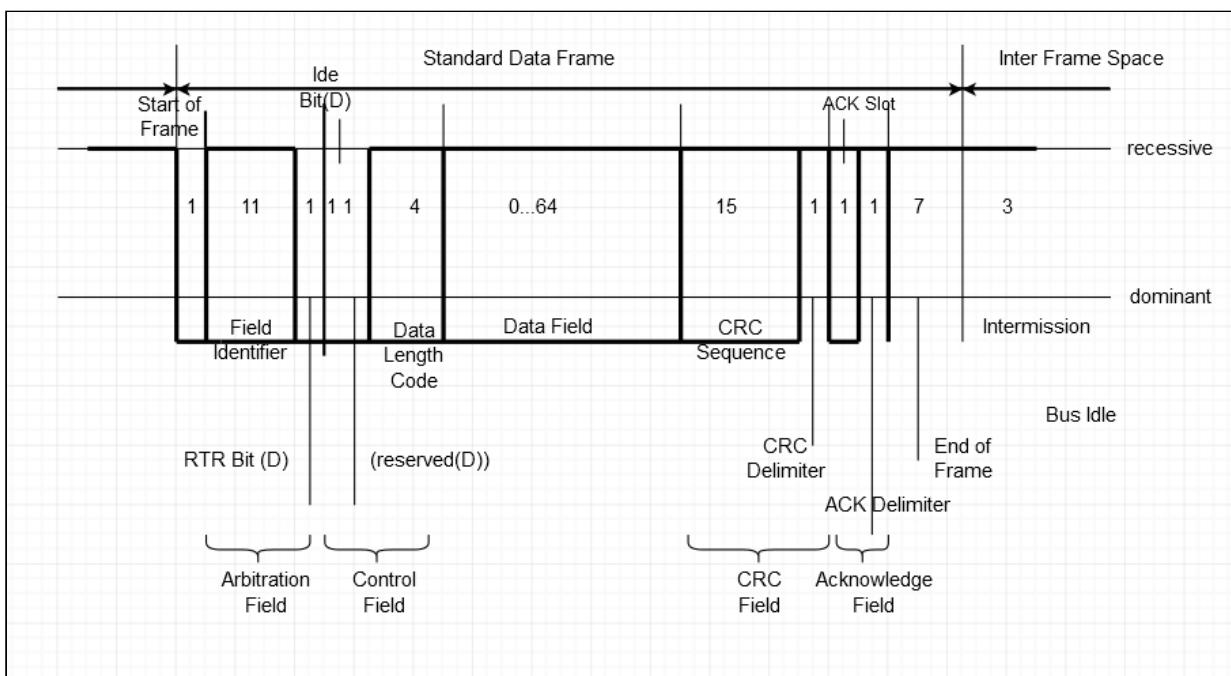


Receive Flowchart

There are two different formats which differ in the length of the IDENTIFIER field: Frames with the number of 11-bit IDENTIFIER are denoted Standard Frames. In contrast, frames containing 29-bit IDENTIFIER are denoted Extended Frames. Message transfer is manifested and controlled by four different frame types: A DATA FRAME carries data from a transmitter to the receivers. A REMOTE FRAME is transmitted by a bus unit to request the transmission of the DATA FRAME with the same IDENTIFIER. An ERROR FRAME is transmitted by any unit on detecting a bus error. An OVERLOAD FRAME is used to provide for an extra delay between the preceding and the succeeding DATA or REMOTE FRAMES. DATA FRAMEs and REMOTE FRAMEs can be used in both Standard Frame Format and Extended Frame Format; they are separated from preceding frames by an INTER FRAME SPACE.

A Data Frame is generated by a CAN node when the node wishes to transmit data. The Standard CAN Data Frame is shown below. A DATA FRAME is composed of seven different bit fields:

- Start of Frame - the frame begins with a dominant Start Of Frame bit for hard synchronization of all nodes.
- Arbitration Field - the Start of Frame bit is followed by the Arbitration Field consisting of 12 bits: the 11-bit identifier, which reflects the contents and priority of the message, and the Remote Transmission Request bit. This bit is used to distinguish a Data Frame (RTR = dominant) from a Remote Frame (RTR = recessive).
- Control Field - consisting of 6 bits. The first bit of this field is called the IDE bit (Identifier Extension) and is at dominant state to specify that the frame is a Standard Frame. The following bit is reserved and defined as a dominant bit. The remaining 4 bytes of the Control Field are the Data Length Code (DLC) and specify the number of bytes of data contained in the message (0-8 bytes).
- Data Field - the data is being sent in the Data Field which is of the length defined by the DLC above (0, 8, 16, ..., 56 or 64 bits).
- Cyclic Redundancy Field (CRC) - is used to detect possible transmission errors. The CRC Field consists of a 15 bit CRC sequence, completed by the recessive CRC Delimiter bit.
- Acknowledge Field (ACK) - during the ACK bit the transmitting node sends out a recessive bit. Any node that has received an error free frame acknowledges the correct reception of the frame by sending back a dominant bit (regardless of whether the node is configured to accept that specific message or not). From this it can be seen that CAN belongs to the “in-bit-response” group of protocols. The recessive Acknowledge Delimiter completes the Acknowledge Slot and may not be overwritten by a dominant bit.
- End of Frame - seven recessive bits ends the Data Frame.



17 . Standard Frame Format

16.2.4 Register Summary

Base Address: 0x4507_0000

| Register Name | Offset | Description |
|---------------|--------|--|
| CAN_MR | 0x00 | CAN controller mode register |
| CAN_CMR | 0x01 | CAN controller Command register |
| CAN_SR | 0x02 | CAN controller Status register |
| CAN_ISR/IACK | 0x03 | Interrupt status and acknowledgment register |
| CAN_IMR | 0x04 | Interrupt Mask register |
| CAN_RMC | 0x05 | Receive Message Counter register |
| CAN_BTIM0 | 0x06 | Bus Timing Register 0 register |
| CAN_BTIM1 | 0x07 | Bus Timing Register 1 register |
| CAN_TXBUF | 0x08 | Transmit Buffer (TX FIFO) register |
| CAN_RXBUF | 0x0C | Receive Buffer (RX FIFO) register |
| CAN_ACR0 | 0x10 | Acceptance Code 0 register |
| CAN_ACR1 | 0x11 | Acceptance Code 1 register |
| CAN_ACR2 | 0x12 | Acceptance Code 2 register |
| CAN_ACR3 | 0x13 | Acceptance Code 3 register |
| CAN_AMR0 | 0x14 | Acceptance Mask 0 register |
| CAN_AMR1 | 0x15 | Acceptance Mask 1 register |
| CAN_AMR2 | 0x16 | Acceptance Mask 2 register |
| CAN_AMR3 | 0x17 | Acceptance Mask 3 register |
| CAN_ECC | 0x18 | Error Code Capture register |
| CAN_RXERR | 0x19 | RX Error Counter register |
| CAN_TXERR | 0x1A | TX Error Counter register |

| Register Name | Offset | Description |
|---------------|--------|--|
| CAN_ALC | 0x1B | Arbitration Lost Code Capture register |

564 . Register Summary

16.2.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, - = Reserved

CAN MODE REGISTER (MR)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 7:3 | - | - | 0 | Reserved |
| 2 | R/W | RM | 1 | Reset Mode |
| 1 | R/W | LOM | 0 | 1- Listen Only Mode 0 - Normal Mode These modes can be set by writing either '1 or 0' to LOM and '0' to RM while in reset mode |
| 0 | R/W | AFM | 0 | hardware acceptance filter scheme 0 - dual filter is used 1 - single filter is used This bit can be written only while in reset mode |

565 . Mode Register Description

CAN COMMAND REGISTER (CMR)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|--------------------|
| 7:4 | - | - | 0 | Reserved |
| 2 | W | TR | 0 | Transmit Request |
| 1 | W | AT | 0 | Abort Transmission |
| 0 | - | - | 0 | Reserved |

566 . Command Register Description

CAN STATUS REGISTER (SR)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 7 | R | RBS | 0 | Receive Buffer Status 1 - means at least one message is in FIFO 0 - means no messages are in FIFO |

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|--|
| 6 | R | DSO | 0 | Data Overrun Status 1 - means RX FIFO encounters overrun 0 - means no overrun occurred since last clear data overrun command |
| 5 | R | TBS | 0 | Transmit Buffer Status 1 - transmit buffer is released for CPU 0 - transmit buffer is locked for CPU(transmission is pending or message is being transmitted). written data is not accepted. |
| 4 | - | - | 0 | Reserved |
| 3 | R | RS | 0 | Receive Status 1 - means that CAN core is receiving a message |
| 2 | R | TS | 0 | Transmit Status 1 - means that CAN core is transmitting a message |
| 1 | R | ES | 1 | Error Status 1 - means that at least one of CAN error counters reached error warning limit (96) |
| 0 | R | BS | 0 | Bus Off Status. 1- node is in bus off state and cannot transmit and receive frames 0 - frame reception and transmission is possible |

567 . Status Register Description

CAN INTERRUPT STATUS / ACKNOWLEDGE REGISTER (ISR)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|--|
| 7 | - | - | 0 | Reserved |
| 6 | R/W | ALI | 0 | Arbitration Lost Interrupt |
| 5 | R/W | EWI | 0 | Error Warning Interrupt |
| 4 | R/W | EPI | 0 | Error Passive Interrupt |
| 3 | R/W | RI | 0 | Receive Interrupt |
| 2 | R/W | TI | 0 | Transmission Interrupt |
| 1 | R/W | BEI | 0 | Bus Error Interrupt |
| 0 | R/W | DOI | 0 | Data Overrun Interrupt 1 - clears DOI interrupt |

568 . Interrupt/Acknowledge Register Description

CAN INTERRUPT MASK REGISTER (IMR)

| Bit | Access | Name | POR Value | Description |
|-----|--------|-------|-----------|------------------------|
| 7 | - | - | 0 | Reserved |
| 6 | R/W | ALIM | 0 | mask for ALI interrupt |
| 5 | R/W | EWIM | 0 | mask for EWI interrupt |
| 4 | R/W | EPIIM | 0 | mask for EPI interrupt |
| 3 | R/W | RIM | 0 | mask for RI interrupt |
| 2 | R/W | TIM | 0 | mask for TI interrupt |
| 1 | R/W | BEIM | 0 | mask for BEI interrupt |
| 0 | R/W | DOIIM | 0 | mask for DOI interrupt |

569 . Interrup Mask Register Description

CAN RECEIVE MESSAGE COUNTER (RMC)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 7:5 | - | - | 0 | Reserved |
| 4:0 | R | RMC | 0 | number of stored message frames in the receive FIFO. The value is incremented on each successful frame reception and decremented by clearing RI interrupt. |

570 . Receive Message Counter Register Description

CAN BUS TIMING REGISTER 0 (BTIM0)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|----------------------------|
| 7:6 | R/W* | SJW | 0 | Synchronization Jump Width |
| 5:0 | R/W* | BRP | 0 | Baud Rate Prescaler |

W* - Register can be written only in reset mode

571 . BUS Timing 0 Register Description

CAN BUS TIMING REGISTER 1 (BTIM1)

| Bit | Access | Name | POR Value | Description |
|-----|--------|-------|-----------|--|
| 7 | R/W* | SAM | 0 | Number of bus level samples 0 - bus level is sampled once 1 - bus level is sampled three times |
| 6:4 | R/W* | TSEG2 | 0 | Number of clock cycles per Time Segment 2 |
| 3:0 | R/W* | TSEG1 | 0 | Number of clock cycles per Time Segment 1 |

W* - Register can be written only in reset mode

572 . BUS Timing 1 Register Description

CAN TRANSMIT BUFFER REGISTER (TBR)

| Bit | Access | Name | POR Value | Description |
|-------|--------|--------|-----------|--|
| 31:24 | W | TXBUF3 | 0 | most significant.frame byte n+3. Address -0x0B |
| 23:16 | W | TXBUF2 | 0 | frame byte n+2. Address -0x0A |
| 15:8 | W | TXBUF1 | 0 | frame byte n+1. Address -0x09 |
| 7:0 | W | TXBUF0 | 0 | least significant . frame byte n . Address -0x08 |

573 . Transmit Buffer Register Description

CAN RECEIVE BUFFER REGISTER (RBR)

| Bit | Access | Name | POR Value | Description |
|-------|--------|--------|-----------|--|
| 31:24 | R | RXBUF3 | 0 | most significant.frame byte n+3. Address -0x0F |
| 23:16 | R | RXBUF2 | 0 | frame byte n+2. Address -0x0E |
| 15:8 | R | RXBUF1 | 0 | frame byte n+1. Address -0x0D |
| 7:0 | R | RXBUF0 | 0 | least significant . frame byte n . Address -0x0C |

574 . Receive Buffer Register Description

CAN ACCEPTANCE CODE REGISTER 0 (ACR0)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|--|
| 7:0 | R/W | ACR0 | 0 | contains bit patterns of messages to be received |

575 . Acceptance Code 0 Register Description

CAN ACCEPTANCE CODE REGISTER 1 (ACR1)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|--|
| 7:0 | R/W | ACR1 | 0 | contains bit patterns of messages to be received |

576 . Acceptance Code 1 Register Description

CAN ACCEPTANCE CODE REGISTER 2 (ACR2)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|--|
| 7:0 | R/W | ACR2 | 0 | contains bit patterns of messages to be received |

577 . Acceptance Code 2 Register Description

CAN ACCEPTANCE CODE REGISTER 3 (ACR3)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|--|
| 7:0 | R/W | ACR3 | 0 | contains bit patterns of messages to be received |

578 . Acceptance Code 3 Register Description

CAN ACCEPTANCE MASK REGISTER 0 (AMR0)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 7:0 | R/W | AMR0 | 0 | contains which bit positions will be compared and which ones are don't care in ACR0 |

579 .Acceptance Mask 0 Register Description

CAN ACCEPTANCE MASK REGISTER 1 (AMR1)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 7:0 | R/W | AMR1 | 0 | contains which bit positions will be compared and which ones are don't care in ACR1 |

580 .Acceptance Mask 1 Register Description

CAN ACCEPTANCE MASK REGISTER 2 (AMR2)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 7:0 | R/W | AMR2 | 0 | contains which bit positions will be compared and which ones are don't care in ACR2 |

581 .Acceptance Mask 2 Register Description

CAN ACCEPTANCE MASK REGISTER 3 (AMR3)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 7:0 | R/W | AMR3 | 0 | contains which bit positions will be compared and which ones are don't care in ACR3 |

582 .Acceptance Mask 3 Register Description

CAN RECEIVE ERROR COUNTER REGISTER (RXERR)

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 7:0 | R | RXERR | 0 | Reflects current value of the receive error counter. If a bus off event occurs, RX error counter is initialized to 0. |

583 .Receive Error Counter Register Description

CAN TRANSMIT ERROR COUNTER REGISTER (TXERR)

| Bit | Access | Name | POR Value | Description |
|-----|--------|-------|-----------|---|
| 7:0 | R | TXERR | 0 | Reflects current value of the transmit error counter. If a bus off event occurs, TX error counter is initialized to 127. |

584 .Transmit Error Counter Register Description

CAN ERROR CODE CAPTURE REGISTER (ECC)

| Bit | Access | Name | POR Value | Description |
|-----|--------|-------|-----------|---|
| 7 | R | RXWRN | 0 | set when RXERR counter is greater than or equal to 96 |
| 6 | R | TXWRN | 0 | set when TXERR counter is greater than or equal to 96 |
| 5 | R | EDIR | 0 | direction of transfer while error occurred. 0 - transmission, 1 - reception |
| 4 | R | ACKER | 0 | ACK error occurred |
| 3 | R | FRMER | 0 | form error occurred |
| 2 | R | CRCER | 0 | CRC error occurred |
| 1 | R | STFER | 0 | stuff error occurred |
| 0 | R | BER | 0 | bit error occurred |

585 . Error Code Capture Register Description

CAN ARBITRATION LOST CODE CAPTURE REGISTER (ALC)

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|-------------|
| 7:5 | - | - | 0 | Reserved |

| Bit | Access | Name | POR Value | Description |
|-----|--------|------|-----------|---|
| 4:0 | R | ALC | 0 | 00 - Arbitration lost in ID28 / 10 01 - Arbitration lost in ID27 / 9 02 - Arbitration lost in ID26 / 8 03 - Arbitration lost in ID25 / 7 04 - Arbitration lost in ID24 / 6 05 - Arbitration lost in ID23 / 5 06 - Arbitration lost in ID22 / 4 07 - Arbitration lost in ID21 / 3 08 - Arbitration lost in ID20 / 2 09 - Arbitration lost in ID19 / 1 10 - Arbitration lost in ID18 / 0 11 - Arbitration lost in SRTR / RTR 12 - Arbitration lost in IDE bit 13 - Arbitration lost in ID17* 14 - Arbitration lost in ID16* 15 - Arbitration lost in ID15* 16 - Arbitration lost in ID14* 17 - Arbitration lost in ID13* 18 - Arbitration lost in ID12* 19 - Arbitration lost in ID11* 20 - Arbitration lost in ID10* 21 - Arbitration lost in ID9* 22 - Arbitration lost in ID8* 23 - Arbitration lost in ID7* 24 - Arbitration lost in ID6* 25 - Arbitration lost in ID5* 26 - Arbitration lost in ID4* 27 - Arbitration lost in ID3* 28 - Arbitration lost in ID2* 29 - Arbitration lost in ID1* 30 - Arbitration lost in ID0* 31 - Arbitration lost in RTR *-extended frame messages only |

586 .Arbitration Lost Code Capture Register Description

16.3 Configurable Timers

16.3.1 General Description

Configurable timers are used for counting clocks and events, capturing events on the GPIOs in input mode and outputting modulated signals. They can be programmed to work in Pulse Width Modulation (PWM) mode in which a pulse width modulated wave is driven on the outputs according to the programmed ON time and OFF times. Configurable Timers are present in MCU HP peripherals.

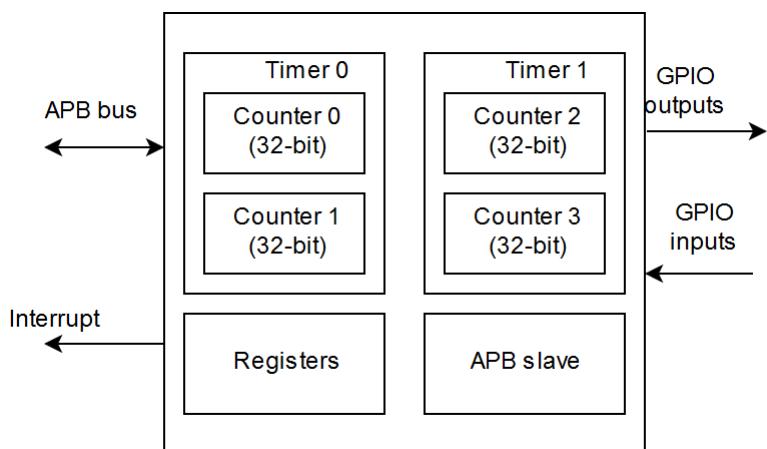
16.3.2 Features

The key features of the Configurable Timers are listed below:

- Supports four 32-bit timers - each 32-bit timer can be configured to contain one 32-bit or two 16-bit timers.
- The timers can be programmed to select clocks or events as ticks.
- Supports up counters, down counters and up-down counters
- Supports 8 inputs and 8 outputs.
- The inputs are used to trigger events like starting, stopping, continuing, increment, decrement or halting the counter, capture counter values, generate interrupt, trigger DMA and control output signals.
- The input triggers can be rising/falling edges, high/low levels, operations like AND/OR on combinations of inputs.
- The output signals can be used to either output modulated signals or can be toggled after a programmable duration once a trigger is generated using the inputs.
- Support for PWM signals as outputs with any cycle/pulse length and superimposing of a waveform on the PWM signal.
- The timers can be used to trigger the DAC at any time in sync with the output PWM signal
- Support total 39 events on input signals.
- Support for DMA flow control
- Generates interrupt for different events.

16.3.3 Functional Description

The CT contain four 32-bit configurable counters as shown in Figure1. The CT communicate with external devices through the GPIOs. Each configuration counter contains independent register set for controlling and accessing internal timers. Four registers are provided to program the selection of the DMA flow control signals for the Output Compare Unit (OCU) mode. Another register is also provided to obtain multiplexed output event of the available output events. Each 32-bit counter has two 16-bit channels. These channels can be programmed to use 32-bits or only 16-bits.



Configurable Timer block diagram

The Configuration counter has two counters each 16-bit wide. There is a feature to use both counters collectively to obtain a 32-bit counter. These counters has the capability to run as free-running counter, start counting, halt the counting operation, continuing the counter operation, increment the counter value and also to capture the counter

value respectively according to the programmed values in the respective registers. Events for respective operations can also be programmed through the registers. There are also control registers available to support OCU operation. 16-bit or 32-bit modes of the counters can be selected through programmable register.

Programming sequence

Events handling

Input signals provided through GPIO pins are used to decide the event type such as rising edge, falling edge, high level triggering or low level triggering. These Events are described in below Table17-2. Appropriate registers should be programmed according to the requirement (see registers description section for more details). Respective counter values can be read using firmware and also output events are obtained through the output pins.

| Event selection | Event type | Description |
|-----------------|------------|---------------------------------|
| 0 | None | No event is selected. |
| 1 | Eve_0_re | Event 0 rising edge. |
| 2 | Eve_1_re | Event 1 rising edge. |
| 3 | Eve_2_re | Event 2 rising edge |
| 4 | Eve_3_re | Event 3 rising edge. |
| 5 | Eve_0_fe | Event 0 falling edge. |
| 6 | Eve_1_fe | Event 1 falling edge. |
| 7 | Eve_2_fe | Event 2 falling edge. |
| 8 | Eve_3_fe | Event 3 falling edge. |
| 9 | Eve_0_rfe | Event 0 rising or falling edge. |
| 10 | Eve_1_rfe | Event 1 rising or falling edge. |
| 11 | Eve_2_rfe | Event 2 rising or falling edge. |
| 12 | Eve_3_rfe | Event 3 rising or falling edge. |
| 13 | Eve_0_lev0 | Event 0 Level 0. |
| 14 | Eve_1_lev0 | Event 1 Level 0. |
| 15 | Eve_2_lev0 | Event 2 Level 0. |
| 16 | Eve_3_lev0 | Event 3 Level 0. |
| 17 | Eve_0_lev1 | Event 0 Level 1. |
| 18 | Eve_1_lev1 | Event 1 Level 1. |
| 19 | Eve_2_lev1 | Event 2 Level 1. |
| 20 | Eve_3_lev1 | Event 3 Level 1. |
| 21 | AND_eve | (Events == and event). |

| Event selection | Event type | Description |
|-----------------|----------------|--|
| 22 | OR_eve | (Event 0 == OR event 0) (Event 1 == OR event 1) (Event 2 == OR event 2) (Event 3 == OR event 3) |
| 23 | AND_ev_clked0 | Event 0 rising edge & AND_eve |
| 24 | OR_ev_clked0 | Event 0 rising edge & OR_eve |
| 25 | AND_ev_clked1 | Event 1 rising edge & AND_eve |
| 26 | OR_ev_clked1 | Event 1 rising edge & OR_eve |
| 27 | AND_ev_clked2 | Event 2 rising edge & AND_eve |
| 28 | OR_ev_clked2 | Event 2 rising edge & OR_eve |
| 29 | AND_ev_clked3 | Event 3 rising edge & AND_eve |
| 30 | OR_ev_clked3 | Event 3 rising edge & OR_eve |
| 31 | AND_ev_clkedr0 | Event 0 rising edge registered & AND_eve |
| 32 | OR_ev_clkedr0 | Event 0 rising edge registered & OR_eve |
| 33 | AND_ev_clkedr1 | Event 1 rising edge registered & AND_eve |
| 34 | OR_ev_clkedr1 | Event 1 rising edge registered & OR_eve |
| 35 | AND_ev_clkedr2 | Event 2 rising edge registered & AND_eve |
| 36 | OR_ev_clkedr2 | Event 2 rising edge registered & OR_eve |
| 37 | AND_ev_clkedr3 | Event 3 rising edge registered & AND_eve |
| 38 | OR_ev_clkedr3 | Event 3 rising edge registered & OR_eve |

587 . Events Description

Procedure for following Configurable timer use cases:

- 1) Free Run Timer (FRT) up mode.
 - Set UP mode in gen_ctrl_reg register.
 - Configure either in 16 or 32 bit mode.
 - Write PEAK value to MAX_REG register.
 - Configure the CT_INTR registers for generating required interrupts.
 - Select the start signals either from input events or software triggers.
- 2) FRT up-down mode.
 - Set UP-DOWN mode in gen_ctrl_reg register.
 - Configure either in 16 or 32 bit mode.
 - Write PEAK value to MAX_REG register.
 - Configure the CT_INTR registers for generating required interrupts.
 - Select the start signals either from input events or software triggers.

-
- 3) FRT down mode.
 - Set DOWN mode in gen_ctrl_reg register.
 - Configure either in 16 or 32 bit mode.
 - Write PEAK value to MAX_REG register.
 - Configure the CT_INTR registers for generating required interrupts.
 - Select the start signals either from input events or software triggers.
 - 4) FRT DOWN-UP mode.
 - Set DOWN-UP mode in gen_ctrl_reg register.
 - Configure either in 16 or 32 bit mode.
 - Write PEAK value to MAX_REG register.
 - Configure the CT_INTR registers for generating required interrupts.
 - Select the start signals either from input events or software triggers.
 - 5) FRT up mode with MAX_BUF register.
 - Enable buf_reg_0/1_en for respective counter.
 - Set UP mode in gen_ctrl_reg.
 - Configure either in 16 or 32 bit mode.
 - Write PEAK value to MAX_REG register.
 - Configure the CT_INTR registers for generating required interrupts.
 - Select the start signals either from input events or software triggers.
 - 6) PWM/WFG outputs
 - Set output_is_ocu
 - Write PEAK value to MAX_REG register.
 - Write cycle/period value to OCU_COMPARE_REG.
 - Select make_output_*_high_sel and make_output_*_low_sel accordingly.
 - For WFG, after above procedure, set wfg_tgl_cnt register for required number of clocks.
 - 7) NZCL/ADCMP
 - Set input events (select input event on which you want to trigger ADC).
 - Select counter
 - Select Interrupt mux registers.
 - Unmask the interrupts/trigger signals according to ADC trig registers.
 - To provide one output event to the ADC, appropriate select value can be programmed using “OUTPUT_EVENT_ADC_SEL”. The multiplexed output event can then be obtained on the “ct_output_event” pin.
 - 8) ICU (input capture)
 - Select the input event on which counter needs to be started, otherwise trigger counter through software option.
 - Select the input event(s) on which counter need to be captured.
 - Select the interrupt generation also to the capture event so that CPU gets the interrupt when capture happen.
 - Read the CAPTURE_REG when interrupt is seen.
 - 9) OCU mode

ON period of the next OCU cycle should be programmed using the “OCU_COMPARE_NEXT_REG”. To avoid the overwriting of the already programmed ON period of the next cycle before it gets executed, “fifo_full” signal should be polled. Program the out of the available “fifo_full” signals using the “MUX_SEL_REG”. Two fifo_full signals for each counter can be selected at the top level. Counter OCUs description is shown in following Table 17-3.

| No | Select | Result becomes |
|----|--------|--|
| 1 | 0 | 0 |
| 2 | 1 | 1 |
| 3 | 2 | It becomes one, when counter matches with ocu_compare_0 |
| 4 | 3 | It becomes one, when counter matches with ocu_compare_1 |
| 5 | 4 | It becomes one, when counter matches with ocu_compare_0 & counter is in up direction |
| 6 | 5 | It becomes one, when counter matches with ocu_compare_1 & counter is in up direction |
| 7 | 6 | It becomes one, when counter matches with ocu_compare_0 & counter is in down direction |
| 8 | 7 | It becomes one, when counter matches with ocu_compare_1 & counter is in down direction |

588 . Counter OCUs description

16.3.4 Register Summary

Base Address: 0x4506_0000

| S.no | Timer Name | Offset value | Description |
|------|------------------|--------------|--|
| 1 | Timer 0 | 0x0000 | Timer 0 offset address |
| 2 | Timer 1 | 0x1000 | Timer 1 offset address |
| 3 | DMA flow control | 0xF000 | DMA flow control multiplexing select register offset address |

589 . List of timer offset addresses

| S.no | Counter Name | Offset value | Description |
|------|--------------|--------------|--------------------------|
| 1 | Counter 0 | 0x000 | Counter 0 offset address |
| 2 | Counter 1 | 0x100 | Counter 1 offset address |

590 . List of counter offset addresses

| Register Name | Offset | Description |
|-----------------------|--------|--|
| CT_GEN_CTRL_SET_REG | 0x0 | General control set register. Holds general control signals like enables and modes. This register is used only for setting the control signals |
| CT_GEN_CTRL_RESET_REG | 0x4 | General control reset register. Holds general control signals like disables. This register is used only for resetting the control signals |
| CT_INTR_STS_REG | 0x8 | Holds the interrupt status |
| CT_INTR_MASK_REG | 0xC | Unwanted interrupts mask. Bits can be set to not to receive the interrupt indication |

| Register Name | Offset | Description |
|-----------------------------------|-------------|--|
| CT_INTR_UNMASK_REG | 0x10 | Wanted interrupts mask. Bits can be set to receive the interrupt indication |
| CT_INTR_ACK_REG | 0x14 | Interrupt clear/ack register |
| CT_MATCH_REG | 0x18 | Match value register. Top value for the timer/counter |
| CT_MATCH_BUF_REG | 0x1C | Match buffer register. This will be loaded to CT_MATCH_REG when counter is active and becomes zero |
| CT_CAPTURE_REG | 0x20 | Captures and holds the counter value at the selected event occurrence |
| CT_COUNTER_REG | 0x24 | Holds the values of counter-1 and counter-2 |
| CT_OCU_CTRL_REG | 0x28 | Bits in this register are used to configure the values corresponding to OCU operation |
| CT_OCU_COMPARE_REG | 0x2C | Holds the threshold values for counter-0 which can be compared for making output event of counter-0 high/low basing on the ocu_ctrl_reg signals for present OCU period |
| CT_OCU_COMPARE2_REG | 0x30 | Holds the threshold values for counter-1 which can be compared for making output event of counter-1 high/low basing on the ocu_ctrl_reg signals for present OCU period |
| CT_OCU_SYNC_REG | 0x34 | Used as starting value for syncing OCU counters |
| CT_OCU_COMPARE_NXT_REG | 0x38 | Holds the threshold for next OCU period which will get loaded in to CT_OCU_COMPARE_REG |
| CT_OCU_COMPARE2_NXT_REG | 0x40 | Holds the threshold for next OCU period which will get loaded in to CT_OCU_COMPARE2_REG |
| CT_WFG_CTRL_REG | 0x3C | Holds WFG related control signals like enables and modes |
| Reserved | 0x40 – 0x4f | Reserved for future use |
| CT_START_COUNTER_EVENT_SEL_REG | 0x50 | Start counter event select register |
| CT_START_COUNTER_AND_EVENT_REG | 0x54 | Start counter “AND” event configuration register |
| CT_START_COUNTER_OR_EVENT_REG | 0x58 | Start counter “OR” event configuration register |
| CT_CONTINUE_COUNTER_EVENT_SEL_REG | 0x5C | Continue counter event select register |
| CT_CONTINUE_COUNTER_AND_EVENT_REG | 0x60 | Continue counter “AND” event configuration register |
| CT_CONTINUE_COUNTER_OR_EVENT_REG | 0x64 | Continue counter “OR” event configuration register |
| CT_STOP_COUNTER_EVENT_SEL_REG | 0x68 | Stop counter event select register |

| Register Name | Offset | Description |
|--|--------|--|
| CT_STOP_COUNTER_AND_EVENT_REG | 0x6C | Stop counter “AND” event configuration register |
| CT_STOP_COUNTER_OR_EVENT_REG | 0x70 | Stop counter “OR” event configuration register |
| CT_HALT_COUNTER_EVENT_SEL_REG | 0x74 | Halt counter event select register |
| CT_HALT_COUNTER_AND_EVENT_REG | 0x78 | Halt counter “AND” event configuration register |
| CT_HALT_COUNTER_OR_EVENT_REG | 0x7C | Halt counter “OR” event configuration register |
| CT_INCREMENT_COUNTER_EVENT_SEL_REG | 0x80 | Increment counter event select register |
| CT_INCREMENT_COUNTER_AND_EVENT_REG | 0x84 | Increment counter “AND” event configuration register |
| CT_INCREMENT_COUNTER_OR_EVENT_REG | 0x88 | Increment counter “OR” event configuration register |
| CT_CAPTURE_COUNTER_EVENT_SEL_REG | 0x8C | Capture counter event select register |
| CT_CAPTURE_COUNTER_AND_EVENT_REG | 0x90 | Capture counter “AND” event configuration register |
| CT_CAPTURE_COUNTER_OR_EVENT_REG | 0x94 | Capture counter “OR” event configuration register |
| CT_OUTPUT_EVENT_SEL_REG | 0x98 | Output event select configuration register |
| CT_OUTPUT_AND_EVENT_REG | 0x9C | Output AND event configuration register |
| CT_OUTPUT_OR_EVENT_REG | 0xA0 | Output OR event configuration register |
| CT_INTR_EVENT_SEL_REG | 0xA4 | Interrupt event select configuration register |
| CT_INTR_AND_EVENT_REG | 0xA8 | Interrupt “AND” event configuration register |
| CT_INTR_OR_EVENT_REG | 0xAC | Interrupt “OR” event configuration register |
| CT_RE_FE_RFE_LEV0LEV1_EVENT_ENABLE_REG | 0xB0 | Rising Edge Falling Edge Rising and Falling Edge level0 level1 event enable register |

591 . List of registers in each counter

CT_MUX_SEL_0_REG

| Register Name | Offset | Description |
|------------------------------|--------|----------------------------------|
| 0xf000 | | Mux sel-0 configuration register |
| CT_MUX_SEL_1_REG | 0xf004 | Mux sel-1 configuration register |
| CT_MUX_SEL_2_REG | 0xf008 | Mux sel-2 configuration register |
| CT_MUX_SEL_3_REG | 0xf00C | Mux sel-3 configuration register |
| CT_OUTPUT_EVENT1_ADC_SEL_REG | 0xf018 | Output event ADC select register |
| CT_OUTPUT_EVENT1_ADC_SEL_REG | 0xf01C | Output event ADC select register |

592 . List of common registers in CT

16.3.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write

CT_GEN_CTRL_SET_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------|-----------|--|
| 31:24 | R | Reserved | 0 | Reserved for future use. |
| 23 | R/W | Buf_reg_1_en | 0x0 | <p>Buffer register gets enabled for MATCH REG. MATCH_BUF_REG is always available and whenever this bit is set only, gets copied to MATCH REG.</p> <p>If write 1 – Buffer will be enabled and in path 0 – No effect.</p> <p>If read 1 – Buffer is enabled and in path 0 – Buffer is not enabled and not in path</p> |

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------|-----------|---|
| 22 | R/W | Counter_1_sync_trig | 0x0* | <p>This is applied to only higher 16 bits of counter.</p> <p>This enables the counter to run/active when sync is found.</p> <p>If write</p> <p>1 – Counter_1 will be active.</p> <p>0 – No effect.</p> <p>If read</p> <p>Read should always return 0.</p> <p>For counter to get increment / decrement, counter should be active and hit with selected event.</p> |
| 21:20 | R/W | Counter_1_up_down | 0x0 | <p>This is applied to only higher 16 bits of counter.</p> <p>This enables the counter to run in up/down/up-down/down-up directions</p> <p>If write</p> <p>0 – No effect to old value.</p> <p>1 – Counter up direction enable</p> <p>2 – Counter down direction enable</p> <p>3 – Both up and down directions enable.</p> <p>If read</p> <p>0 – Counter_1 is in down-up counting mode.</p> <p>1 – Counter_1 is in up counting mode.</p> <p>2 – Counter_1 is in down counting mode.</p> <p>3 – Counter_1 is in up-down counting mode.</p> |
| 19 | R/W | Counter_1_trig_frm_re g | 0x0* | <p>This is valid only when the counter is in two 16 bit counters mode.</p> <p>This enables the counter to run/active.</p> <p>If write</p> <p>1 – Counter_1 will be active.</p> <p>0 – No effect.</p> <p>If read</p> <p>Read should always return 0.</p> <p>For counter to get increment / decrement, counter should be active and hit with selected event.</p> |

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------------|-----------|--|
| 18 | R/W | Periodic_en_Counter_1_0x0_frm_reg | 0x0 | <p>This is valid only when the counter is in two 16 bit counters mode.</p> <p>This enables the counter to re-run even after expire/saturation value hit.</p> <p>If write</p> <p>1 – Counter_1 will be in periodic mode.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Counter_1 is in periodic mode.</p> <p>0 – Counter_1 is not in periodic mode.</p> |
| 17 | R/W | Soft_reset_Counter_1_f0x0*rm_reg | 0x0* | <p>This is valid only when the counter is in two 16 bit counters mode.</p> <p>This resets the counter on the write.</p> <p>If write</p> <p>1 – Counter_1 will be reset.</p> <p>0 – No effect.</p> <p>If Read</p> <p>Read always should return 0.</p> |
| 16:8 | R | Reserved | 0x0 | Reserved for future use. |
| 7 | R/W | Buf_reg_0_en | 0x0 | <p>Buffer register gets enabled for MATCH REG. MATCH_BUF_REG is always available and whenever this bit is set only, gets copied to MATCH REG.</p> <p>If write</p> <p>1 – Buffer will be enabled and in path</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Buffer is enabled and in path</p> <p>0 – Buffer is not enabled and not in path.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 6 | R/W | Counter_0_sync_trig | 0x0* | <p>This is applied to 32 bits of counter only when the counter is in 32 bit counter mode otherwise this will be applied to only lower 16 bits of counter.</p> <p>This enables the counter to run/active when sync is found.</p> <p>If write</p> <p>1 – Counter_0 will be active.</p> <p>0 – No effect.</p> <p>If read</p> <p>Read should always return 0.</p> <p>For counter to get increment / decrement, counter should be active and hit with selected event.</p> |
| 5:4 | R/W | Counter_0_up_down | 0x0 | <p>This is applied to 32 bits of counter only when the counter is in 32 bit counter mode otherwise this will be applied to only lower 16 bits of counter.</p> <p>This enables the counter to run in up/down/up-down/down-up directions</p> <p>If write</p> <p>0 – No effect to old value.</p> <p>1 – Counter up direction enable</p> <p>2 – Counter down direction enable</p> <p>3 – Both up and down directions enable.</p> <p>If read</p> <p>0 – Counter_0 is in down-up counting mode.</p> <p>1 – Counter_0 is in up counting mode.</p> <p>2 – Counter_0 is in down counting mode.</p> <p>3 – Counter_0 is in up-down counting mode.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------------------|-----------|--|
| 3 | R/W | Counter_0_trig_frm_re g | 0x0* | <p>This is applied to 32 bits of counter only when the counter is in 32 bit counter mode otherwise this will be applied to only lower 16 bits of counter.</p> <p>This enables the counter to run/active.</p> <p>If write</p> <p>1 – Counter_0 will be active.</p> <p>0 – No effect.</p> <p>If read</p> <p>Read should always return 0.</p> <p>For counter to get increment / decrement, counter should be active and hit with selected event.</p> |
| 2 | R/W | Periodic_en_Counter_0 _frm_reg | 0x0 | <p>This is applied to 32 bits of counter only when the counter is in 32 bit counter mode otherwise this will be applied to only lower 16 bits of counter.</p> <p>This enables the counter to re-run even after expire/saturation value hit.</p> <p>If write</p> <p>1 – Counter_1 will be in periodic mode.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Counter_1 is in periodic mode.</p> <p>0 – Counter_1 is not in periodic mode.</p> <p>The saturation value is programmed in “match_reg”. Also note that, in periodic mode, two values less than the required period needs to be programmed in the “match_reg”. For example, if the counter needs to run from 0 to 4 (i.e period is 5 counter values), then program 3 in the match_reg in periodic mode.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------------|-----------|--|
| 1 | R/W | Soft_reset_counter_0_f rm_reg | 0x0* | <p>This is applied to 32 bits of counter only when the counter is in 32 bit counter mode otherwise this will be applied to only lower 16 bits of counter.</p> <p>This resets the counter on the write.</p> <p>If write</p> <p>1 – Counter_0 will be reset. 0 – No effect.</p> <p>If Read</p> <p>Read always should return 0.</p> |
| 0 | R/W | Counter_in_32_bit_mo de | 0x0 | <p>Counter_1 and Counter_0 will be merged and used as a single 32 bit counter. In this mode, Counter_0 modes/triggers/enables will be used.</p> <p>If write</p> <p>1 – Counter will be in 32 bit mode. 0 – No effect.</p> <p>If read</p> <p>1 – Counter is in 32 bit mode. 0 – Counter is in two 16 bit counters mode.</p> |

* - Indicates self clear.

593 .GEN_CTRL_SET_REG Register Description

CT_GEN_CTRL_RESET_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------|-----------|--|
| 31:24 | R | Reserved | 0 | Reserved for future use. |
| 23 | R/W | Buf_reg_1_en | 0x0 | <p>Buffer register gets enabled for MATCH REG. MATCH_BUF_REG is always available and whenever this bit is set only, gets copied to MATCH REG.</p> <p>If write</p> <p>1 – Buffer will be disabled and in path 0 – No effect.</p> <p>If read</p> <p>1 – Buffer is enabled and in path 0 – Buffer is not enabled and not in path.</p> |
| 22 | R | Counter_1_sync_trig | 0x0 | Self clear bit. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------------------|-----------|---|
| 21:20 | R/W | Counter_1_up_down | 0x0 | <p>This is applied to only higher 16 bits of counter.</p> <p>This enables the counter to run in up/down/up-down/down-up directions</p> <p>0 – Counter_1 is in down-up counting mode.</p> <p>1 – Counter_1 is in up counting mode.</p> <p>2 – Counter_1 is in down counting mode.</p> <p>3 – Counter_1 is in up-down counting mode.</p> |
| 19 | R | Counter_1_trig_frm_re g | 0x0 | This is a self clear bit in set register. |
| 18 | R/W | Periodic_en_Counter_1 _frm_reg | 0x0 | <p>This is valid only when the counter is in two 16 bit counters mode.</p> <p>This enables the counter to re-run even after expire/saturation value hit.</p> <p>If write</p> <p>1 – Counter_1 will be in single shot mode.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Counter_1 is in periodic mode.</p> <p>0 – Counter_1 is in single shot mode.</p> <p>The saturation value is programmed in “match_reg_1”. Also note that, in periodic mode, two values less than the required period needs to be programmed in the “match_reg_1”. For example, if the counter needs to run from 0 to 4 (i.e period is 5 counter values), then program 3 in the match_reg in periodic mode.</p> |
| 17 | R | Soft_reset_Counter_1_f rm_reg | 0x0 | This is a self clear bit in set register. |
| 16:8 | R | Reserved | 0x0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------------------|-----------|--|
| 7 | R/W | Buf_reg_0_en | 0x0 | <p>Buffer register gets enabled for MATCH REG. MATCH_BUF_REG is always available and whenever this bit is set only, gets copied to MATCH REG.</p> <p>If write</p> <p>1 – Buffer will be disabled and in path 0 – No effect.</p> <p>If read</p> <p>1 – Buffer is enabled and in path 0 – Buffer is not enabled and not in path.</p> |
| 6 | R | Counter_0_sync_trig | 0x0 | This is a self clear bit in set register. |
| 5:4 | R/W | Counter_0_up_down | 0x0 | <p>This is applied to 32 bits of counter only when the counter is in 32 bit counter mode otherwise this will be applied to only lower 16 bits of counter.</p> <p>This enables the counter to run in up/down/up-down/down-up directions</p> <p>0 – Counter_0 is in down-up counting mode. 1 – Counter_0 is in up counting mode. 2 – Counter_0 is in down counting mode. 3 – Counter_0 is in up-down counting mode.</p> |
| 3 | R | Counter_0_trig_frm_re g | 0x0 | This is a self clear bit in set register. |
| 2 | R/W | Periodic_en_Counter_0 _frm_reg | 0x0 | <p>This is applied to 32 bits of counter only when the counter is in 32 bit counter mode otherwise this will be applied to only lower 16 bits of counter.</p> <p>This enables the counter to re-run even after expire/saturation value hit.</p> <p>If write</p> <p>1 – Counter_1 will be in single count mode. 0 – No effect.</p> <p>If read</p> <p>1 – Counter_1 is in periodic mode. 0 – Counter_1 is in single shot mode.</p> |
| 1 | R | Soft_reset_counter_0_f rm_reg | 0x0 | This is a self clear bit in set register. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 0 | R/W | Counter_in_32_bit_mode | 0x0 | <p>Counter_1 and Counter_0 will be merged and used as a single 32 bit counter. In this mode, Counter_0 modes/triggers/enables will be used.</p> <p>If write</p> <p>1 – Counter will be in two 16 bit counters mode. 0 – No effect.</p> <p>If read</p> <p>1 – Counter is in 32 bit mode. 0 – Counter is in two 16 bit counters mode.</p> |

594 .GEN_CTRL_RESET_REG Register Description

CT_INTR_STS_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------|-----------|---|
| 31:20 | R | Reserved | 0 | Reserved for future use. |
| 19 | R | Counter_1_is_peak_l0 | 0 | Counter 1 hit peak (MATCH) in active mode. |
| 18 | R | Counter_1_is_zero_l0 | 0 | Counter 1 hit zero in active mode. |
| 17 | R | fifo_1_full_l | 0 | Indicates the FIFO full signal of channel-1 |
| 16 | R | intr_1_l | 0 | This is derived from the input events. |
| 15:4 | R | Reserved | 0 | Reserved for future use. |
| 3 | R | Counter_0_is_peak_l0 | 0 | Counter 0 hit peak (MATCH) in active mode. |
| 2 | R | Counter_0_is_zero_l0 | 0 | Counter 0 hit zero in active mode. |
| 1 | R | fifo_0_full_l | 0 | Indicates the FIFO full signal of channel-0 |
| 0 | R | intr_0_l | 0 | This is derived from the input events. |

595 .INTR_STS Register Description

CT_INTR_MASK_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--------------------------|
| 31:20 | R | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|---|
| 19 | R/W | Counter_1_is_peak_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 18 | R/W | Counter_1_is_zero_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 17 | R/W | fifo_1_full_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 16 | R/W | intr_1_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 15:4 | R | Reserved | 0 | Reserved for future use |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 3 | R/W | Counter_0_is_peak_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 2 | R/W | Counter_0_is_zero_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 1 | R/W | fifo_0_full_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 0 | R/W | intr_0_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be masked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |

596 . INTR_MASK Register Description

CT_INTR_UNMASK_REG

| Bit | Access | Function | POR Value | Description | |
|-------|--------|----------|-----------|--------------------------|--|
| 31:20 | R | Reserved | 0 | Reserved for future use. | |

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|---|
| 19 | R/W | Counter_1_is_peak_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 18 | R/W | Counter_1_is_zero_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 17 | R/W | fifo_1_full_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 16 | R/W | intr_1_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 15:4 | R | Reserved | 0 | Reserved for future use |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 3 | R/W | Counter_0_is_peak_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 2 | R/W | Counter_0_is_zero_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 1 | R/W | fifo_0_full_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |
| 0 | R/W | intr_0_l | 0x0 | <p>Interrupt mask signal.</p> <p>If write</p> <p>1 – Interrupt will be unmasked.</p> <p>0 – No effect.</p> <p>If read</p> <p>1 – Interrupt is unmasked.</p> <p>0 – Interrupt is masked.</p> |

597 .INTR_UNMASK Register Description

CT_INTR_ACK_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--------------------------|
| 31:20 | R | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|--|
| 19 | R/W | Counter_1_is_peak_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |
| 18 | R/W | Counter_1_is_zero_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |
| 17 | R/W | fifo_1_full_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |
| 16 | R/W | intr_1_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |
| 15:4 | R | Reserved | 0 | Reserved for future use |
| 3 | R/W | Counter_0_is_peak_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |
| 2 | R/W | Counter_0_is_zero_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------|-----------|--|
| 1 | R/W | fifo_0_full_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |
| 0 | R/W | intr_0_l | 0x0* | <p>Interrupt ack signal.</p> <p>If write</p> <p>1 – Interrupt will be deasserted.</p> <p>0 – No effect.</p> <p>If read 0 should be returned as this is self clear bit.</p> |

598 .INTR_ACK Register Description

CT_MATCH_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------|-----------|--|
| 31:16 | R/W | Counter_1_match | 0xffff | <p>Counter upper part match register. This will act as match register for counter-1. When even this is hit/matched, counter 1 generates interrupt/output events.</p> <p>In 32-bit mode and 16-bit mode, both the matches work independently for each of the counters.</p> <p>So for 32-bit mode, if we want to get match_value as “0x0002_0000” for entire 32-bit counter, then Counter_1_match should be programmed as “0x0002” and</p> <p>“Counter_0_match” should be programmed as “0xffff”. This is because, counter-1 will be increment only after counter-0 overflows.</p> <p>If we want to get match value as “0x0002_00ff”, then Counter_1_match should be programmed as “0x0002” and initially Counter_0_match should be programmed as “0xffff” in order to make counter-1 increment and then we can keep counter_0_match should be programmed to “0x00ff”.</p> |
| 15:0 | R/W | Counter_0_match | 0xffff | Counter lower part match register. This will act as match register for counter-0. When even this is hit/matched, counter 1 generates interrupt/output events. |

599 .MATCH Register Description

CT_MATCH_BUF_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------|-----------|---|
| 31:16 | R/W | Counter_1_match_b uf | 0xfffff | This gets copied to MATCH register if bug_reg_1_en is set. Copying is done when counter 1 is active and hits 0. |
| 15:0 | R/W | Counter_0_match_b uf | 0xfffff | This gets copied to MATCH register if bug_reg_0_en is set. Copying is done when counter 0 is active and hits 0. |

600 . MATCH_BUF Register Description

CT_CAPTURE_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------|-----------|---|
| 31:16 | R | Counter_1_capture | 0x0 | This is a latched value of counter upper part when the selected capture_event occurs. |
| 15:0 | R | Counter_0_capture | 0x0 | This is a latched value of counter lower part when the selected capture_event occurs. |

601 . Capture Register Description

CT_COUNTER_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------|-----------|-----------------------------------|
| 31:16 | R/W | Counter 1 | 0x0 | This holds the value of counter-1 |
| 15:0 | R/W | Counter 0 | 0x0 | This holds the value of counter-0 |

602 . Counter Register Description

CT_OCU_CTRL_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------|-----------|---|
| 31:28 | R/W | Reserved | 0 | Reserved for future use. |
| 27:25 | R/W | Make_output_1_low_s el | 0 | Check counter ocus for possibilities. When this is hit output will be made low. |
| 24:22 | R/W | Make_output_1_high_ sel | 0 | Check counter ocus for possibilities. When this is hit output will be made high. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------|-----------|---|
| 21 | R/W | ocu_1_mode_8_16_m ode | 0 | Indicates whether entire 16 bits or only 8-bits of the channel-1 are used in OCU mode. 1. - 8 bit mode 1. - 16 bit mode |
| 20 | R/W | ocu_1_dma_mode | 0 | Indicates whether the OCU – DMA mode is active or not for channel-1 |
| 19:17 | R/W | sync_with_1 | 0 | Indicates whether the other channel is in sync with this channel-1 |
| 16 | R/W | output_1_is_ocu | 0 | Indicates whether the output is in OCU mode or not for channel-1 |
| 15:12 | R/W | Reserved | 0 | Reserved for future use. |
| 11:9 | R/W | Make_output_0_low_s el | 0 | Check counter ocus for possibilities. When this is hit output will be made low. |
| 8:6 | R/W | Make_output_0_high_ sel | 0 | Check counter ocus for possibilities. When this is hit output will be made high. |
| 5 | R/W | ocu_0_mode_8_16 | 0 | Indicates whether entire 16 bits or only 8-bits of the channel-0 are used in OCU mode 1- 8 bit mode 0 - 16 bit mode |
| 4 | R/W | ocu__0_dma_mode | 0 | Indicates whether the OCU – DMA mode is active or not for channel-0 |
| 3:1 | R/W | sync_with_0 | 0 | Indicates whether the other channel is in sync with this channel-0 |
| 0 | R/W | output_0_is_ocu | 0 | Indicates whether the output is in OCU mode or not for channel-0 |

603 .OCU_CTRL Register Description

CT_OCU_COMPARE_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|--|
| 31:16 | R/W | ocu_compare_1_re g | 0 | Holds one of the two threshold values of present OCU period-1 which ultimately gets compared with the counter-1 count value to make output-1 high/low according to the configuration selected in ocu_ctrl_reg (counter 1) |

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------|-----------|---|
| 15:0 | R/W | ocu_compare_0_reg | 0 | Holds one of the two threshold values of present OCU period-2 which ultimately gets compared with the counter-0 count value to make output-0 high/low according to the configuration selected in ocu_ctrl_reg (counter 0) |

604 . OCU Compare Register Description

CT_OCU_COMPARE2_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------|-----------|---|
| 31:16 | R/W | ocu_compare2_1_reg | 0 | Holds one of the two threshold values of present OCU period-1 which ultimately gets compared with the counter-1 count value to make output-1 high/low according to the configuration selected in ocu_ctrl_reg (counter 1) |
| 15:0 | R/W | ocu_compare2_0_reg | 0 | Holds one of the two threshold values of present OCU period-2 which ultimately gets compared with the counter-0 count value to make output-0 high/low according to the configuration selected in ocu_ctrl_reg (counter 0) |

605 . OCU Compare2 Register Description

CT_OCU_SYNC_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------|-----------|---|
| 31:16 | R/W | ocu_sync_reg_1 | 0 | Holds the starting point of channel-1 for synchronization purpose |
| 15:0 | R/W | ocu_sync_reg_0 | 0 | Holds the starting point of channel-0 for synchronization purpose |

606 . OCU Sync Register Description

CT_OCU_COMPARE_NXT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------|-----------|---|
| 31:16 | R/W | ocu_compare_nxt_reg | 0 | Holds the threshold value of next OCU period-1 which ultimately gets loaded in to ocu_compare_reg in dma mode for (counter 0) |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|----------------------|------------------|---|
| 15:0 | R/W | ocu_compare_nxt_re0g | 0 | Holds the threshold value of next OCU period-2 which ultimately gets loaded in to ocu_compare_reg in dma mode for (counter 0) |

607 .OCU_Compare_Nxt Register Description

CT_OCU_COMPARE2_NXT_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|------------------------|------------------|--|
| 31:16 | R/W | ocu_compare2_nxt_re0eg | 0 | Holds the threshold value of next OCU period-1 which ultimately gets loaded in to ocu_compare2_reg in dma mode for (counter 1) |
| 15:0 | R/W | ocu_compare2_nxt_re0eg | 0 | Holds the threshold value of next OCU period-2 which ultimately gets loaded in to ocu_compare2_reg in dma mode for (counter 1) |

608 .OCU_Compare_Nxt2 Register Description

CT_WFG_CTRL_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|----------------------|------------------|---|
| 31:24 | R/W | Wfg_tgl_cnt_1_peak | 0 | WFG mode output toggle count clock for channel 1. |
| 23:22 | R/W | Reserved | 0 | Reserved for future use. |
| 21:19 | R/W | Make_output_1_tgl_10 | 0 | Check the counter ocus possibilities for description for channel 1. |
| 18:16 | R/W | Make_output_1_tgl_00 | 0 | Check the counter ocus possibilities for description for channel 1. |
| 15:8 | R/W | Wfg_tgl_cnt_0_peak | 0 | WFG mode output toggle count clock for channel 0. |
| 7:6 | R/W | Reserved | 0 | Reserved for future use. |
| 5:3 | R/W | Make_output_0_tgl_10 | 0 | Check the counter ocus possibilities for description for channel 0. |
| 2:0 | R/W | Make_output_0_tgl_00 | 0 | Check the counter ocus possibilities for description for channel 0. |

609 .WFG_Ctrl Register Description

CT_START_COUNTER_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------------------|-----------|--|
| 31:22 | R | Reserved | 0 | Reserved for future use. |
| 21:16 | R/W | Start_Counter_1_even 0 t_sel | | For two 16 bit counters mode: Event select for starting the Counter_1. For 32 bit counter mode: Invalid. Please refer to events table for description. |
| 15:6 | R/W | Reserved | 0 | Reserved for future use. |
| 5:0 | R/W | Start_Counter_0_even 0 t_sel | | For two 16 bit counters mode: Event select for starting the Counter_0. For 32 bit counter mode: Event select for starting counter. |

610 . Start_Counter_Event_Sel Register Description

CT_START_COUNTER_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | start_counter_1_and_0 vld | | Indicates which bits in [19:16] are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Start_Counter_1_AND_0 event | | For two 16 bit counters mode: AND expression for AND event in start counter_1 event. Ex: If [19:16] is 1100 then level 1 is checked on input event 3 and input event 2 and level 0 is checked on input event 1 and input event 0. Again considering of bits out of [19:16] depends on corresponding valid bits programmed in [27:24] For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | start_counter_0_and_vld | | Indicates which bits in 3:0 are valid for considering AND event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------|-----------|---|
| 3:0 | R/W | Start_Counter_0_AND_0 event | | <p>For two 16 bit counter mode: AND expression for AND event in start Counter_0 event.</p> <p>Ex: If [19:16] is 1100 then level 1 is checked on input event 3, input event 2 and level 0 is checked on input event 1 and input event 0.</p> <p>Again considering of bits out of [19:16] depends on corresponding valid bits programmed in [27:24]</p> <p>For 32 bit counter mode: AND expression is valid for AND event in start counter event.</p> |

611 . Start_Counter_And_Event Register Description

CT_START_COUNTER_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------|-----------|---|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | start_counter_1_or_vld | 0 | Indicates which bits in [19:16] are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Start_Counter_1_OR_0 event | | <p>For two 16 bit counters mode: OR expression for OR event in start counter event.</p> <p>Ex: If [19:16] is 1100 then level 1 is checked on input event 3 or input event 2 or level 0 is checked on input event 1 or input event 0.</p> <p>Again considering of bits out of [19:16] depends on corresponding valid bits programmed in [27:24]</p> <p>For 32 bit counter mode :Invalid.</p> |
| 15:12 | R/W | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | start_counter_0_or_vld | | Indicates which bits in 3:0 are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------|-----------|---|
| 3:0 | R/W | Start_Counter_0_OR_0 event | | <p>For two 16 bit counter mode: OR expression valid bits for OR event in start Counter_0 event.</p> <p>Ex: If [19:16] is 1100 then level 1 is checked on input event 3 or input event 2 or level 0 is checked on input event 1 or input event 0.</p> <p>Again considering of bits out of [19:16] depends on corresponding valid bits programmed in [27:24]</p> <p>For 32 bit counter mode OR expression is valid for OR event in start counter event.</p> |

612 . Start_Counter_Or_Event Register Description

CT_CONTINUE_COUNTER_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------------------|-----------|---|
| 31:22 | R | Reserved | 0 | Reserved for future use. |
| 21:16 | R/W | Continue_Counter_1_e0 vent_sel | | <p>For two 16 bit counters mode: Event select for continuing the Counter_1.</p> <p>For 32 bit counter mode: Invalid.</p> <p>Please refer to events table for description.</p> |
| 15:6 | R/W | Reserved | 0 | Reserved for future use. |
| 5:0 | R/W | Continue_Counter_0_e0 vent_sel | | <p>For two 16 bit counters mode: Event select for continuing the Counter_0.</p> <p>For 32 bit counter mode: Event select for continuing counter.</p> |

613 . Continue_Counter_Event_Sel Register Description

CT_CONTINUE_COUNTER_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | continue_counter_1_and0 _vld | | Indicates which bits in [19:16] are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Continue_Counter_1_AN0 D_event | | <p>For two 16 bit counters mode: AND expression for AND event in continue counter event.</p> <p>For 32 bit counter mode : Invalid.</p> |
| 15:12 | R/W | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-------------------------------|------------------|---|
| 11:8 | R/W | Continue_counter_0_and_vld | | Indicates which bits in [3:0] are valid for considering AND event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Continue_Counter_0_AN_D_event | 0 | For two 16 bit counter mode: AND expression for AND event in continue Counter_0 event. For 32 bit counter mode: AND expression is valid for AND event in continue counter event. |

614 .Continue_Counter_And_Event Register Description

CT_CONTINUE_COUNTER_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|------------------------------|------------------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | continue_counter_1_or_0_vld | | Indicates which bits in [19:16] are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Continue_Counter_1_O_R_event | 0 | For two 16 bit counters mode: OR expression is valid for OR event in continue counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R/W | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | continue_counter_0_or_vld | | Indicates which bits in [3:0] are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Continue_Counter_0_O_R_event | 0 | For two 16 bit counter mode: OR expression is valid for OR event in continue Counter_0 event. For 32 bit counter mode: OR expression is valid for OR event in continue counter event. |

615 .Continue_Counter_Or_Event Register Description

CT_STOP_COUNTER_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|---------------------------|------------------|--|
| 31:22 | R | Reserved | 0 | Reserved for future use. |
| 21:16 | R/W | Stop_Counter_1_even_t_sel | 0 | For two 16 bit counters mode: Event select for Stopping the Counter_1. For 32 bit counter mode: Invalid. Please refer to events table for description. |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|------------------------------|------------------|---|
| 15:6 | R/W | Reserved | 0 | Reserved for future use. |
| 5:0 | R/W | Stop_Counter_0_even t_sel | 0 | For two 16 bit counters mode: Event select for Stopping the Counter_0. For 32 bit counter mode: Event select for Stopping counter. |

616 . Stop_Counter_Event_Sel Register Description

CT_STOP_COUNTER_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-------------------------------|------------------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | stop_counter_1_and_v ld | 0 | Indicates which bits in [19:16] are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Stop_Counter_1_AND_0 event | 0 | For two 16 bit counters mode: AND expression is valid for AND event in stop counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R/W | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | stop_counter_0_and_v ld | 0 | Indicates which bits in [3:0] are valid for considering AND event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Stop_Counter_0_AND_0 event | 0 | For two 16 bit counter mode: AND expression is valid for AND event in stop Counter_0 event. For 32 bit counter mode: AND expression is valid for AND event in stop counter event. |

617 . Stop_Counter_And_Event Register Description

CT_STOP_COUNTER_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|------------------------------|------------------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | stop_counter_1_or_vl d | 0 | Indicates which bits in [19:16] are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Stop_Counter_1_OR_0 event | 0 | For two 16 bit counters mode: OR expression is valid for OR event in Stop counter event. For 32 bit counter mode : Invalid. |

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | stop_counter_0_or_vld | | Indicates which bits in [3:0] are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Stop_Counter_0_OR_0 event | | For two 16 bit counter mode: OR expression is valid for OR event in Stop Counter_0 event. For 32 bit counter mode: OR expression is valid for OR event in Stop counter event. |

618 . Stop_Counter_Or_Event Register Description

CT_HALT_COUNTER_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------|-----------|---|
| 31:23 | R | Reserved | 0 | Reserved for future use. |
| 22 | W | Resume_from_halt_counter_1 | 0 | When halt event for counter1 is occurred, then counter1 went into halt mode. To resume this counter from halt mode, set this bit from f/w. If the counter1 entered into halt mode, there is effect of any event on counter1 till this is asserted. It is self clear bit. |
| 21:16 | R/W | Halt_Counter_1_event_0_sel | 0 | For two 16 bit counters mode: Event select for Halting the Counter_1. For 32 bit counter mode: Invalid. Please refer to events table for description. |
| 15:7 | R | Reserved | 0 | Reserved for future use. |
| 6 | W | Resume_from_halt_counter_0 | 0 | When halt event for counter0 is occurred, then counter0 went into halt mode. To resume this counter from halt mode, set this bit from f/w. If the counter0 entered into halt mode, there is effect of any event on counter0 till this is asserted. It is self clear bit. |
| 5:0 | R/W | Halt_Counter_0_event_0_sel | 0 | For two 16 bit counters mode: Event select for Halting the Counter_0. For 32 bit counter mode: Event select for Halting counter. |

619 . Halt_Counter_Event_Sel Register Description

CT_HALT_COUNTER_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | halt_counter_1_and_vld | | Indicates which bits in [19:16] are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Halt_Counter_1_AND_0 event | | For two 16 bit counters mode: AND expression is valid for AND event in stop counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | halt_counter_0_and_vld | | Indicates which bits in [3:0] are valid for considering AND event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Halt_Counter_0_AND_0 event | | For two 16 bit counter mode: AND expression is valid for AND event in stop Counter_0 event. For 32 bit counter mode: AND expression is valid for AND event in stop counter event. |

620 . Halt Counter And Event Register Description

CT_HALT_COUNTER_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | halt_counter_1_or_vld | 0 | Indicates which bits in [19:16] are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Halt_Counter_1_OR_e0 vent | | For two 16 bit counters mode: OR expression is valid for OR event in Halt counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | halt_counter_0_or_vld | | Indicates which bits in [3:0] are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------------|-----------|--|
| 3:0 | R/W | Halt_Counter_0_OR_event | 0 | For two 16 bit counter mode: OR expression is valid for OR event in Halt Counter_0 event. For 32 bit counter mode: OR expression is valid for OR event in Halt counter event. |

621 . Halt_Counter_Or_Event Register Description

CT_INCREMENT_COUNTER_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------------|-----------|--|
| 31:22 | R | Reserved | 0 | Reserved for future use. |
| 21:16 | R/W | Increment_Counter_1_0.event_sel | 0 | For two 16 bit counters mode: Event select for incrementing the Counter_1. For 32 bit counter mode: Invalid. Please refer to events table for description. |
| 15:6 | R | Reserved | 0 | Reserved for future use. |
| 5:0 | R/W | Increment_Counter_0_0.event_sel | 0 | For two 16 bit counters mode: Event select for incrementing the Counter_0. For 32 bit counter mode: Event select for incrementing counter. |

622 . Increment_Counter_Event_Sel Register Description

CT_INCREMENT_COUNTER_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | increment_counter_1_and_vld | 0 | Indicates which bits in [19:16] are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Increment_Counter_1_A_0.event | 0 | For two 16 bit counters mode: AND expression is valid for AND event in stop counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | increment_counter_0_and_vld | 0 | Indicates which bits in [3:0] are valid for considering AND event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------------|-----------|--|
| 3:0 | R/W | Increment_Counter_0_A ND_event | 0 | For two 16 bit counter mode: AND expression is valid for AND event in stop Counter_0 event. For 32 bit counter mode: AND expression is valid for AND event in stop counter event. |

623 . Increment_Counter_And_Event Register Description

CT_INCREMENT_COUNTER_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | increment_counter_1_or _vld | 0 | Indicates which bits in [19:16] are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Increment_Counter_1_O R_event | 0 | For two 16 bit counters mode: OR expression is valid for OR event in Increment counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | increment_counter_0_or _vld | 0 | Indicates which bits in [3:0] are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Increment_Counter_0_O R_event | 0 | For two 16 bit counter mode: OR expression is valid for OR event in Increment Counter_0 event. For 32 bit counter mode: OR expression is valid for OR event in Increment counter event. |

624 . Increment_Counter_Or_Event Register Description

CT_CAPTURE_COUNTER_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------------|-----------|---|
| 31:22 | R | Reserved | 0 | Reserved for future use. |
| 21:16 | R/W | Capture_Counter_1_e vent_sel | 0 | For two 16 bit counters mode: Event select for Capturing the Counter_1. For 32 bit counter mode: Invalid. Please refer to events table for description. |
| 15:6 | R | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------|-----------|---|
| 5:0 | R/W | Capture_Counter_0_event_sel | 0 | For two 16 bit counters mode: Event select for Capturing the Counter_0. For 32 bit counter mode: Event select for Capturing counter. |

625 .Capture_Counter_Event_Sel Register Description

CT_CAPTURE_COUNTER_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | capture_counter_1_and_vld | 0 | Indicates which bits in [19:16] are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Capture_Counter_1_AN_D_event | 0 | For two 16 bit counters mode: AND expression is valid for AND event in stop counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | capture_counter_0_and_vld | | Indicates which bits in [3:0] are valid for considering AND event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Capture_Counter_0_AN_D_event | 0 | For two 16 bit counter mode: AND expression for AND event in stop Counter_0 event. For 32 bit counter mode: AND expression for AND event in stop counter event. |

626 .Capture_Counter_And_Event Register Description

CT_CAPTURE_COUNTER_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------|-----------|---|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | capture_counter_1_or_vld | 0 | Indicates which bits in [19:16] are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Capture_Counter_1_OR_event | 0 | For two 16 bit counters mode: OR expression is valid for OR event in Capture counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------|-----------|--|
| 11:8 | R/W | capture_counter_0_or_vld | | Indicates which bits in [3:0] are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | Capture_Counter_0_OR_0_event | 0 | For two 16 bit counter mode: OR expression is valid for OR event in Capture Counter_0 event. For 32 bit counter mode: OR expression is valid for OR event in Capture counter event. |

627 .Capture Counter Or Event Register Description

CT_OUTPUT_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------|-----------|---|
| 31:22 | R | Reserved | 0 | Reserved for future use. |
| 21:16 | R/W | output_event_sel | 0 | For two 16 bit counters mode: Event select for output event from counter_0. For 32 bit counter mode: Invalid. Please refer to events table for description. |
| 15:6 | R | Reserved | 0 | Reserved for future use. |
| 5:0 | R/W | output_event_sel | 0 | For two 16 bit counters mode: Event select for output event from Counter_0. For 32 bit counter mode: Event select for output event |

628 .Output_Event_Sel Register Description

CT_OUTPUT_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | output_0_and_vld | 0 | Indicates which bits in [19:16] are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Output_1_AND_even_0_t | 0 | For two 16 bit counters mode: AND expression is valid for AND event in output counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | output_0_and_vld | | Indicates which bits in [3:0] are valid for considering AND event |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|---------------------|------------------|---|
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | output_0_AND_event0 | | For two 16 bit counter mode: AND expression for AND event in output Counter_0 event. For 32 bit counter mode AND expression for AND event in output counter event. |

629 .Output_And_Event Register Description

CT_OUTPUT_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-------------------|------------------|--|
| 31:28 | R | Reserved | 0 | Reserved for future use. |
| 27:24 | R/W | output_0_or_vld | 0 | Indicates which bits in [19:16] are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | Output_1_or_event | 0 | For two 16 bit counters mode: AND expression is valid for OR event in output counter event. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | 0 | Reserved for future use. |
| 11:8 | R/W | output_0_or_vld | | Indicates which bits in [3:0] are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | output_0_or_event | 0 | For two 16 bit counter mode: OR expression for OR event in output Counter_0 event. For 32 bit counter mode: OR expression for OR event in output counter event. |

630 .Output_Or_Event Register Description

CT_INTR_EVENT_SEL_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 31:22 | R | Reserved | | Reserved |
| 21:16 | R/W | intr_event_sel | 0 | For two 16 bit counters mode: Event select for interrupt event from counter_0. For 32 bit counter mode: Invalid. Please refer to events table for description. |
| 15:6 | R | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------|-----------|--|
| 5:0 | R/W | intr_event_sel | 0 | For two 16 bit counters mode: Event select for interrupt event from Counter_0. For 32 bit counter mode: Event select for output event |

631 .Intr_Event_Sel Register Description

CT_INTR_AND_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------|-----------|--|
| 31:28 | R | Reserved | | Reserved |
| 27:24 | R/W | intr_1_and_vld | 0 | Indicates which bits in 19:16 are valid for considering AND event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | intr_1_and_event | 0 | For two 16 bit counters mode: AND expression is valid for AND event in interrupt event for counter_1. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | | Reserved |
| 11:8 | R/W | intr_0_and_vld | | Indicates which bits in 3:0 are valid for considering AND event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | intr_0_and_event | 0 | For two 16 bit counter mode: AND expression for AND event in Counter_0 interrupt event. For 32 bit counter mode: AND expression for AND event in counter interrupt event. |

632 .Intr_And_Event Register Description

CT_INTR_OR_EVENT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------|-----------|---|
| 31:28 | R | Reserved | | Reserved |
| 27:24 | R/W | intr_1_or_vld | 0 | Indicates which bits in 19:16 are valid for considering OR event |
| 23:20 | R | Reserved | 0 | Reserved for future use. |
| 19:16 | R/W | intr_1_or_event | 0 | For two 16 bit counters mode: OR expression is valid for OR event in interrupt event for counter_1. For 32 bit counter mode : Invalid. |
| 15:12 | R | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------|-----------|--|
| 11:8 | R/W | intr_0_or_vld | | Indicates which bits in 3:0 are valid for considering OR event |
| 7:4 | R/W | Reserved | 0 | Reserved for future use. |
| 3:0 | R/W | intr_0_or_event | 0 | For two 16 bit counter mode: OR expression for OR event in Counter_0 interrupt event. For 32 bit counter mode: OR expression for OR event in counter interrupt event. |

633 .Intr_Or_Event Register Description

CT_RE_FE_RFE_LEV0LEV1_EVENT_ENABLE_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------|-----------|---|
| 31:20 | R | Reserved | | Reserved |
| 19:16 | R/W | Input_event_lev1_enable | 0xF | Input event level1 enables: 0001: input event0 is enabled 0010: input event1 is enabled 0011: input event0 and event1 are enabled |
| 15:12 | R/W | Input_event_lev0_enable | 0xF | Input event level0 enables: 0001: input event0 is enabled 0010: input event1 is enabled 0011: input event0 and event1 are enabled |
| 11:8 | R/W | Input_event_rfe_enable | 0xF | Input event rising and falling edge enables: 0001: input event0 is enabled 0010: input event1 is enabled 0011: input event0 and event1 are enabled |
| 7:4 | R/W | Input_event_fe_enable | 0xF | Input event falling edge enables: 0001: input event0 is enabled 0010: input event1 is enabled 0011: input event0 and event1 are enabled |
| 3:0 | R/W | Input_event_re_enable | 0xF | Input event rising edge enables: 0001: input event0 is enabled 0010: input event1 is enabled 0011: input event0 and event1 are enabled |

634 .RE_FE_RFE_Lev0_Lev1_Event_Enable Register Description

CT_MUX_SEL_0_REG

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|--|
| 3:0 | R/W | mux_sel_0 | 0 | Select value to select first output value “fifo_0_full[0]” out of all the “fifo_0_full_muxed” signals of counter_0 |

635 . Mux_Sel_0_Reg Description

CT_MUX_SEL_1_REG

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|--|
| 3:0 | R/W | mux_sel_1 | 0 | Select value to select first output value “fifo_0_full[1]” out of all the “fifo_0_full_muxed” signals of counter_0 |

636 . Mux_Sel_1_Reg Description

CT_MUX_SEL_2_REG

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|--|
| 3:0 | R/W | mux_sel_2 | 0 | Select value to select first output value “fifo_1_full[0]” out of all the “fifo_1_full_muxed” signals of counter_1 |

637 . Mux_Sel_2_Reg Description

CT_MUX_SEL_3_REG

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|--|
| 3:0 | R/W | mux_sel_3 | 0 | Select value to select first output value “fifo_1_full[0]” out of all the “fifo_1_full_muxed” signals of counter_1 |

638 . Mux_Sel_3_Reg Description

CT_OUTPUT_EVENT1_ADC_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------|-----------|--|
| 4:0 | R/W | output_event1_sel | 0 | Select signals to select one output event out of all the output events “output_event_0”, “output_event_1”, “output_event_2”, “output_event_3” which can be used as one of the two enables to enable ADC module |

639 . Output_Event1_ADC_sel Register Description

CT_OUTPUT_EVENT2_ADC_SEL_REG

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------|-----------|--|
| 4:0 | R/W | output_event2_sel | 0 | Select signals to select one output event out of all the output events “output_event_0”, “output_event_1”, “output_event_2”, “output_event_3” which can be used as one of the two enables to enable ADC module |

640 . Output_Event2_ADC_sel Register Description

16.4 CRC Accelerator

16.4.1 General Description

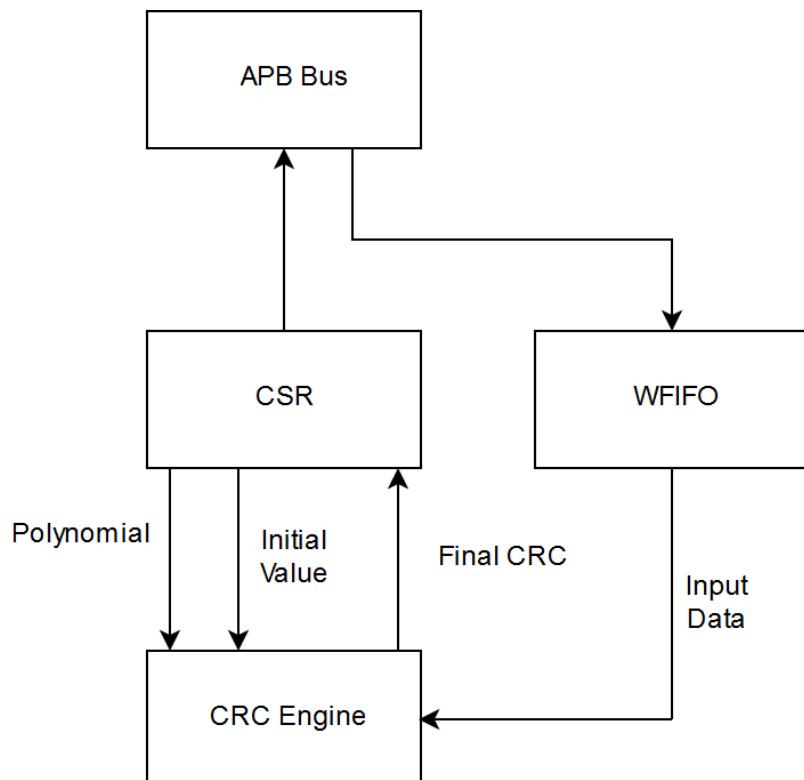
CRC (cyclic redundancy check) is used in a wide range of applications as a first data integrity check. The CRC Accelerator is present in MCU HP peripherals.

16.4.2 Features

- Supports 1-32 bit polynomials.
- Supports 1-32 bit stream-in data widths.
- Supports DMA flow control.

16.4.3 Functional Description

CRC accelerator computes CRC without any cycle penalty for any continuous data provided. It has 16x37 WFIFO, which holds 16 rows with 32 bit data and 5 bit length in each row. Zero indicates 32 bits of data is valid and the rest of values directly indicates the valid bits in each row. The LFSR is initialized with polynomial and initial value. Then input packet is read from WFIFO. After computation of CRC, status is updated on registers.



CRC accelerator block diagram

Programming Sequence for Computing CRC

1. Polynomial has to be loaded to POLYNOMIAL Register.
2. Polynomial width/length of the polynomial has to be loaded to polynomial_ctrl register. Clear the polynomial_width by writing 0x1f into polynomial_ctrl_reset register and write the actual width/length into polynomial_ctrl_set register.
3. LFSR_INIT has to be updated if required. Set LFSR_INIT_CTRL_SET [clear_lfsr] if LFSR required to be cleared. Set LFSR_INIT_CTRL_SET [init_lfsr] if LFSR_STATE required to be initialized.
4. Input data width :
 - a. Width from ULI: This is the default mode with out any configuration. Set DIN_CTRL_RESET [din_width_from_reg] and DIN_CTRL_RESET [din_width_from_cnt] to get the engine into this ULI width mode. In this mode, input data will be written to FIFO along with the length computed from the ULI_BE. For example if uli_be is 0xf (hsize is 2, 32 bit), a value 0x1f is written to FIFO as length token.
 - b. Width from reg: Set DIN_CTRL_SET [din_width_from_reg] and DIN_CTRL_RESET [din_width_from_cnt] to get the engine into this Width from reg mode. In this mode, input data will be written to FIFO along with the length from the DIN_CTRL [din_width]. For example if DIN_CTRL [din_width] is 0xf (hsize is 2, 32 bit), a value 0x1f is written to FIFO as length token.
 - c. Width from cnt: Set DIN_CTRL_SET [din_width_from_cnt] and DIN_CTRL_RESET [din_width_from_reg] to get the engine into this Width from cnt mode. Write the number of bytes that the packet/data has in total into DIN_NUM_BYTES register. In this mode, input data will be written to FIFO along with the length from either DIN_NUM_BYTES or ULI whichever is less. For example, if DIN_NUM_BYTES is 10 and a ULI_BE 0xf write happened, FIFO will be written with 0x1f as length token and DIN_NUM_BYTES will become 6. If DIN_NUM_BYTES is 1 and a ULI_BE 0x3 write happened, FIFO will be written with 0x7 as length token and DIN_NUM_BYTES will become 0.
5. FIFO indication signals afull and aempty, can be configured through DIN_CTRL register.
6. Write the data into the DIN_FIFO register which is indirectly mapped to FIFO input. Address incremented writes are not supported.
7. Monitor DIN_STS [calc_done] to get set to know the completion of computation of final CRC. In case of din_width_from_cnt, monitor DIN_STS [din_num_bytes_done] also to get set.
8. Read LFSR_STATE register for final CRC.

16.4.4 Register Summary

Base Address: 0x4508_0000

| Register Name | Offset | Description |
|-------------------------|--------|---|
| CRC_GEN_CTRL_SET | 0x0 | General control set register. Holds general control signals like soft_rst. This register is used only for setting the control signals. |
| CRC_GEN_CTRL_RESET | 0x4 | General control reset register. Holds general control signals like soft_rst. This register is used only for resetting the control signals. |
| CRC_GEN_STS | 0x8 | General status register. This holds the general status signals. |
| CRC_POLYNOMIAL | 0xC | Polynomial register. This register holds the polynomial that will be used to compute the final CRC. |
| CRC_POLYNOMIAL_CTRL_SET | 0x10 | Polynomial control set register. This register holds the control signals for polynomial like the valid length of polynomial. This register access can only set the bits in polynomial control register. |

| Register Name | Offset | Description |
|---------------------------|--------|---|
| CRC_POLYNOMIAL_CTRL_RESET | 0x14 | Polynomial control reset register. This register holds the control signals for polynomial like the valid length of polynomial. This register access can only reset the bits in polynomial control register. |
| CRC_LFSR_INIT_VAL | 0x18 | LFSR initialization value register. This register holds the initialization value of LFSR. |
| CRC_LFSR_INIT_CTRL_SET | 0x1C | LFSR initialization value control set register. This register holds the control signals for LFSR initialization. This register can only set the bits in the LFSR initialization control register. |
| CRC_LFSR_INIT_CTRL_RESET | 0x20 | LFSR initialization value control reset register. This register holds the control signals for LFSR initialization. This register can only reset the bits in the LFSR initialization control register. |
| CRC_DIN_FIFO | 0x24 | Input data FIFO is mapped under this address. Anything to be written to FIFO has to be written to this register. |
| CRC_DIN_CTRL_SET | 0x28 | Input data FIFO control set register. This holds the control signals required to handle the input data. This register can only set the control signals. |
| CRC_DIN_CTRL_RESET | 0x2C | Input data FIFO control reset register. This holds the control signals required to handle the input data. This register can only reset the control signals. |
| CRC_DIN_NUM_BYTES | 0x30 | Input data number of bytes. This holds the number of bytes that will be provided to compute the final CRC in one the input data modes. |
| CRC_DIN_STS | 0x34 | Input data status register. This holds the status indication signal like FIFO flow controls. |
| CRC_LFSR_STATE | 0x38 | LFSR state register. This holds the final CRC value. |

641 . Register Summary

16.4.5 Register Description

Legend:

R = Read-only, W = Write-only, RW = Read/Write, - = Reserved

CRC_GEN_CTRL_SET

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | RW | Reserved | 0 | Reserved for future use. |
| 0 | RW | soft_rst | 0* | Soft reset. This clears the FIFO and settles all the state machines to their IDLE. *- indicates self clear bit |

642 . GEN_CTRL_SET Register Description

CRC_GEN_CTRL_RESET

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--------------------------|
| 31:0 | RW | Reserved | 0 | Reserved for future use. |

643 . GEN_CTRL_RESET Register Description

CRC_GEN_STS

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------|-----------|---|
| 31:2 | R | Reserved | 0 | Reserved for future use. |
| 1 | R | din_num_bytes_done | 0 | When number of bytes requested for computation of final CRC is read from fifo by internal FSM, this will get set to 1, otherwise 0. |
| 0 | R | calc_done | 0 | When the computation of final CRC with the data out of fifo, this will get set to 1, otherwise 0. |

644 . GEN_STS Register Description

CRC_POLYNOMIAL

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|---|
| 31:0 | RW | Polynomial | 0 | Polynomial register. This register holds the polynomial with which the final CRC is computed. |

645 . POLYNOMIAL Register Description

CRC_POLYNOMIAL_CTRL_SET

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-----------|--|
| 31:5 | RW | Reserved | 0 | Reserved for future use. |
| 4:0 | RW | polynomial_width_set | | Polynomial width set. Number of bits/width of the polynomial has to be written here for the computation of final CRC. If a new width has to be configured, clear the existing length first by writing 0x1f in polynomial_ctrl_reset register. |

646 . POLYNOMIAL_CTRL_SET Register Description

CRC_POLYNOMIAL_CTRL_RESET

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--------------------------|
| 31:5 | RW | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|---|
| 4:0 | RW | polynomial_width_set | | <p>Polynomial width set. Number of bits/width of the polynomial has to be written here for the computation of final CRC.</p> <p>If a new width has to be configured, clear the existing length first by writing 0x1f in polynomial_ctrl_reset register.</p> |

647 . POLYNOMIAL_CTRL_RESET Register Description

CRC_LFSR_INIT_VAL

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------|-----------|--|
| 31:0 | RW | LFSR initialization value | 0 | This holds LFSR initialization value. When ever LFSR needs to be initialized, this has to be updated with the init value and trigger init_lfsr in LFSR_INIT_CTRL_SET register. |

648 . LFSR_INIT_VAL Register Description

CRC_LFSR_INIT_CTRL_SET

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-----------|--|
| 31:3 | RW | Reserved | 0 | Reserved for future use. |
| 2 | RW | use_swapped_init_val | 0 | <p>Use bit swapped init value. If this is set bit swapped version of LFSR init value will be loaded / initialized to LFSR state.</p> <p>1 – use_swapped_init_val will be enabled. 0 – No effect.</p> <p>When read 1 – use_swapped_init_val is enabled. 0 – use_swapped_init_val is disabled.</p> |
| 1 | RW | init_lfsr | 0* | <p>Initialize LFSR state. When this is set LFSR state will be initialized with LFSR_INIT_VAL/bit swapped LFSR_INIT_VAL in the next cycle.</p> <p>1 – Initialization will be done in next cycle. 0 – No effect.</p> <p>When read Read 0 always.</p> <p>* - indicates self clear bit</p> |
| 0 | RW | clear_lfsr | 0* | <p>Clear LFSR state. When this is set, LFSR state is cleared to 0.</p> <p>* - indicates self clear bit</p> |

649 . LFSR_INIT_CTRL_SET Register Description

CRC_LFSR_INIT_CTRL_RESET

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-----------|---|
| 31:3 | RW | Reserved | 0 | Reserved for future use. |
| 2 | RW | use_swapped_init_val | 0 | Use bit swapped init value. If this is set bit swapped version of LFSR init value will be loaded / initialized to LFSR state. 1 – use_swapped_init_val will be disabled. 0 – No effect. When read 1 – use_swapped_init_val is enabled. 0 – use_swapped_init_val is disabled. |
| 1 | RW | Reserved. | 0 | Reserved. |
| 0 | RW | Reserved. | 0 | Reserved. |

650 .LFSR_INIT_CTRL_RESET Register Description

CRC_DIN_FIFO

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:0 | W | DIN_FIFO | 0 | FIFO input port is mapped to this register. Data on which the final CRC has to be computed has to be loaded to this FIFO. |

651 .DIN_FIFO Register Description

CRC_DIN_CTRL_SET

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|---|
| 31:28 | RW | fifo_afull_threshold | 0 | FIFO almost full threshold value. This has to be cleared by writing 0xf0000000 into din_ctrl_reset before updating any new value. Bits which are 1s in the wdata[31:28] will be set in threshold value. When read Actual threshold value will be read. |
| 27:24 | RW | fifo_aempty_threshold | 0 | FIFO almost empty threshold value. This has to be cleared by writing 0x0f000000 into din_ctrl_reset before updating any new value. When write Bits which are 1s in the wdata[27:24] will be set in threshold value. When read Actual threshold value will be read |
| 23:9 | RW | Reserved | 0 | Reserved for future use. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|---|
| 8 | RW | reset_fifo_ptrs | 0* | <p>Reset fifo pointer. This clears the FIFO.</p> <p>When this is set, FIFO will be cleared.</p> <p>When write</p> <p>1 – FIFO will be cleared in the next cycle.</p> <p>0 – No effect.</p> <p>When read</p> <p>Read 0 always.</p> <p>*- indicates self clear bit</p> |
| 7 | RW | use_swapped_din | 0 | <p>Use bit swapped input data. If this is set, input data will be swapped and filled in to FIFO.</p> <p>Whatever read out from FIFO will be directly fed to LFSR engine.</p> <p>When write</p> <p>1 – Bit swapped data will be filled in to FIFO.</p> <p>0 – No effect.</p> <p>When read</p> <p>1 – Bit swapped data is filled in to FIFO.</p> <p>0 – Direct write data is filled in to FIFO.</p> |
| 6 | RW | din_width_from_cnt | 0 | <p>Valid number of bits in the input data. In default, number of valid bits in the input data is taken from ULI (uli_be).</p> <p>If this is set, a mix of ULI length and number of bytes remaining will form the valid bits (which ever is less that will be considered as valid bits).</p> <p>When write</p> <p>1 – Din width will be taken from both apb and cnt value.</p> <p>0 – No effect.</p> <p>When read</p> <p>1 – Din width is from ULI and cnt value.</p> <p>0 – Din width doesn't consider cnt value.</p> <p>This overrides the din_width_from_reg.</p> |
| 5 | RW | din_width_from_reg | 0 | <p>Valid number of bits in the input data. In default, number of valid bits in the input data is taken from ULI (uli_be).</p> <p>If this is set, whatever is the input size, only din_ctrl_reg[4:0] is taken as valid length/width for inout data.</p> <p>When write</p> <p>1 – Din valid width will be taken from reg.</p> <p>0 – No effect.</p> <p>When read</p> <p>1 – Din valid width is taken from reg.</p> <p>0 – Din valid width is not taken from reg.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------|-----------|--|
| 4:0 | RW | din_width_reg | 0 | <p>Valid number of bits in the input data in din_width_from_reg set mode.</p> <p>Before writing a new value into this, din_ctrl_reset_reg has to be written with 0x1f to clear this field as these are set/clear bits.</p> |

652 . DIN_CTRL_SET Register Description

CRC_DIN_CTRL_RESET

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------|-----------|--|
| 31:28 | RW | fifo_afull_threshold | 0 | <p>FIFO almost full threshold value. This has to be cleared by writing 0xf0000000 into din_ctrl_reset before updating any new value.</p> <p>When write Bits which are 1s in the wdata[31:28] will be reset in threshold value.</p> <p>When read Actual threshold value will be read.</p> |
| 27:24 | RW | fifo_aempty_threshold | 0 | <p>FIFO almost empty threshold value. This has to be cleared by writing 0x0f000000 into din_ctrl_reset before updating any new value.</p> <p>When write Bits which are 1s in the wdata[27:24] will be reset in threshold value.</p> <p>When read Actual threshold value will be read</p> |
| 23:9 | RW | Reserved | 0 | Reserved for future use. |
| 8 | RW | Reserved | 0 | Reserved. |
| 7 | RW | use_swapped_din | 0 | <p>Use bit swapped input data. If this is set, input data will be swapped and filled in to FIFO.</p> <p>Whatever read out from FIFO will be directly fed to LFSR engine.</p> <p>When write 1 – Direct data will be filled in to FIFO. 0 – No effect.</p> <p>When read 1 – Bit swapped data is filled in to FIFO. 0 – Direct write data is filled in to FIFO.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|---|
| 6 | RW | din_width_from_cnt | 0 | <p>Valid number of bits in the input data. In default, number of valid bits in the input data is taken from ULI (uli_be). If this is set, a mix of ULI length and number of bytes remaining will form the valid bits (which ever is less that will be considered as valid bits).</p> <p>When write</p> <p>1 – Din width won't consider cnt value. 0 – No effect.</p> <p>When read</p> <p>1 – Din width is from ULI and cnt value. 0 – Din width doesn't consider cnt value.</p> <p>This overrides the din_width_from_reg.</p> |
| 5 | RW | din_width_from_reg | 0 | <p>Valid number of bits in the input data. In default, number of valid bits in the input data is taken from ULI (uli_be). If this is set, whatever is the input size, only din_ctrl_reg[4:0] is taken as valid length/width for input data.</p> <p>When write</p> <p>1 – Din valid width will not taken from reg. 0 – No effect.</p> <p>When read</p> <p>1 – Din valid width is taken from reg. 0 – Din valid width is not taken from reg.</p> |
| 4:0 | RW | din_width_reg | 0 | Valid number of bits in the input data in din_width_from_reg set mode. Before writing a new value into this, din_ctrl_reset_reg has to be written with 0x1F to clear this field as these are set/clear bits. |

653 . DIN_CTRL_RESET Register Description

CRC_DIN_NUM_BYTES

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------|-----------|--|
| 31:0 | RW | din_num_bytes | 0 | <p>inout data number of bytes. In din_width_from_cnt mode, this register has to be loaded with the number of bytes that the entire packet contains.</p> <p>Write on to this address will update a down-counter running in the engine. For every ULI write operation on to the FIFO in the din_width_from_cnt mode, this counter will be manipulated by engine.</p> |

654 . DIN_NUM_BYTES Register Description

CRC_DIN_STS

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------|-----------|--------------------------------------|
| 31:10 | R | Reserved | 0 | Reserved for future use. |
| 9:4 | R | fifo_occ | 0 | FIFO occupancy. |
| 3 | R | fifo_full | 0 | FIFO full indication status. |
| 2 | R | fifo_afull | 0 | FIFO almost full indication status. |
| 1 | R | fifo_aempty | 1 | FIFO almost empty indication status. |
| 0 | R | fifo_empty | 1 | FIFO empty indication status. |

655 . DIN_STS Register Description

CRC_LFSR_STATE

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|--|
| 31:0 | RW | lfsr_state | 0 | For write operation, if LFSR dynamic loading is required this can be used for writing the LFSR state directly. For read operation, Final CRC/LFSR current state can be read through this address. |

656 . LFSR_STATE Register Description

16.5 eFuse Controller

16.5.1 General Description

The chipset provides 256 eFuse bits as a one-time programmable memory location. These bits use 32-bit addressing with each address containing 8 bits. The eFuse controller is used to program and read these bits. The 255th eFuse bit is programmed to 1'b1 and tested as part of manufacturing tests. Hence this bit has to be marked as Reserved with a default value to '1'.

16.5.2 Features

- Supports eFuse programming and read operations
- Supports memory mapped and FSM based read operation
- Supports Key Loading to security blocks

16.5.3 Functional Description

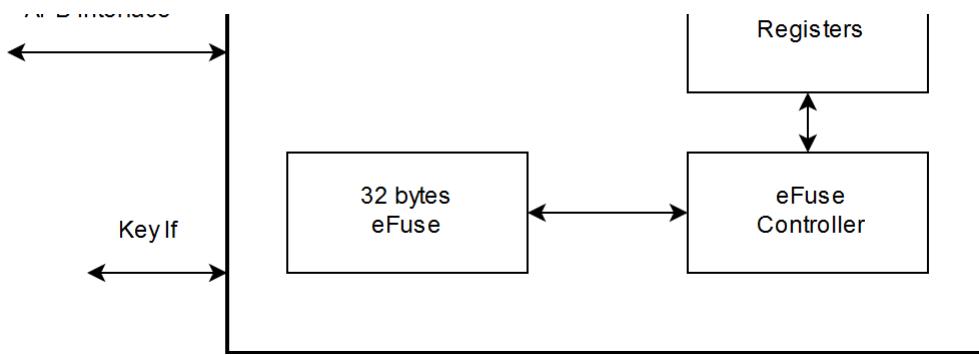
eFuse Controller block diagram is shown in Figure1. eFuse controller is used to write or read data from 32 bytes eFuse. Address lines A4~A0 are used to select one from 32 Bytes in eFuse. Remaining address lines A7~A5 are not used during read operation.

8 bits data is readout from eFuse at the same time during sensing. Address lines A7~A5 are used to select one of 8 bits in each byte during write operation. So address lines A7~A0 will select only one bit in eFuse during write operation.

A7~A0 need to be ready before the STROBE pulse is generated to avoid programming of the wrong data.

APR Interface





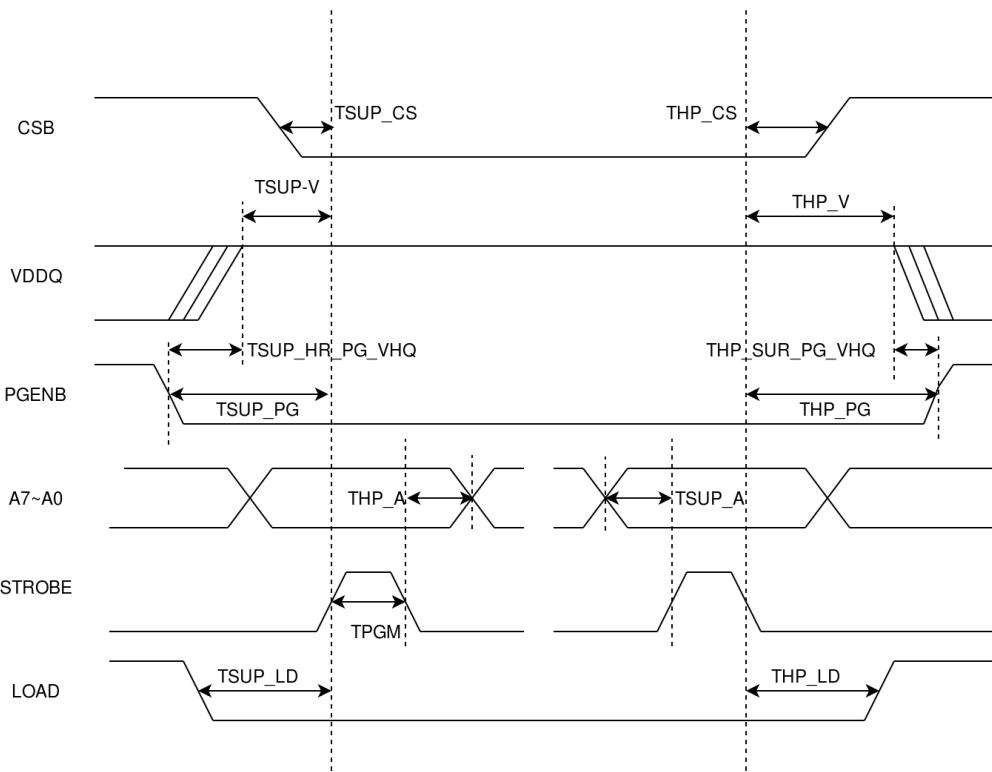
eFuse Controller Block Diagram

Programming Sequence

Write Operation

Each time, only one eFuse bit cell can be programmed by using this controller. Write operation is possible in direct access mode. The steps to be followed are specified below:

- Program ENABLE_EFUSE_WRITE and EFUSE_DA_ENABLE of EFUSE_CTRL_REG register to one
- Program 8 bit address, in EFUSE_DA_ADDR Register.
- Program CSB, PGENB, STROBE, and LOAD in EFUSE_DA_CTRL_REG with proper delays mention in below program mode timing spec table. Refer to the below example write operation for more details.
- Toggle the signals according to below timing diagram.



EFUSE Controller Timing Diagram for write mode

Program mode timing spec

| Parameter | Description | Min | Typ | Max | Units |
|-----------|--|-----|-----|-----|-------|
| TSUP_HR_ | PGENB falling edge to VDDQ rising | 6 | - | - | ns |
| PG_VQ | edge timing constraint during power on | | | | |
| THP_SUR_ | PGENB rising edge to VDDQ falling | 6 | - | - | ns |
| PG_VQ | edge timing constraint during power on | | | | |
| TSUP_CS | CSB to STROBE setup time into | 6 | - | - | ns |
| | Program mode | | | | |
| THP_CS | CSB to STROBE hold time out of | 6 | - | - | ns |
| | Program mode | | | | |
| TSUP_V | VDDQ to STROBE setup time into | 10 | - | - | ns |
| (Note 1) | Program mode | | | | |
| THP_V | VDDQ to STROBE hold time out of | 10 | - | - | ns |
| | Program mode | | | | |
| TSUP_PG | PGENB to STROBE setup time into | 16 | - | - | ns |
| | Program mode | | | | |
| THP_PG | PGENB to STROBE hold time into | 16 | - | - | ns |
| | Program mode | | | | |
| TPGM | Program mode STROBE pulse width | 1.8 | 2 | 2.2 | us |
| | high | | | | |
| TSUP_A | A11~A0 to STROBE setup time into | 6 | - | - | ns |
| | Program mode | | | | |
| THP_A | A11~A0 to STROBE hold time into | 6 | - | - | ns |
| | Program mode | | | | |
| TSUP_LD | LOAD to STROBE setup time into | 6 | - | - | ns |
| | Program mode | | | | |
| THP_LD | LOAD to STROBE hold time into | 6 | - | - | ns |
| | Program mode | | | | |

Write mode example:

- Set ENABLE_EFUSE_WRITE and EFUSE_DA_ENABLE of EFUSE_CTRL_REG register
- Set BIT(3), BIT(1) and BIT(0) in EFUSE_DA_CTRL_CLEAR_REG for asserting LOAD,CSB, PGENB
- Wait 6ns minimum time
- Load 8 bit address for M4EFUSE into EFUSE_DA_ADDR_REG

- Set BIT(2) to assert STROBE signal in EFUSE_DA_CTRL_SET_REG
- Load Strobe count value and strobe en bit in EFUSE_DA_CLR_STROBE_REG based on APB CLK frequency

Example :

If APB Clk =100MHz, Clock period $1/100 = 10\text{nsec}$

Delay for TPGM is 2us. Calculate no off cycles required for STROBE clear using below formula Delay/Clock period $2000/10 = 200$ cycles for 100MHz. Load this value in EFUSE_DA_CLR_STROBE_REG[8:0]

- Poll for Strobe bit clear in EFUSE_STATUS_REG register
- Set LOAD and CSB bits in EFUSE_DA_CTRL_SET_REG
- Wait 6nsec minimum time
- Set PGENB bit in EFUSE_DA_CTRL_SET_REG
- Clear BIT(2) bit in EFUSE_CTRL_REG

Read Operation

Read operation can be done in one of the following ways as mentioned below

Indirect access in EFUSE controller

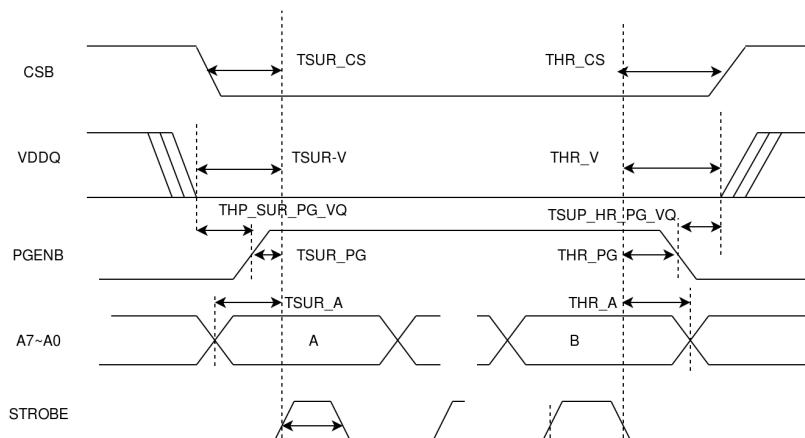
- Program EFUSE_RD_TMNG_PARAM_REG according to the apb bus clk frequency
- Program the 5-bit byte address in EFUSE_READ_ADDR_REG to read data
- Set EFUSE_ENABLE bit in EFUSE_CTRL_REG
- Wait for EFUSE_READ_FSM to become one in EFUSE_READ_DATA_REG
- EFUSE_READ_DATA of EFUSE_READ_DATA_REG will have the 8-bit read data

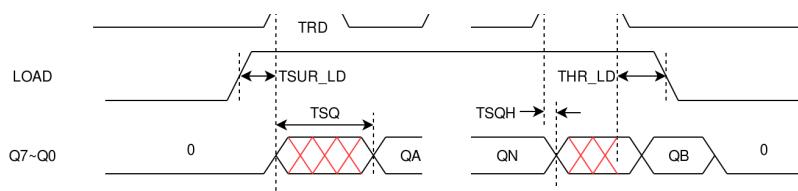
Direct access by registers

To program the eFuse in read mode, drive CSB to low, PGENB to high and LOAD to high. during this read operation, when the read Address A4~A0 is ready, a pulse needs to be sent on the STROBE pin.

Then read data to be sensed on data lines Q7~Q0. Read data is invalid once STROBE goes high.

- Program EFUSE_DA_ENABLE of EFUSE_CTRL_REG to one
- Program 5 bit byte address, in EFUSE_DA_ADDR Register (The MSB three bits should be zeros to enable byte addressing).
- Program CSB, PGENB, STROBE, and LOAD in EFUSE_DA_CTRL_REG with proper delays mention in below read mode timing spec table. Refer to the below example write operation for more details.
- Toggle the signals according to below shown timing diagram.
- Wait for Strobe bit clear in EFUSE_STATUS_REG[10]
- read 8 bit data from EFUSE_STATUS_REG





EFUSE Controller Timing Diagram for Read mode

Read mode timing spec

| Parameter | Description | Min | Typ | Max | Units |
|---------------|--|-----|-----|-----|-------|
| TSUP_HR_PG_VQ | PGENB falling edge to VDDQ rising edge timing constraint during power on | 6 | - | - | ns |
| THP_SUR_PG_VQ | PGENB rising edge to VDDQ falling edge timing constraint during power on | 6 | - | - | ns |
| TSUR_CS | CSB to STROBE setup time into Read mode | 6 | - | - | ns |
| THR_CS | CSB to STROBE hold time out of Read mode | 6 | - | - | ns |
| TSUR_V | VDDQ to STROBE setup time into Read mode | 12 | - | - | ns |
| THR_V | VDDQ to STROBE hold time out of Read mode | 12 | - | - | ns |
| TSUR_PG | PGENB to STROBE setup time into Read mode | 6 | - | - | ns |
| THR_PG | PGENB to STROBE hold time into Read mode | 6 | - | - | ns |
| TRD | Read mode STROBE pulse width high | 42 | - | - | ns |
| TSUR_A | A8~A0 to STROBE setup time into Read mode | 6 | - | - | ns |
| THR_A | A8~A0 to STROBE hold time into Read mode | 6 | - | - | ns |
| TSUR_LD | LOAD to STROBE setup time into Read mode | 6 | - | - | ns |
| THR_LD | LOAD to STROBE hold time into Read mode | 6 | - | - | ns |

| Parameter | Description | Min | Typ | Max | Units |
|-----------|--|-----|-----|-----|-------|
| TSQ | Q7~Q0 access time from STROBE rising edge [with output load: 0.1pF] | - | - | 42 | ns |
| TSQH | Q7~Q0 hold time to the next STROBE | - | - | 0 | ns |

Direct read mode example

- Set effuse_direct_access enable bit in EFUSE_CTRL_REG
- SET BIT(1) in EFUSE_DA_CTRL_SET_REG for CSB High
- Set BIT(3), BIT(2) and BIT(0) in EFUSE_DA_CTRL_CLEAR_REG for Lowering LOAD,STROBE, PGEB
- Wait 6ns minimum time
- SET BIT(1) in EFUSE_DA_CTRL_CLEAR_REG for CSB low
- Load 9 bit address for TA eFuse 5bit address for M4 eFuse into EFUSE_DA_ADDR_REG
- Set BIT(2) to set STROBE signal in EFUSE_DA_CTRL_SET_REG
- Load Strobe count value and strobe en bit in EFUSE_DA_CLR_STROBE_REG based on APB CLK frequency

Example :

If APB Clk =100MHz Clock period (1/100 = 10nsec)

Delay for TRD is minimum 42nsec. Calculate no off cycles required for STROBE clear using below formula
Delay/Clock period (42/10 = 4.2 cycles for 100MHz) Load result value in EFUSE_DA_CLR_STROBE_REG[8:0]

- Poll for Strobe bit clear in EFUSE_STATUS_REG
- Read data from EFUSE_STATUS_REG[9:2] bits
- Set LOAD and PGEB bits in EFUSE_DA_CTRL_SET_REG
- Wait for 6nsec minimum time
- Reset CSB in EFUSE_DA_CTRL_SET_REG

Memory mapped access

- Program EFUSE_RD_TMNG_PARAM_REG according to clock frequency used.
- Set EFUSE_ENABLE bit in EFUSE_CTRL_REG
- In Memory Mapped access, length of the transfer has to be programmed in the EFUSE_MEM_MAP_LEN register with the value either 1(16 Bit Read) or 0 (Default 8 bit Read).
- Address [13] is set then the memory mapped access is enabled else register access will be enabled.
- Address[4:0] are used as eFuse read address
- Read data is available on APB bus

16.5.4 Register Summary

Base Address: 0x4600_C000

| Register Name | Offset | Description |
|-------------------------|--------|---|
| EFUSE_DA_ADDR_REG | 0x00 | Address Register in Direct Access |
| EFUSE_DA_CTRL_SET_REG | 0x04 | Control Set Register in Direct Access |
| EFUSE_DA_CTRL_CLEAR_REG | 0x08 | Control Clear Register in Direct Access |
| EFUSE_CTRL_REG | 0x0C | eFuse Control Register |
| EFUSE_READ_ADDR_REG | 0x10 | Read address Register |

| Register Name | Offset | Description |
|------------------------------------|--------|--|
| EFUSE_READ_DATA_REG | 0x14 | Read Data Register |
| EFUSE_STATUS_REG | 0x18 | eFuse Status Register |
| EFUSE_RD_TMNG_PARAM_REG | 0x1C | Read Timing Parameter Register |
| EFUSE_MEM_MAP_LENGTH | 0x24 | Memory Mapped Length Register |
| EFUSE_READ_BLOCK_STARTING_LOCATION | 0x28 | Read Block Starting Location Register |
| EFUSE_READ_BLOCK_END_LOCATION | 0x2C | Read Block Ending Location Register |
| EFUSE_READ_BLOCK_ENABLE | 0x30 | Read Block Enable Register |
| EFUSE_DA_CLR_STROBE_REG | 0x34 | Direct Access Strobe Clearing Count Register |

657 . Register Summary

16.5.5 Register Description

Legend:

R = Read-only, W = Write-only, RW = Read/Write, - = Reserved

EFUSE_DA_ADDR_REG

| Bit | Access | Function | Reset Value | Description |
|---------|--------|-----------|-------------|--|
| [31:16] | - | Reserved | 0 | Reserved |
| [15:0] | RW | ADDR_BITS | 0 | These bits specifies the address to write or read from eFuse macro model |

658 . EFUSE_DA_ADDR_REG Description

EFUSE_DA_CTRL_SET_REG

| Bit | Access | Function | Reset Value | Description |
|--------|--------|----------------------------|-------------|---|
| [31:4] | R | Reserved | 0 | Reserved bits |
| 3 | RW | Set Load enable (LOAD) | 0 | 1 - Sets eFuse load enable (LOAD) pin when direct accessing is enabled 0 – no effect |
| 2 | R | Reserved | 0 | Reserved |
| 1 | RW | Set Chip Enable (CSB) | 1 | 1 - Sets eFuse Chip enable (CSB) pin when direct accessing is enabled 0 – no effect |
| 0 | RW | Set Program enable (PGENB) | 1 | 1 - Sets eFuse program enable (PGENB) pin when direct accessing is enabled 0 – no effect |

659 . EFUSE_DA_CTRL_SET_REG Description

EFUSE_DA_CTRL_CLEAR_REG

| Bit | Access | Function | Reset Value | Description |
|--------|--------|------------------------------|-------------|--|
| [31:4] | R | Reserved | 0 | Reserved bits |
| 3 | RW | Clear Load enable (LOAD) | 0 | 1 - Clear eFuse load enable (LOAD) pin when direct accessing is enabled 0 - no effect |
| 2 | R | Reserved | 0 | Reserved |
| 1 | RW | Clear Chip Enable (CSB) | 1 | 1 - Clear eFuse Chip enable (CSB) pin when direct accessing is enabled 0 - no effect |
| 0 | RW | Clear Program enable (PGENB) | 1 | 1 - Clear eFuse program enable (PGENB) pin when direct accessing is enabled 0 - no effect |

660 .EFUSE_DA_CTRL_CLEAR_REG Description

EFUSE_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|--------|--------|--------------------------|-------------|--|
| [31:3] | R | Reserved | 0 | Reserved bits |
| 2 | RW | enable_eFuse_write | 0 | Controls the switch on VDDIQ for eFuse read/write. 0 – VDDIQ is gated 1 – VDDIQ is supplied |
| 1 | RW | eFuse direct path enable | 0 | This bit specifies whether the eFuse direct path is enabled or not for direct accessing of the eFuse pins 1 – eFuse direct accessing enabled 0 - eFuse direct accessing disabled |
| 0 | RW | eFuse enable | 0 | This bit specifies whether the eFuse module is enabled or not 1 – eFuse module enabled 0 - eFuse module disabled |

661 .EFUSE_CTRL_REG Description

EFUSE_READ_ADDR_REG

| Bit | Access | Function | Reset Value | Description |
|----------|--------|----------|-------------|---|
| [31:16] | - | Reserved | 0 | Reserved |
| 15 | W | do_read | 1 | Enables read FSM after eFuse is enabled |
| [14: 13] | R | Reserved | 0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|--------|--------|-------------------|-------------|--|
| [12:0] | RW | Read address bits | 0 | These bits specifies the address from which read operation has to be performed |

662 .EFUSE_READ_ADDR_REG Description

EFUSE_READ_DATA_REG

| Bit | Access | Function | Reset Value | Description |
|---------|--------|----------------|-------------|---|
| [31:16] | - | Reserved | 0 | Reserved |
| 15 | R | Read FSM done | 0 | Indicates read FSM is done. After this read data is available in EFUSE_READ_DATA_REGISTER[7:0] |
| [14:8] | R | Reserved | 0 | Reserved bits |
| [7:0] | RW | Read data bits | 0 | These bits specifies the data bits that are read from a given address specified in the EFUSE_READ_ADDRESS_REGISTER[8:0] |

663 .EFUSE_READ_DATA_REG Description

EFUSE_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|---------|--------|------------------|-------------|--|
| [31:11] | R | Reserved | 0 | Reserved bits |
| 10 | R | Strobe Clear bit | 0 | This bit indicates STROBE signal goes low after strobe count value reached '0' |
| 9:2 | R | eFuse_dout_sync | 0 | This bit specifies the 8-bit data read out from the eFuse macro. This is synchronized with pclk. |
| 1 | R | Reserved | 0 | Reserved bit |
| 0 | R | eFuse Enabled | 0 | This bit specifies whether the eFuse is enabled or not 1 - eFuse enabled 0 - eFuse not enabled |

664 .EFUSE_STATUS_REG Description

EFUSE_RD_TMNG_PARAM_REG

| Bit | Access | Function | Reset Value | Description |
|---------|--------|---------------|-------------|--|
| [31:12] | R | Reserved bits | 0 | Reserved bits |
| [11:8] | RW | tHRA | 5 | for 32x8 macro: A4-A0 to STROBE hold time into Read mode 5122x8 macro: A8-A0 to STROBE hold time into Read mode |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|---|
| [7:4] | RW | tSQ | 2 | Q7-Q0 access time from STROBE rising edge |
| [3:0] | RW | tSUR_CS | 1 | CSB to STROBE setup time into read mode |

665 .EFUSE_RD_TMNG_PARAM_REG Description

EFUSE_MEM_MAP_LENGTH

| Bit | Access | Function | Reset Value | Description |
|--------|--------|-----------------------|-------------|---------------------------------|
| [31:1] | R | Reserved bits | 0 | Reserved bits |
| 0 | RW | eFuse_mem_map_length0 | | 0: 8 bit read 1: 16 bit read |

666 .EFUSE_MEM_MAP_LENGTH Description

EFUSE_READ_BLOCK_STARTING_LOCATION

| Bit | Access | Function | Reset Value | Description |
|---------|--------|--------------------------------|-------------|---|
| [31:16] | - | Reserved | 0 | Reserved |
| [15:0] | RW | eFuse_read_block_starting_addr | | Starting address from which the read has to be blocked. Once the end address is written, it cannot be changed till power on reset is given. |

667 .EFUSE_READ_BLOCK_STARTING_LOCATION Description

EFUSE_READ_BLOCK_END_LOCATION

| Bit | Access | Function | Reset Value | Description |
|---------|--------|---------------------------|-------------|---|
| [31:16] | - | Reserved | 0 | Reserved |
| [15:0] | RW | eFuse_read_block_end_addr | 0 | End address till which the read has to be blocked. Once the end address is written , it cannot be changed till power on reset is given. |

668 .EFUSE_READ_BLOCK_END_LOCATION Description

EFUSE_READ_BLOCK_ENABLE

| Bit | Access | Function | Reset Value | Description |
|--------|--------|---------------|-------------|---------------|
| [31:1] | R | Reserved bits | 0 | Reserved bits |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-------------------------|--------------------|--|
| 0 | RW | eFuse_read_block_enable | 0 | <p>Enable for blocking the read access from a programmable memory location (Start and end address register).</p> <p>Once the blocking is enabled , it cannot be disabled till power on reset is given.</p> |

669 .EFUSE_READ_BLOCK_ENABLE Description

EFUSE_DA_CLR_STROBE_REG

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------------|--------------------|---|
| [31:10] | R | Reserved bits | 0 | Reserved bits |
| 9 | RW | eFuse_strobe_en | 0 | Self clear bit. 1'b1 : Enable Strobe signal |
| [8:0] | RW | eFuse_strobe_clr_cnt1 | | Strobe signal Clear count in direct access mode. value depends on APB clock frequency of eFuse controller |

670 .EFUSE_DA_CLR_STROBE_REG Description

16.6 Enhanced GPIO (EGPIO)

16.6.1 General Description

The Enhanced GPIO functionality is used to configure the functionality on GPIO pins. There is one instance in MCU HP Domain which is used to control the SoC GPIOs (GPIO_n; n=0:63) and another in MCU ULP Domain which is used to control the ULP GPIO's (ULP_GPIO_n; n=0:15). The features and functionality is same for both the instances except for the Register Base Address.

All GPIO pins in the MCU HP / MCU ULP Domains are grouped into multiple ports. Each port consists of a maximum 16 pins. The ports provide access to multiple GPIO pins at once. They also support set, clear and toggle features. Each pin can be programmed as an output or as an input port for various functions.

16.6.2 Features

The key features of the EGPIO are listed below:

- Supports various alternate functions like set, clear, toggle on all the pins
- Option to program mode and direction for the each GPIO pin independently
- Supports edge and level detection based on which interrupts will be raised.
- MCU HP EGPIOS support 8 pin, 4 wakeup and 4 group interrupts
- MCU ULP EGPIOS support 8 pin, 2 wakeup and 2 group interrupts

16.6.3 Functional Description

The EGPIO is encapsulates the GPIO port, interrupt and configuration register related logic.

The interrupt block encapsulates the group interrupt, wakeup interrupt and pin interrupt logic.

The port set, clear, toggle, masked load, masked read ,read, bit load and word load functions are implemented in this block.

The GPIO_n (n=25:30) needs to be enabled before using them as GPIOs as per the functionality described below. These GPIOs can be enabled through GPIO_25_30_CONFIG_REG Register.

! Please note that GPIO_n (n=0:5) are dedicated for the Secure Zone Processor's Flash interface. The MCU should NOT be changing any configuration related to these GPIOs under any circumstances since it may lead to the Flash content being corrupted, rendering the chip unusable. This is applicable to MCU HP EGPI0 Instance.

GPIO Port

GPIO port provides access to multiple GPIO pins. This supports set, clear, toggle, read, masked load, masked read, bit load/read and word load/read functions on the port pins. Five GPIO ports are present in EGPI0. port_load_reg and mask_reg registers are present in each GPIO port. The port_load_reg holds the GPIO data to be put on the GPIO output lines. When a load happens on the APB to any of the set, clear, toggle, etc register address spaces, the bits in the port load register are changed accordingly.

Registers affecting multiple port pins at a time:

- port load
- set
- clear
- toggle
- masked load
- masked read
- port read

Registers affecting single pin at a time:

- bit load/read
- word load/read

Data to be put on GPIO output lines has to be loaded into port load register. When a write is made to the set register, the bits positions which are set in the write data are set in the port load register. When a write is made to the clear register, the bits positions which are set in the write data are cleared in the port load register. When a write is made to the toggle register, the bits positions which are set in the write data are toggled in the port load register. When a write is made to the masked load register, only the bit positions that are not masked in the mask register are copied to the port load register as is. Other bits are not altered. If a bit in the mask register is set to '0', it is masked. Reading a masked load register provides the status of the GPIO input lines that are not masked in mask register. Other bits appear as '0'.

The bit load and word load registers affect only a single pin at a time. When bit load register of a pin is written, the value in the 0th position of the bit load write data is loaded into the corresponding bit position in the port load register. When a non-zero value is written into the word load register of a pin, the corresponding pin bit in the port load register is set. When a zero is written, the corresponding bit is reset.

Programming sequence

Following is the programming sequence to use EGPIOs.

1. **Clock enable:**
 - a. Enable clock as described in MCU HP Clock Architecture or MCU ULP Clock Architecture Sections.
2. **GPIO-OEN(Direction)/MODE**
 - a. Program mode of all gpio's as 0. Program 5:2 bits as 0 in all GPIO_CONFIG_REG_* registers
 - b. Program OEN (BIT(0) as 0 for all GPIO_REG_* registers
3. **GPIO set/clear/toggling**
 - a. Program PORT_SET_REG_n for all the 5 ports. This will set all gpio's output as 1
 - b. Program PORT_CLEAR_REG_n for all the 5 ports as 0xFFFF. This will set all gpio's output as 0
4. **REN programming**
 - a. Program REN of all the GPIO's to be 1 in Pad Configuration registers.

- b. ULP GPIO's REN has to be programmed as per ULP GPIO Pad Configuration registers
5. **GPIO Status Read:**
- a. Program OEN BIT(0) as 1 for all GPIO registers. EGPIOS are programmed in input mode.
 - b. Poll for PORT_READ_REG_n to get the proper GPIO status.
 - c. Follow the same procedure all bits for all 5 EGPIOS to get the status.

Following is the programming sequence to use EGPIOS in open drain mode.

1. **Clock enable:**
 - a. Enable clock as described in MCU HP Clock Architecture or MCU ULP Clock Architecture Sections.
2. **GPIO-OEN(Direction)/MODE**
 - a. Program mode of all gpio's as 0. Program 5:2 bits as 0 in all GPIO_CONFIG_REG_* registers
 - b. Program OEN (BIT(0) as 1 for all GPIO_REG_* registers. Open drain is observed on all gpio's output.
 - c. Program OEN (BIT(0) as 0 for all GPIO_REG_* registers. This will set all gpio's output as 0.
3. **GPIO clear**
 - a. Program PORT_CLEAR_REG_n for all the 5 ports as 0xFFFF.
4. **REN programming**
 - a. Program REN of all the GPIO's to be 0 in Pad Configuration registers.

GPIO interrupt generation

The GPIO inputs are monitors and generates interrupts when the programmed pattern is detected. It generates 8 pin interrupts, 4 group interrupts and 4 wakeup interrupts.

Registers present for interrupt generation are:

- pin_intr_ctrl -> Control registers for 8 pin interrupts
- pin_intr_sts -> Status registers for pin interrupts.
- group_intr_ctrl --> Control registers for 4 group and 4 wakeup interrupts
- group_intr_sts --> Status registers for 4 group and 4 wakeup interrupts

The pin interrupt control registers provide enables and mask for each pin interrupt. Any of the 8 GPIO pins can be mapped on these 8 pin interrupts. The GPIO pin to be mapped to the interrupt line has to be configured in the pin_intr_ctrl register of the respective interrupt pin. Each of these GPIOs are monitors and raises an interrupt when the programmed pattern is detected. For pin interrupts, rise edge, fall edge and level interrupt status is mapped to the respective pin's status register.

The group and the wakeup interrupts are both generated by monitoring the status of multiple pins. The difference between the interrupts lies in the GPIO input pins considered for interrupt generation. The group interrupt generation is based on synchronized interrupt pins whereas wakeup interrupt generation considers unsynchronized pins. The control information as to which GPIO lines to consider for interrupts generation is present in the group interrupt control registers for both group and wakeup interrupts. The masking logic is also present. The enables for the GPIO pins to be considered and the polarity of the pin that contributes to interrupt are provided. For group interrupts, unmasked versions of the group interrupts and synchronized version of the wakeup interrupts are mapped to status registers. The wakeup interrupt itself is generated based on unsynchronized pins, i.e., clock need not be present for the generation of this interrupt. When clock is not present, the bit in the status register will not be updated. If the wakeup condition persists on the GPIO input lines till the clock is provided, then the synchronized wakeup interrupt bit gets updated. wakeup interrupt can be used to wake up the chip from sleep state.

16.6.4 Register Summary

Base Address : 0x4600_8000

| Register Name | Offset | Description |
|-----------------------|--------|--|
| GPIO_25_30_CONFIG_REG | 0x0C | Configuration Register for GPIO_n(n=25:30) |

Base Address for MCU HP Instance: 0x4613_0000

Base Address for MCU ULP Instance: 0x2404_C000

| Register Name | Offset | Description |
|----------------------|--------|-------------------------------|
| GPIO_CONFIG_REG_0 | 0x00 | GPIO Configuration Register 0 |
| GPIO_CONFIG_REG_1 | 0x10 | GPIO Configuration Register 1 |
| GPIO_CONFIG_REG_2 | 0x20 | GPIO Configuration Register 2 |
| GPIO_CONFIG_REG_3 | 0x30 | GPIO Configuration Register 3 |
| GPIO_CONFIG_REG_4 | 0x40 | GPIO Configuration Register 4 |
| GPIO_CONFIG_REG_5 | 0x50 | GPIO Configuration Register 5 |
| GPIO_CONFIG_REG_6 | 0x60 | GPIO Configuration Register 6 |
| GPIO_CONFIG_REG_7 | 0x70 | GPIO Configuration Register 7 |
| BIT_LOAD_REG_0 | 0x4 | Bit Load Register 0 |
| BIT_LOAD_REG_1 | 0x14 | Bit Load Register 1 |
| BIT_LOAD_REG_2 | 0x24 | Bit Load Register 2 |
| BIT_LOAD_REG_3 | 0x34 | Bit Load Register 3 |
| BIT_LOAD_REG_4 | 0x44 | Bit Load Register 4 |
| BIT_LOAD_REG_5 | 0x54 | Bit Load Register 5 |
| BIT_LOAD_REG_6 | 0x64 | Bit Load Register 6 |
| BIT_LOAD_REG_7 | 0x74 | Bit Load Register 7 |
| WORD_LOAD_REG_0 | 0x8 | Word Load Register 0 |
| WORD_LOAD_REG_1 | 0x18 | Word Load Register 1 |
| WORD_LOAD_REG_2 | 0x28 | Word Load Register 2 |
| WORD_LOAD_REG_3 | 0x38 | Word Load Register 3 |
| WORD_LOAD_REG_4 | 0x48 | Word Load Register 4 |
| WORD_LOAD_REG_5 | 0x58 | Word Load Register 5 |
| WORD_LOAD_REG_6 | 0x68 | Word Load Register 6 |
| WORD_LOAD_REG_7 | 0x78 | Word Load Register 7 |
| PORT_LOAD_REG | 0x1000 | Port Load Register |
| PORT_SET_REG | 0x1004 | Port Set Register |
| PORT_CLR_REG | 0x1008 | Port Clear Register |
| PORT_MASKED_LOAD_REG | 0x100c | Port Masked Load Register |
| PORT_TOGGLE_REG | 0x1010 | Port Toggle Register |

| Register Name | Offset | Description |
|----------------------|---------------|---|
| PORT_READ_REG | 0x1014 | Port Read Register |
| GPIO_INTR_CTRL_0 | 0x1200 | GPIO Interrupt Control Register 0 |
| GPIO_INTR_CTRL_1 | 0x1208 | GPIO Interrupt Control Register 1 |
| GPIO_INTR_CTRL_2 | 0x1210 | GPIO Interrupt Control Register 2 |
| GPIO_INTR_CTRL_3 | 0x1218 | GPIO Interrupt Control Register 3 |
| GPIO_INTR_CTRL_4 | 0x1220 | GPIO Interrupt Control Register 4 |
| GPIO_INTR_CTRL_5 | 0x1228 | GPIO Interrupt Control Register 5 |
| GPIO_INTR_CTRL_6 | 0x1230 | GPIO Interrupt Control Register 6 |
| GPIO_INTR_CTRL_7 | 0x1238 | GPIO Interrupt Control Register 7 |
| GPIO_INTR_STAT_0 | 0x1204 | GPIO Interrupt Status Register 0 |
| GPIO_INTR_STAT_1 | 0x120C | GPIO Interrupt Status Register 1 |
| GPIO_INTR_STAT_2 | 0x1214 | GPIO Interrupt Status Register 2 |
| GPIO_INTR_STAT_3 | 0x121C | GPIO Interrupt Status Register 3 |
| GPIO_INTR_STAT_4 | 0x1224 | GPIO Interrupt Status Register 4 |
| GPIO_INTR_STAT_5 | 0x122C | GPIO Interrupt Status Register 5 |
| GPIO_INTR_STAT_6 | 0x1230 | GPIO Interrupt Status Register 6 |
| GPIO_INTR_STAT_7 | 0x123C | GPIO Interrupt Status Register 7 |
| GPIO_GRP_INTR_CTRL_0 | 0x1240 | GPIO Group Interrupt Control Register 0 |
| GPIO_GRP_INTR_STS_0 | 0x1244 | GPIO Group Interrupt Status Register 0 |
| GPIO_GRP_INTR_CTRL_1 | 0x1248 | GPIO Group Interrupt Control Register 1 |
| GPIO_GRP_INTR_STS_1 | 0x124C | GPIO Group Interrupt Status Register 1 |
| GPIO_GRP_INTR_CTRL_2 | 0x1250 | GPIO Group Interrupt Control Register 2 |
| GPIO_GRP_INTR_STS_2 | 0x1254 | GPIO Group Interrupt Status Register 2 |
| GPIO_GRP_INTR_CTRL_3 | 0x1258 | GPIO Group Interrupt Control Register 3 |
| GPIO_GRP_INTR_STS_3 | 0x125C | GPIO Group Interrupt Status Register 3 |

671 . Register Summary Table

16.6.5 Register Description

GPIO_25_30_CONFIG_REG

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--------------------|
| 15:11 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------|-------------|---|
| 10 | R/W | GPIO_25_30_EN | 0 | Writing 1 to this enables the functionality for GPIO_n (n=25:30). |
| 7:6 | R | Reserved | 0 | Reserved |

672 .GPIO_25_30_CONFIG_REG Description

GPIO_CONFIG_REG_n

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------------|-------------|---|
| 15:12 | R | Reserved | 0x0 | Reserved |
| 11 | R/W | Group Interrupt 2 polarity | 0 | Decides the active value of the pin to be considered for group interrupt 2 generation when enabled'0' – group interrupt gets generated when gpio input pin status is zero'1' – grp interrupt gets generated when gpio input pin status is '1' |
| 10 | R/W | Group Interrupt 2 enable | 0 | When set, the corresponding GPIO is pin is selected for group intr 2 generation |
| 9 | R/W | Group Interrupt 1 Polarity | 0 | Decides the active value of the pin to be considered for group interrupt 1 generation when enabled'0' – group interrupt gets generated when gpio input pin status is zero'1' – grp interrupt gets generated when gpio input pin status is '1' |
| 8 | R/W | Group Interrupt 1 enable | 0 | When set, the corresponding GPIO is pin is selected for group intr 1 generation |
| 7:6 | R | Reserved | 0x0 | Reserved |
| 5:2 | R/W | Mode | 0x0 | GPIO Pin Mode. Ranges 0000 -> Mode 0 to 1111 -> Mode 15 Used for GPIO Pin Muxing |
| 1 | R/W | Port Mask | 0 | Port mask value '1' – When set, pin is masked when written/read through PORT MASK REG. |
| 0 | R/W | Direction | 1 | Direction of the GPIO pin. '1' – INPUT, '0' – OUTPUT |

673 .GPIO_CONFIG_REG_n Description

BIT_LOAD_REG_n

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|---|
| 0 | R/W | Bit Load | N/A | Loads 0th bit on to the pin on write. And reads the value on pin on read into 0th bit |

674 . BIT_LOAD_REG_n Description

WORD_LOAD_REG_n

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------|-------------|--|
| 15:0 | R/W | Word Load | N/A | Loads 1 on the pin when any of the bit in load value is 1. On read, pass the bit status into all bits. |

675 . WORD_LOAD_REG_n Description

PART_LOAD_REG_n

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------|-------------|---|
| 15:0 | R/W | Port Load | 0x0 | Loads the value on to pin on write. And reads the value of load register on read. |

676 . PORT_LOAD_REG_n Description

PART_MASKED_LOAD_REG_n

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------|-------------|---|
| 15:0 | R/W | Port Masked Load | 0x0 | Only loads into pins which are not masked. On read, pass only status unmasked pins(Mask present in GPIO CONFIG REG) |

677 . PORT_MASKED_LOAD_REG_n Description

PART_SET_REG_n

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 15:0 | W | Port Set | N/A | Sets the pin when corresponding bit is high. Writing zero has no effect. |

678 . PORT_SET_REG_n Description

PORT_CLEAR_REG_n

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| 15:0 | W | Port Clear | N/A | Clears the pin when corresponding bit is high. Writing zero has no effect. |

679 .PORT_CLEAR_REG_n Description

PORT_TOGGLE_REG_n

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| 15:0 | W | Port Toggle | N/A | Toggles the pin when corresponding bit is high. Writing zero has not effect. |

680 .PORT_TOGGLE_REG_n Description

PORT_READ_REG_n

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-------------------|--------------------|--|
| 15:0 | R | Port Read (Input) | 0x0 | Reads the value on GPIO pins irrespective of the pin mode. |

681 .PORT_READ_REG_n Description

GPIO_INTR_CTRL_REG_n

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-------------------|--------------------|---|
| 13:12 | R/W | Port Number | 0x0 | GPIO Port to be chosen for interrupt generation. |
| 11:8 | R/W | Pin Number | 0x0 | GPIO Pin to be chosen for interrupt generation |
| 7:5 | R | Reserved | 0x0 | Reserved |
| 4 | R/W | Mask | 1 | Masks the interrupt. Interrupt will still be seen in status register when enabled'1' – intr masked'0' – intr unmasked |
| 3 | R/W | Fall Edge Enable | 0 | enables interrupt generation when falling edge is detected on pin'1' – intr enabled'0' – disabled |
| 2 | R/W | Rise Edge Enable | 0 | enables interrupt generation when rising edge is detected on pin'1' – intr enabled '0' – disabled |
| 1 | R/W | Level Low Enable | 0 | enables interrupt generation when pin level is '0'"1' – intr enabled'0' – disabled |
| 0 | R/W | Level High Enable | 0 | enables interrupt generation when pin level is '1"1' – intr enabled'0' – disabled |

682 .GPIO_INTR_CTRL_REG_n Description

GPIO_INTR_STATUS_REG_n

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------|-------------|---|
| 4 | W | Mask Clear | 0 | When 1 is written mask bit gets cleared. On read, this bit should result it in 0 |
| 3 | W | Mask Set | 0 | When 1 is written mask bit will get set. On read, this bit should result it in 0 |
| 2 | R/W | Fall Edge Status | 0 | Gets set when fall edge is enabled and occurs. When 1 is written it gets cleared. Writing 0 has not effect |
| 1 | R/W | Rise Edge Status | 0 | Gets set when rise edge is enabled and occurs. When 1 is written it gets cleared. Writing 0 has not effect |
| 0 | R/W | Interrupt Status | 0 | Gets set when interrupt is enabled and occurs. When 1 is written it gets cleared. Also clears rise edge and fall edge status bits. Writing 0 has not effect |

683 .GPIO_INTR_STATUS_REG_n Description

GPIO_GRP_INTR_n_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------|-------------|---|
| 4 | R/W | Mask | 1 | 1-mask,0-unmask (Not used for wakeup interrupts) |
| 3 | R/W | Enable Interrupt | 0 | 1-enable normal group interrupt, 0-disable normal group interrupt (Wakeup interrupt is unaffected by this bit) |
| 2 | R/W | Enable Wakeup | 0 | For wakeup generation, actual pin status has to be seen(before double ranking point) Set to 1 if grp interrupt has to be used as wakeup source |
| 1 | R/W | Level/Edge | 0 | 1- Edge(cannot be used as wakeup) 0 - Level |
| 0 | R/W | AND/OR | 0 | 0 - AND 1- Or |

684 .GPIO_GRP_INTR_n_CTRL_REG Description

GPIO_GRP_INTR_n_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------|-------------|---|
| 4 | W | Mask Clear | 0 | clear bit version of Mask bit in PORT_GRP_INTR_n_CTRL_REG_N . Gives zero on read |
| 3 | W | Mask Set | 0 | Set bit version of Mask bit in PORT_GRP_INTR_n_CTRL_REG_N . Gives zero on read |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------|-------------|---|
| 2 | N/A | Reserved | 0 | Reserved |
| 1 | R | Wakeup | 0 | Double ranked version of wakeup. Gets set when wakeup is enabled and occurs. |
| 0 | R/W | Interrupt Status | 0 | Interrupt status is available in this bit when interrupt is enabled and generated. When '1' is written, interrupt gets cleared. |

685.GPIO_GRP_INTR_n_STATUS_REG Description

16.7 Generic SPI Master

16.7.1 General Description

The Generic SPI Master is present in MCU HP peripherals. It provides an I/O interface to a wide variety of SPI compatible peripheral devices. SPI is a synchronous four-wire interface consisting of two data pins(MOSI, MISO), a device select pin (CSN) and a gated clock pin(SCLK). With the two data pins, it allows for full-duplex operation to other SPI compatible devices. Typical SPI compatible peripheral devices that can be used to interface are:

- LCD displays
- A/D converters
- D/A converters
- Codecs
- Micro-controllers
- Flash

16.7.2 Features

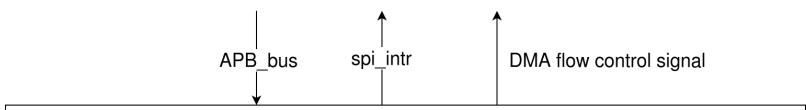
- Supports full duplex Single-bit SPI master mode.
- Support for Mode-0 and Mode-3 (Motorola)
- Supports both Full speed and High speed modes.
- Connect upto four SPI peripheral devices
- SPI clock out is programmable to meet required baud rates
- Generates interrupt for different events.

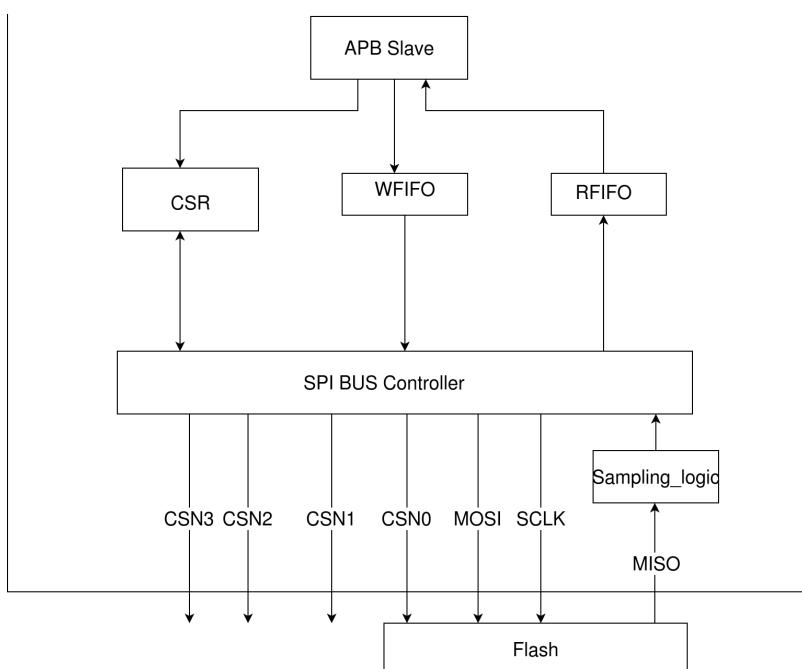
The following features of the Generic SPI Master help reducing the load on the processor:

- Support upto 32K bytes of read data from a SPI device in a single read operation.
- Support for byte-wise swapping of read and write data.
- Programmable FIFO thresholds with maximum FIFO depth of 16 and support for DMA

16.7.3 Functional Description

SPI has option to select four SPI devices by using respective CSN. CSN is active low signal. It is asserted during either write operation or read operation. The SCLK signal is a gated clock that is only active during data transfers for the duration of the transferred word. The number of active edges is equal to the number of bits driven on the data lines. The clock rate is determined by the 8-bit value of GSPI_CLK_DIVISION_FACTOR register. SCLK, CSN and MOSI are output signals and MISO is input signal. The SCLK signal is used to shift out and shift in the data driven onto the MISO and MOSI lines. The data is always shifted out on neg-edge of the clock and sampled on the either pos-edge or neg-edge of the clock depending on full speed mode or high speed mode respectively. Full speed and High speed modes are configured using GSPI_BUS_MODE[0] before start of any SPI transactions. Similarly, mode0 and mode3 (clock polarity) are programmable by asserting bit in GSPI_BUS_MODE register initially. SPI controller block diagram is shown in Figure1.





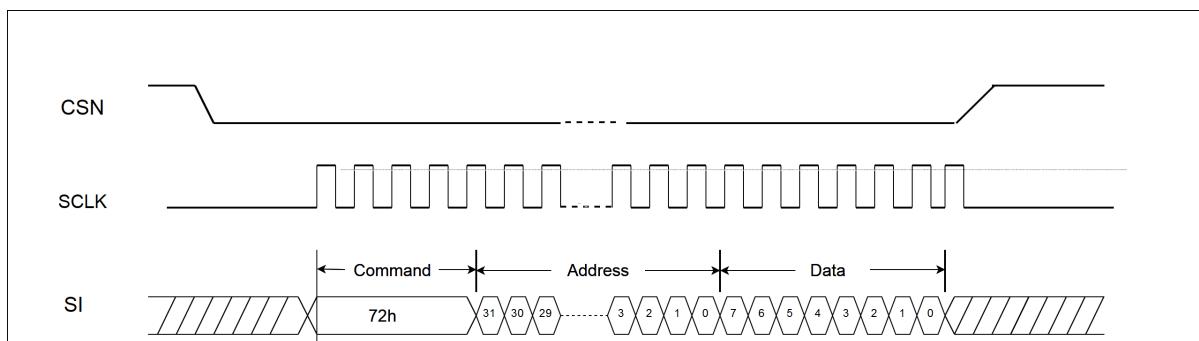
General SPI block diagram

Programming Sequence

SPI supports read and write operations. Before starting any operation, SPI bus busy signal has to be checked by polling SPI_STATUS[0] register bit. Then program beat size, number of bytes to read (only for read operation), select CSN to drive, Assert CSN, set read or write bits in GSPI registers.

Write Operation

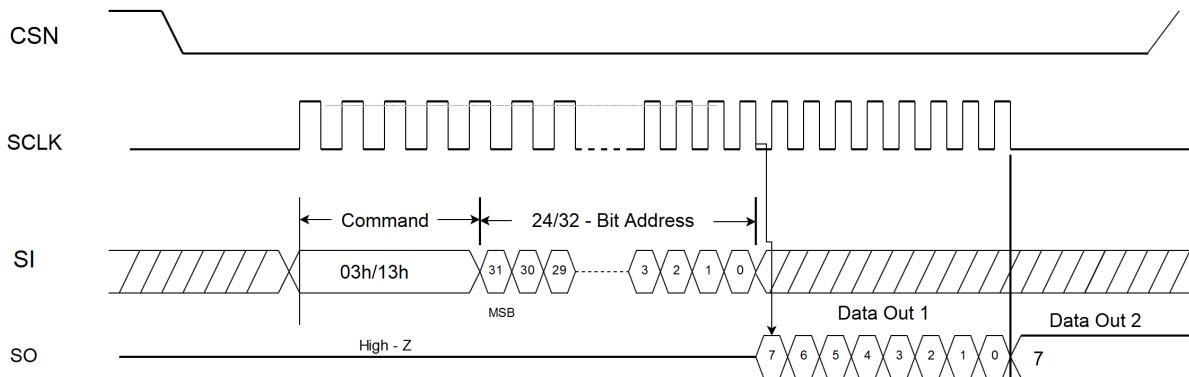
- Assert respective CSN in GSPI_CONFIG1[0]
- Write command, address and write data in WRITE_FIFO register.
- Write valid number of bytes into the GSPI_WRITE_DATA2 register.
- Option to enable USE_PREV_LNTH bit in GSPI_WRITE_DATA2 register to use present programmed length for further write operations also.
- Write valid number of bytes minus one into GSPI_CONFIG1 and set the bit 'TAKE_WR_SIZE_FRM_REG' to consider this byte ordered length to write into flash
- Assert write operation bit in GSPI_CONFIG1[1]
- Write data either in DMA mode or IO mode
- After writing complete data in WRITE_FIFO, SPI bus busy signal has to be checked by polling SPI_STATUS[0] register bit
- Once SPI bus is idle, de-assert the respective CSN.



SPI bus write operation waveform

Read Operation

- Assert respective CSN in GSPI_CONFIG1[0]
- Write command and address in WRITE_FIFO register.
- Assert write operation bit in GSPI_CONFIG1[1]
- check for wfifo empty status
- de-assert write operation bit in GSPI_CONFIG1[1]
- Write the number of bytes to read in GSPI_CONFIG1[12:3] and keep the number of bytes minus one in GSPI_CONFIG1 and trigger read operation by asserting GSPI_CONFIG1[2] bit. Poll for read fifo not empty and read from the READ_FIFO register.
- Read data either in DMA mode or IO mode
- After reading complete data, de-assert the respective CSN
- After completion of read/write operation, GSPI controller raises an interrupt.



SPI bus read operation waveform

SPI bus interface waveform is shown in Fig.2. command, address and write data are initiated by MCU or DMA in write mode. There may be a gap between command, address writing on SPI bus and read data from SPI bus. Because these instructions are issued by MCU firmware directly. Once read is issued, there is no any overhead clock cycles for reading data from flash/device up to end of burst transfer unless read data FIFO is full. it supports up to 32K bytes of data from SPI device in a single read operation. There is no any overhead clock cycles between command, address and write data if data is available in GSPI_WRITE_FIFO before completion of current SPI operation in write mode.

16.7.4 Register Summary

Base Address: 0x4503_0000

| Register Name | Offset | Description |
|---------------------|--------|-----------------------------------|
| GSPI_CLK_CONFIG | 0x00 | GSPI Clock Configuration Register |
| GSPI_BUS_MODE | 0x04 | GSPI Bus Mode Register |
| GSPI_CONFIG1 | 0x10 | GSPI Configuration 1 Register |
| GSPI_CONFIG2 | 0x14 | GSPI Configuration 2 Register |
| GSPI_WRITE_DATA2 | 0x18 | GSPI Write Data 2 Register |
| GSPI_FIFO_THRESHOLD | 0x1C | GSPI FIFO Threshold Register |
| GSPI_STATUS | 0x20 | GSPI Status Register |

| Register Name | Offset | Description |
|--------------------------|-----------|-------------------------------------|
| GSPI_INTR_MASK | 0x24 | GSPI Interrupt Mask Register |
| GSPI_INTR_UNMASK | 0x28 | GSPI Interrupt Unmask Register |
| GSPI_INTR_STS | 0x2C | GSPI Interrupt Status Register |
| GSPI_INTR_ACK | 0x30 | GSPI Interrupt Acknowledge Register |
| GSPI_STS_MC | 0x34 | GSPI State Machine Monitor Register |
| GSPI_CLK_DIVISION_FACTOR | 0x38 | GSPI Clock Division Factor Register |
| GSPI_CONFIG3 | 0x3C | GSPI Configuration 3 Register |
| GSPI_WRITE_FIFO | 0x80-0xBC | GSPI Write Data FIFO Register |
| GSPI_READ_FIFO | 0x80-0xBC | GSPI Read Data FIFO Register |

686 . Register Summary Table

16.7.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, - = Reserved

GSPI_CLK_CONFIG

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------|-------------|--|
| 31:2 | R/W | Reserved | 1 | Reserved |
| 1 | R/W | GSPI_CLK_EN | 0 | GSPI clock enable. 0 – Dynamic clock gating is enabled in side GSPI controller. 1 – Full time clock is enabled for GSPI controller. |
| 0 | R/W | GSPI_CLK_SYNC | 0 | If the clock frequency to FLASH (spi_clk) and SOC clk is same. 1: SCLK clock and SOC clock are same. 0: Divided SOC clock is connected SCLK. Division value is programmable. |

687 . GSPI_CLK_CONFIG Description

GSPI_BUS_MODE

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------------|-------------|--|
| 31:12 | R/W | Reserved | 0 | Reserved |
| 11 | R/W | SPI_HIGH_PERFORMANCE_CE_EN | 0 | High performance features are enabled when this bit is set to one. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|------------------------------|--------------------|---|
| 10:5 | R/W | GSPI_GPIO_MODE_ENA 0 BLES | | These bits are used to map GSPI on GPIO pins. For more details go through GSPI mapping on GPIO section. |
| 4 | R/W | GSPI_CLK_MODE_CSN30 | | 0 – Mode 0 , GSPI_CLK is low when GSPI_CS is high for chip select3 (csn3) 1 – Mode 3 , GSPI_CLK is high when GSPI_CS is high for chip select3 (csn3) |
| 3 | R/W | GSPI_CLK_MODE_CSN20 | | 0 – Mode 0 , GSPI_CLK is low when GSPI_CS is high for chip select2 (csn2) 1 – Mode 3 , GSPI_CLK is high when GSPI_CS is high for chip select2 (csn2) |
| 2 | R/W | GSPI_CLK_MODE_CSN10 | | 0 – Mode 0 , GSPI_CLK is low when GSPI_CS is high for chip select1 (csn1) 1 – Mode 3 , GSPI_CLK is high when GSPI_CS is high for chip select1 (csn1) |
| 1 | R/W | GSPI_CLK_MODE_CSN00 | | 0 – Mode 0 , GSPI_CLK is low when GSPI_CS is high for chip select0 (csn0) 1 – Mode 3 , GSPI_CLK is high when GSPI_CS is high for chip select0 (csn0) |
| 0 | R/W | GSPI_DATA_SAMPLE_E 0 DGE | | Samples MISO data on clock edges. This should be ZERO for mode3 clock. 0 – Posedge of loop back spi_pad_clk. 1 – Negedge of loop back spi_pad_clk. |

688 . GSPI_BUS_MODE Description

GSPI_CONFIG1

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|------------------------------|--------------------|--|
| 31:16 | R/W | Reserved | 0 | Reserved |
| 15 | R/W | SPI_FULL_DUPLEX_EN | 0 | Full duplex mode enable. 0 – Full duplex mode disabled. 1 – Full duplex mode enabled. Full duplex mode means reading while writing. When this bit is enabled, while writing the data to slaves connected to GSPI controller, reads the data from slave selected among the slaves connected to GSPI controller and stores in read_fifo. This fifo will get automatically flush after 16 reads and the fifo is not empty to write into. |
| 14:13 | R/W | GSPI_MANUAL_CSN_SE 0 LECT | | Indicates which CSn is valid. Can be programmable in manual mode. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|---|-------------|--|
| 12:3 | R/W | GSPI_MANUAL_RD_CNT | 0 | Indicates total number of bytes to be read |
| 2 | R/W | GSPI_MANUAL_RD | 0 | Read enable for manual mode when CS is low. |
| 1 | R/W | Read enable for manual mode when CS is low. | 0 | Write enable for manual mode when CS is low. |
| 0 | R/W | GSPI_MANUAL_CSN | 1 | SPI CS in manual mode. |

689 . GSPI_CONFIG1 Description

GSPI_CONFIG2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------------------|-------------|---|
| 31:12 | R/W | Reserved | 0 | Reserved |
| 11 | R/W | MANUAL_GSPI_MODE | 0 | <p>Internally the priority is given to manual mode.</p> <p>0 – SPI mode. When HW_CTRLD_MODE is enabled only, priority is given to manual mode (Breaking of manual mode operations won't be happen even though auto mode request is pending).</p> <p>It gives grant to AUTO mode requests after completion of current manual mode request (csn low to csn high).</p> <p>1 – Host SPI mode. When HW_CTRLD_MODE is enabled, priority is given to auto mode (Breaking of manual mode operation can be happen). Break happens after the beat completion.</p> <p>After breaking, serves AUTO_MODE requests and continues the manual mode operation (already break).</p> <p>Beat refers to write size.</p> |
| 10 | R/W | TAKE_GSPI_MANUAL_W_R_SIZE_FRM_REG | 0 | <p>1 – Take write size from Manual config register1[20:19].</p> <p>0 – No action. Takes write size from fifo [19:16].</p> |
| 9 | R/W | Reserved | 0 | Reserved |
| 8 | R/W | GSPI_MANUAL_SIZE_FM_REG | 1 | <p>Manual reads and manual writes (If take_manual_size_from_reg bit is 1) follow this size.</p> <p>0 – 1 Byte (8 – bit mode)</p> <p>1 – 2 Bytes (16 – bit mode)</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------------------|-------------|--|
| 7 | R/W | GSPI_RD_DATA_SWAP_ MNL_CSN3 | 1 | Swap the read data inside the GSPI controller it-self. 0 – Manual read data swap is disabled for csn3. 1 – Manual read data swap is enabled for csn3. |
| 6 | R/W | GSPI_RD_DATA_SWAP_ MNL_CSN2 | 1 | Swap the read data inside the GSPI controller it-self. 0 – Manual read data swap is disabled for csn2. 1 – Manual read data swap is enabled for csn2. |
| 5 | R/W | GSPI_RD_DATA_SWAP_ MNL_CSN1 | 1 | Swap the read data inside the GSPI controller it-self. 0 – Manual read data swap is disabled for csn1. 1 – Manual read data swap is enabled for csn1. |
| 4 | R/W | GSPI_RD_DATA_SWAP_ MNL_CSN0 | 1 | Swap the read data inside the GSPI controller it-self. 0 – Manual read data swap is disabled for csn0. 1 – Manual read data swap is enabled for csn0. |
| 3 | R/W | GSPI_WR_DATA_SWAP_ MNL_CSN3 | 0 | Swap the write data inside the GSPI controller it-self. 0 – Manual write data swap is disabled for csn3. 1 – Manual write data swap is enabled for csn3. |
| 2 | R/W | GSPI_WR_DATA_SWAP_ MNL_CSN2 | 0 | Swap the write data inside the GSPI controller it-self. 0 – Manual write data swap is disabled for csn2. 1 – Manual write data swap is enabled for csn2. |
| 1 | R/W | GSPI_WR_DATA_SWAP_ MNL_CSN1 | 0 | Swap the write data inside the GSPI controller it-self. 0 – Manual write data swap is disabled for csn1. 1 – Manual write data swap is enabled for csn1. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------------|-------------|--|
| 0 | R/W | GSPI_WR_DATA_SWAP_MNL_CSNO | 0 | Swap the write data inside the GSPI controller it-self. 0 – Manual write data swap is disabled for csn0. 1 – Manual write data swap is enabled for csn0. |

690 .GSPI_CONFIG2 Description

GSPI_WRITE_DATA2

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------------|-------------|---|
| 31:8 | R/W | Reserved | 0 | Reserved |
| 7 | R/W | Use_prev_length | 0 | Use previous length. 1 – Uses previously programmed length in [3:0] of this register for next writes. 0 – No action. Note: TAKE_WR_SIZE_FRM_REG bit should be zero to consider this register. |
| 6:4 | R/W | Reserved | 0 | Reserved |
| 3:0 | R/W | GSPI_MANUAL_WRITE_DATA2 | 0 | Number of bits to be written in write mode. We can select from 1 bit to 16 bits. Update number of bits to be write along with data in write FIFO. The data is written into least 16 bits of a 16x20 FIFO and most 4 bits of FIFO contains information regarding number of bits valid. Note: TAKE_WR_SIZE_FRM_REG bit should be zero to consider these bits. |

691 .GSPI_WRITE_DATA2 Description

GSPI_FIFO_THRLD

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|-----------------------------|
| 31:10 | R/W | Reserved | 0 | Reserved |
| 9 | R/W | RFIFO_RESET | 0 | Read FIFO reset |
| 8 | R/W | WFIFO_RESET | 0 | Write FIFO reset |
| 7:4 | R/W | FIFO_AFULL_THRLD | 12 | FIFO almost full threshold |
| 3:0 | R/W | FIFO_AEMPTY_THRLD | 7 | FIFO almost empty threshold |

692 .GSPI_FIFO_THRLD Description

GSPI_STATUS

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|---|
| 31:11 | R/W | Reserved | 0 | Reserved |
| 10 | R | GSPI_MANUAL_CSN | 1 | Provide the status of chip select signal. 0 – Active. 1 – Inactive. |
| 9 | R | GSPI_manual_rd_cnt | 0 | This is a result of 10 bits ORing counter 1 – Read transactions are in pending (to be done) 0 – No read transactions are in pending |
| 8 | R | fifo_aempty_rfifo_s | 1 | Aempty status indication for Rfifo in manual mode |
| 7 | R | fifo_empty_rfifo_s | 1 | Empty status indication for Rfifo in manual mode |
| 6 | R | Reserved | 0 | Reserved |
| 5 | R | fifo_full_rfifo | 0 | Full status indication for Rfifo in manual mode |
| 4 | R | Reserved | 0 | Reserved |
| 3 | R | fifo_empty_wfifo | 1 | Empty status indication for Wfifo in manual mode |
| 2 | R | fifo_afull_wfifo_s | 0 | Afull status indication for Wfifo in manual mode |
| 1 | R | fifo_full_wfifo_s | 0 | Full status indication for Wfifo in manual mode |
| 0 | R | GSPI_busy | 0 | State of Manual mode. 1 - A read, write or dummy cycle operation is in process in manual mode. 0 – GSPI controller is IDLE in Manual mode. |

693 . GSPI_STATUS Description

GSPI_INTR_MASK

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------------|-------------|--|
| 31:7 | R/W | Reserved | 0 | Reserved |
| 6 | R/W | Fifo_empty_rfifo_mask | 0 | 1 – Read fifo is empty intr mask 0 – Don't touch. |
| 5 | R/W | Fifo_full_wfifo_mask | 0 | 1 – write fifo full intr mask. 0 – Don't touch. |
| 4 | R/W | Fifo_afull_wfifo_mask | 0 | 1 – Write fifo almost full intr mask. 0 – Don't touch. |
| 3 | R/W | Fifo_aempty_wfifo_mask0 | 0 | 1 – write fifo almost empty intr mask. 0 – Don't touch. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 2 | R/W | Fifo_afull_rfifo_mask | 0 | 1 – read fifo almost full intr mask. 0 – Don't touch. |
| 1 | R/W | Fifo_aempty_rfifo_mask | 0 | 1 – Read fifo almost empty intr mask. 0 – Don't touch. |
| 0 | R/W | GSPI_INTR_MASK | 0 | GSPI Interrupt mask bit 1 – mask the GSPI intr. 0 – Don't touch |

694 .GSPI_INTR_MASK Description

GSPI_INTR_UNMASK

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------------|-------------|---|
| 31:7 | R/W | Reserved | 0 | Reserved |
| 6 | R/W | Fifo_empty_rfifo_unmask | 0 | 1 – Read fifo is empty intr unmask 0 – Don't touch. |
| 5 | R/W | Fifo_full_wfifo_unmask | 0 | 1 – write fifo full intr unmask. 0 – Don't touch. |
| 4 | R/W | Fifo_afull_wfifo_unmask | 0 | 1 – Write fifo almost full intr unmask. 0 – Don't touch. |
| 3 | R/W | Fifo_aempty_wfifo_unmask | 0 | 1 – write fifo almost empty intr unmask. 0 – Don't touch. |
| 2 | R/W | Fifo_afull_rfifo_unmask | 0 | 1 – read fifo almost full intr unmask. 0 – Don't touch. |
| 1 | R/W | Fifo_aempty_rfifo_unmask | 0 | 1 – Read fifo almost empty intr unmask. 0 – Don't touch. |
| 0 | R/W | GSPI_INTR_UNMASK | 0 | GSPI Interrupt unmask bit 1 – unmask the GSPI intr. 0 – Don't touch |

695 .GSPI_INTR_UNMASK Description

GSPI_INTR_STS

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------|-------------|--|
| 31:7 | R | Reserved | 0 | Reserved |
| 6 | R | Fifo_empty_rfifo_lvl | 1 | 1 – Read fifo is empty. 0 – Read fifo is not empty. |
| 5 | R | Fifo_full_wfifo_lvl | 0 | 1 – write fifo full 0 – write fifo not full |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------------------|-------------|--|
| 4 | R | Fifo_afull_wfifo_lvl | 0 | 1 – Write fifo almost full threshold 0 – Write fifo not reached almost full threshold |
| 3:2 | R | Reserved | 0 | Reserved |
| 1 | R | Fifo_aempty_rfifo_lvl | 1 | 1 – Read fifo reached almost empty threshold. 0 – Read fifo doesn't reach almost empty threshold. |
| 0 | R | GSPI_INTR_LVL | 0 | GSPI Interrupt Status bit 1 – GSPI raised a interrupt 0 – no interrupt |

696 .GSPI_INTR_STS Description

GSPI_INTR_ACK

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------|-------------|---|
| 31:7 | W | Reserved | 0 | Reserved |
| 6 | W | Fifo_empty_rfifo_ack | 0* | 1 – Read fifo is empty intr ack 0 – Don't touch |
| 5 | W | Fifo_full_wfifo_ack | 0* | 1 – write fifo full intr ack 0 – Don't touch |
| 4 | W | Fifo_afull_wfifo_ack | 0* | 1 – Write fifo almost full intr ack 0 – Don't touch |
| 3:2 | W | Reserved | 0* | Reserved |
| 1 | W | Fifo_aempty_rfifo_ack | 0* | 1 – Read fifo almost empty intr ack. 0 – Don't touch. |
| 0 | W | GSPI_INTR_ACK | 0* | GSPI Interrupt ack bit 1 – GSPI intr ack. 0 – Don't touch |

697 .GSPI_INTR_ACK Description

GSPI_STS_MC

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|---|
| 31:16 | R | Reserved | 0 | Reserved |
| 15:3 | R | spi_rd_cnt | 0 | number of pending bytes to be read by device. |
| 2:0 | R | Bus_ctrl_pstate | 0 | Provides SPI bus controller present state. |

698 .GSPI_STS_MC Description

GSPI_CLK_DIVISION_FACTOR

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------|-------------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | GSPI_CLK_DIV_FACT OR | 0 | Provides GSPI clock division factor to the clock divider, which takes SOC clock as input clock and generates required clock according to division factor. |

699 . GSPI_CLK_DIVISION_FACTOR Description

GSPI_CONFIG3

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------------|-------------|--|
| 15 | R/W | Reserved | 0 | Reserved |
| 14:0 | R/W | SPI_MANUAL_RD_LNT 0 H_TO_BC | 0 | Bits are used to indicate the total number of bytes to read from flash during read operation. This is valid only spi_high_performance_en is enabled. |

700 . GSPI_CONFIG3 Description

GSPI_WRITE_FIFO

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------|-------------|---|
| | W | WRITE_FIFO | 0 | FIFO data is written into this address space. |

701 . GSPI_WRITE_FIFO Description

GSPI_READ_FIFO

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-----------|-------------|--|
| | R | READ_FIFO | 0 | FIFO data is read from this address space. |

702 . GSPI_READ_FIFO Description

16.8 Hardware Random Number Generator

16.8.1 General Description

The Hardware Random Number Generator (HRNG) generates 32-bit random numbers. These random numbers are generated using either a True Random Number Generator or a Pseudo Random Number Generator. These random number generators can be selectively enabled or disabled. Typically, the output of the HRNG is used as a seed for Deterministic Random Bit Generator (DRBG) based random number generators. HRNG is present in MCU HP peripherals.

16.8.2 Features

- Supports 32-bit True Random Number Generator

- Supports 32-bit Pseudo Random Number Generator
- Option to selectively enable these random number generators

16.8.3 Functional Description

True random number generator can be enabled by setting the HRNG_CTRL[0] bit. Pseudo random number generator is enabled by setting the HRNG_CTRL[1] bit. Resetting the HRNG_CTRL[1] or HRNG_CTRL[0] bit, disables the respective random number generators. Only one of them can be active at any point of time. It is recommended to disable the true random number generator when it is not in use to avoid excessive power consumption. The generated random number can be read using 32 bit HRNG_RAND_NUM register. There is a soft reset (HRNG_CTRL[2] bit) to reset the scrambled data to its default value.

16.8.4 Register Summary

Base Address: 0x4108_0000

| Register Name | Offset | Description |
|---------------|--------|-------------------------------|
| HRNG_CTRL | 0x00 | HRNG control register |
| HRNG_RAND_NUM | 0x04 | 32 bit random number register |

703 . Register Summary Table

16.8.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, - = Reserved

HRNG_CTRL Register

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------|-----------|---|
| [31:3] | R | Reserved | 0 | Reserved for future use. |
| 2 | R/W | soft reset | 0 | Soft_reset to reset the scrambled data to its default value (For zeroize purposes) 1 – Reset the scrambled data 0 – Not reset. |
| 1 | R/W | HRNG_PRBS_ST | 0 | This bit is used to start the pseudo random number generation. '1'- Enables pseudo random number generation '0'- Disables pseudo random number generation |
| 0 | R/W | HRNG_TRNG_ST | 0 | This bit is used to start the true number generation. '1'- Enables true random number generation '0'- Disables true random number generation |

704 . HRNG_CTRL Register Description

HRNG_RAND_NUM Register

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------|-------------|---|
| [31:0] | R | HRNG_RAND_NUM | 0x0000_0380 | Generated random number can be read from this register. |

705 . HRNG_RAND_NUM Register Description

16.9 I2C Master and slave

16.9.1 General Description

There are four I²C Master/Slave controllers - two in the MCU HP peripherals (I2C1, I2C2), one in the NWP/security subsystem and one in the MCU ULP subsystem (ULP_I2C). The I2C interface allows the processor to serve as a master or slave on the I2C bus.

16.9.2 Features

Each of these support the following features:

- I²C standard compliant bus interface with open-drain pins
- Configurable as Master or Slave
- Four speed modes: Standard Mode (100 kbps), Fast Mode (400 kbps), Fast Mode Plus (1Mbps) and High-Speed Mode (3.4 Mbps)
- 7 or 10-bit addressing
- 7 or 10-bit combined format transfers
- Support for Clock synchronization and Bus Clear
- Programmable SDA Hold time

The I²C controllers also support additional features listed below to reduce the load on the processor:

- Integrated transmit and receive buffers with support for DMA
- Bulk transmit mode in I²C Slave mode
- Interrupt based operation (polled mode also available)

The I²C in the MCU ULP subsystem (ULP_I2C) supports the following additional power-save features:

- After the DMA is programmed in PS2 state for I²C transfers, the MCU can switch to PS1 state (processor is shutdown) while the I²C controller continues with the data transfer
- In PS1 state (ULP Peripheral mode) the I²C controller completes the data transfer and, triggered by the Peripheral Interrupt, shifts either to the sleep state (without processor intervention) or the active state

The NWP/Security subsystem I2C supports following additional feature:

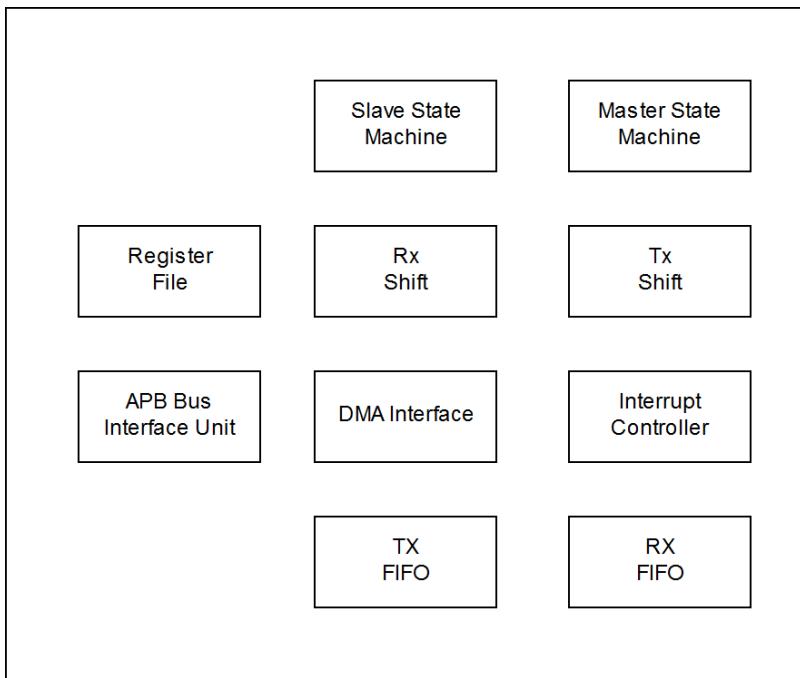
- Ability to connect to external "hardware secure element" accessible through secure API interface

16.9.3 Functional Description

The I2C is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus.

These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a "transmitter" or "receiver," depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave. The I2C module can operate in standard mode (with data rates 0 to 100 Kb/s), fast mode (with

data rates less than or equal to 400 Kb/s), fast mode plus (with data rates less than or equal to 1000 Kb/s), high-speed mode (with data rates less than or equal to 3.4 Mb/s). high-speed mode and fast mode devices are downward compatible. The I2C is made up of an AMBA APB slave interface, an I2C interface, and FIFO logic to maintain coherency between the two interfaces. A simplified block diagram of the I2C is shown in Figure 1.



I2C Block Diagram

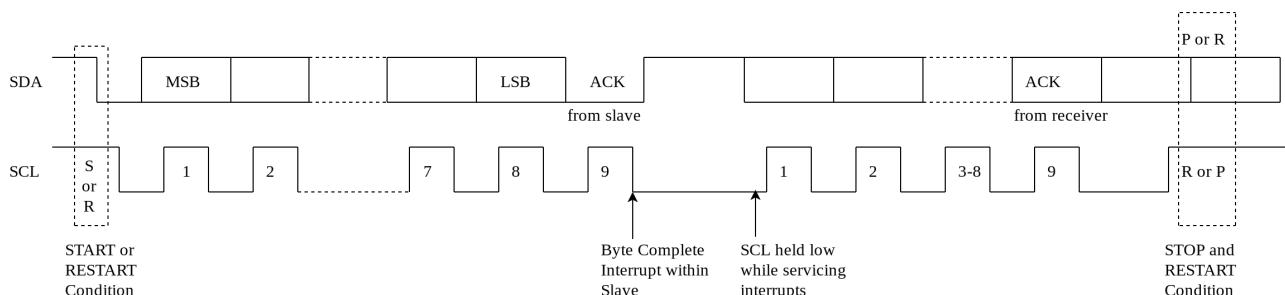
The I2C can be controlled via software to be either I2C master or I2C slave.

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave.

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address. If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data.

This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART

condition. This behavior is shown in Figure 2.



Data transfer on the I2C Bus

16.9.4 Operating Modes

Slave Mode Operation

Initial Configuration

To use the i2c as a slave, perform the following steps:

- Disable the i2c by writing a '0' to bit 0 of the IC_ENABLE register.
- Write to the IC_SAR register (bits 9:0) to set the slave address. This is the address to which the i2c responds.
- Write to the IC_CON register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the i2c in slave-only mode by writing a '0' into bit 6 (IC_SLAVE_DISABLE) and a '0' to bit 0 (MASTER_MODE).
- Enable the i2c by writing a '1' in bit 0 of the IC_ENABLE register.

Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses this I2C and requests data, this I2C acts as a slave-transmitter and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches the slave address in the IC_SAR register of this I2C.
2. The I2C acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
3. The I2C asserts the RD_REQ interrupt (bit 5 of the IC_RAW_INTR_STAT register) and holds the SCL line low. It is in a wait state until software responds.
If the RD_REQ interrupt has been masked, due to IC_INTR_MASK[5] register (M_RD_REQ bit field) being set to 0, then it is recommended that a hardware and/or software timing routine be used to instruct the CPU to perform periodic reads of the IC_RAW_INTR_STAT register.
 - a. Reads that indicate IC_RAW_INTR_STAT[5] (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted.
 - b. Software must then act to satisfy the I2C transfer.
 - c. The timing interval used should be in the order of 10 times the fastest SCL clock period the I2C can handle. For example, for 400 kb/s, the timing interval is 25us.
4. If there is any data remaining in the Tx FIFO before receiving the read request, then the I2C asserts a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register) to flush the old data from the TX FIFO.
If the TX_ABRT interrupt has been masked, due to of IC_INTR_MASK[6] register (M_TX_ABRT bit field) being set to 0, then it is recommended that re-using the timing routine (described in the previous step), or a similar one, be used to read the IC_RAW_INTR_STAT register.
 - a. Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.
 - b. There is no further action required from software.
 - c. The timing interval used should be similar to that described in the previous step for the IC_RAW_INTR_STAT[5] register.
5. Software writes to the IC_DATA_CMD register with the data to be written (by writing a '0' in bit 8).
6. Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the IC_RAW_INTR_STAT register before proceeding. If the RD_REQ and/ or TX_ABRT interrupts have been masked, then clearing of the IC_RAW_INTR_STAT register will have already been performed when either the R_RD_REQ or R_TX_ABRT bit has been read as 1.

7. The I2C releases the SCL and transmits the byte.
8. The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses this I2C and is sending data, this I2C acts as a slave-receiver and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches this I2C's slave address in the IC_SAR register.
2. This I2C acknowledges the sent address and recognizes the direction of the transfer to indicate that the I2C is acting as a slave-receiver.
3. I2C receives the transmitted byte and places it in the receive buffer
4. I2C asserts the RX_FULL interrupt (IC_RAW_INTR_STAT[2] register).
If the RX_FULL interrupt has been masked, due to setting IC_INTR_MASK[2] register to 0 or setting IC_TX_TL to a value larger than 0, then it is recommended that a timing routine be implemented for periodic reads of the IC_STATUS register. Reads of the IC_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.
5. Software may read the byte from the IC_DATA_CMD register (bits 7:0).
6. The other master device may hold the I2C bus by issuing a RESTART condition, or release the bus by issuing a STOP condition.

Master Mode Operation

Initial Configuration

1. Disable the I2C by writing 0 to bit 0 of the IC_ENABLE register.
2. Write to the IC_CON register to set the maximum speed mode supported for slave operation (bits 2:1) and to specify whether the I2C starts its transfers in 7/10 bit addressing mode when the device is a slave (bit 3).
3. Write to the IC_TAR register the address of the I2C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I2C. The desired speed of the I2C master-initiated transfers, either 7-bit or 10-bit addressing, is controlled by the IC_10BITADDR_MASTER bit field (bit 12).
4. Only applicable for high-speed mode transfers. Write to the IC_HS_MADDR register the desired master code for the I2C. The master code is programmer-defined.
5. Enable the I2C by writing a 1 to bit 0 of the IC_ENABLE register.
6. Now write the transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_DATA_CMD register is written before the I2C is enabled, the data and commands are lost as the buffers are kept cleared when I2C is not enabled.

START and STOP Generation

When operating as an I2C master, putting data into the transmit FIFO causes the I2C to generate a START condition on the I2C bus. Writing a 1 to IC_DATA_CMD[9] causes the I2C to generate a STOP condition on the I2C bus; a STOP condition is not issued if this bit is not set, even if the transmit FIFO is empty.

Master Transmit and Master Receive

The I2C supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx FIFO and Command Register (IC_DATA_CMD). The *CMD* bit [8] should be written to 0 for I2C write operations. Subsequently, a read

command may be issued by writing “don’t cares” to the lower byte of the IC_DATA_CMD register, and a 1 should be written to the *CMD* bit. The I2C master continues to initiate transfers as long as there are commands present in the transmit FIFO. If the transmit FIFO becomes empty, the master either inserts a STOP condition after completing the current transfers, or it checks to see if IC_DATA_CMD[9] is set to 1.

1. If set to 1, it issues a STOP condition after completing the current transfer.
2. If set to 0, it holds SCL low until next command is written to the transmit FIFO

16.9.5 Register Summary

I2C1 Base Address: 0x4401_0000

I2C2 Base Address: 0x4704_0000

ULP_I2C Base Address: 0x2404_0000

| Register Name | Offset | Description |
|------------------|--------|---|
| IC_CON | 0x00 | Control Register |
| IC_TAR | 0x04 | Target Address Register |
| IC_SAR | 0x08 | Slave Address Register |
| IC_HS_MADDR | 0x0C | High Speed Master Mode Code Address Register |
| IC_DATA_CMD | 0x10 | Rx/Tx Data Buffer and Command Register |
| IC_SS_SCL_HCNT | 0x14 | Standard Speed I2C Clock SCL High Count Register |
| IC_SS_SCL_LCNT | 0x18 | Standard Speed I2C Clock SCL Low Count Register |
| IC_FS_SCL_HCNT | 0x1C | Fast Mode or Fast Mode Plus I2C Clock SCL High Count Register |
| IC_FS_SCL_LCNT | 0x20 | Fast Mode or Fast Mode Plus I2C Clock SCL Low Count Register |
| IC_HS_SCL_HCNT | 0x24 | High Speed I2C Clock SCL High Count Register |
| IC_HS_SCL_LCNT | 0x28 | High Speed I2C Clock SCL Low Count Register |
| IC_INTR_STAT | 0x2C | Interrupt Status Register |
| IC_INTR_MASK | 0x30 | Interrupt Mask Register |
| IC_RAW_INTR_STAT | 0x34 | Raw Interrupt Status Register |
| IC_RX_TL | 0x38 | Receive FIFO Threshold Register |
| IC_TX_TL | 0x3C | Transmit FIFO Threshold Register |
| IC_CLR_INTR | 0x40 | Clear Combined and Individual Interrupt Register |
| IC_CLR_RX_UNDER | 0x44 | Clear RX_UNDER Interrupt Register |
| IC_CLR_RX_OVER | 0x48 | Clear RX_OVER Interrupt Register |
| IC_CLR_TX_OVER | 0x4C | Clear TX_OVER Interrupt Register |
| IC_CLR_RD_REQ | 0x50 | Clear RD_REQ Interrupt Register |
| IC_CLR_TX_ABRT | 0x54 | Clear TX_ABRT Interrupt Register |
| IC_CLR_RX_DONE | 0x58 | Clear RX_DONE Interrupt Register |
| IC_CLR_ACTIVITY | 0x5c | Clear ACTIVITY Interrupt Register |

| Register Name | Offset | Description |
|-------------------------------|---------------|--|
| IC_CLR_STOP_DET | 0x60 | Clear STOP_DET Interrupt Register |
| IC_CLR_START_DET | 0x64 | Clear START_DET Interrupt Register |
| IC_CLR_GEN_CALL | 0x68 | Clear GEN_CALL Interrupt Register |
| IC_ENABLE | 0x6C | Enable Register |
| IC_STATUS | 0x70 | Status Register |
| IC_TXFLR | 0x74 | Transmit FIFO Level Register |
| IC_RXFLR | 0x78 | Receive FIFO Level Register |
| IC_SDA_HOLD | 0x7C | SDA Hold Time Length Register |
| IC_TX_ABRT_SOURCE | 0x80 | Transmit Abort Source Register |
| IC_SLV_DATA_NACK_ONLY | 0x84 | Generate Slave Data NACK Register |
| IC_DMA_CR | 0x88 | DMA Control Register |
| IC_DMA_TDLR | 0x8C | DMA Transmit Data Level Register |
| IC_DMA_RDLR | 0x90 | Receive Data Level Register |
| IC_SDA_SETUP | 0x94 | SDA Setup Register |
| IC_ACK_GENERAL_CALL | 0x98 | ACK General Call Register |
| IC_ENABLE_STATUS | 0x9C | Enable Status Register |
| IC_FS_SPKLEN | 0xA0 | SS and FS Spike Suppression Limit Register |
| IC_HS_SPKLEN | 0xA4 | HS Spike Suppression Limit Register |
| IC_CLR_RESTART_DET | 0xA8 | Clear RESTART_DET Interrupt Register |
| IC_COMP_PARAM_1 | 0xF4 | Component Parameter Register 1 |
| IC_COMP_VERSION | 0xF8 | Component Version Register |
| IC_COMP_TYPE | 0xFC | Component Type Register |
| IC_SCL_STUCK_AT_LOW_TIMEOUT | 0xAC | SCL Stuck at Low Timeout |
| IC_SDA_STUCK_AT_LOW_TIMEOUT | 0xB0 | SDA Stuck at Low Timeout |
| IC_CLR_SCL_STUCK_DET | 0xB4 | Clear SCL Stuck at Low Detect Interrupt Register |
| IC_DEVICE_ID | 0xB8 | Device ID |
| IC_SMBUS_CLOCK_LOW_SEXT | 0xBC | SMBUS Slave Clock Extend Timeout Register |
| IC_SMBUS_CLOCK_LOW_MEXT | 0xC0 | SMBUS Master extend clock Timeout Register |
| IC_SMBUS_THIGH_MAX_IDLE_COUNT | 0xC4 | SMBus Thigh MAX Bus-Idle count Register |
| IC_SMBUS_INTR_STAT | 0xC8 | SMBUS Interrupt Status Register |
| IC_SMBUS_INTR_MASK | 0xCC | Interrupt Mask Register |
| IC_SMBUS_INTR_RAW_STATUS | 0xD0 | SMBUS Raw Interrupt Status Register |

| Register Name | Offset | Description |
|-------------------|--------|---------------------------------|
| IC_CLR_SMBUS_INTR | 0xD4 | Clear SMBUS Interrupt Register |
| IC_OPTIONAL_SAR | 0xD8 | Optional Slave Address Register |
| IC_SMBUS_UDID_LSB | 0xDC | SMBUS ARP UDID LSB Register |

706 . Register Summary Table

16.9.6 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, N/A = Reserved

IC_CON

This register can be written only when the i2c is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 31:12 | N/A | Reserved | | Reserved |
| 11 | R/W | BUS_CLEAR_FEATURE_CTRL | 0 | <p>In Master Mode:</p> <ul style="list-style-type: none"> • 1'b1: Bus Clear Feature is enabled • 1'b0: Bus Clear Feature is disabled <p>In Slave Mode, this register bit is not applicable.</p> |
| 10 | R/W | STOP_DET_IF_MASTER_ACTIVE | 0 | <p>In Master mode</p> <ul style="list-style-type: none"> • 1'b1: Issues the STOP_DET interrupt only when the master is active • 1'b0: Issues the STOP_DET irrespective of whether the master is active |
| 9 | N/A | Reserved | 0 | Reserved |
| 8 | R/W | TX_EMPTY_CTRL | 0 | This bit controls the generation of the TX_EMPTY interrupt, as described in the IC_RAW_INTR_STAT register. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|--|
| 7 | R/W | STOP_DET_IFADDRESSED | 0 | <p>In slave mode:</p> <p>1'b1 – issues the STOP_DET interrupt only when it is addressed.</p> <p>1'b0 – issues the STOP_DET irrespective of whether it's addressed or not.</p> <p>Dependencies: This register bit value is applicable in the slave mode only (MASTER_MODE = 1'b0)</p> <p>NOTE: During a general call address, this slave does not issue the STOP_DET interrupt if STOP_DET_IF_ADDRESSED = 1'b1, even if the slave responds to the general call address by generating ACK.</p> <p>The STOP_DET interrupt is generated only when the transmitted address matches the slave address (SAR).</p> |
| 6 | R/W | IC_SLAVE_DISABLE | 0x1 | <p>If this bit is set (slave is disabled), I2C functions only as a master and does not perform any action that requires a slave.</p> <p>0: slave is enabled 1: slave is disabled</p> <p>NOTE: Software should ensure that if this bit is written with '0,' then bit 0 should also be written with a '0'.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------|-----------|---|
| 5 | R/W | IC_RESTART_EN | 0x1 | <p>Determines whether RESTART conditions may be sent when acting as a master.</p> <p>Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several I2C operations.</p> <p>0: disable 1: enable</p> <p>When the RESTART is disabled, the I2C master is incapable of performing the following functions:</p> <ul style="list-style-type: none"> • Sending a START BYTE • Performing any high-speed mode operation • Performing direction changes in combined format mode • Performing a read operation with a 10-bit address <p>By replacing RESTART condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple I2C transfers.</p> <p>If the above operations are performed, it will result in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register.</p> |
| 4 | R | IC_10BITADDR_MASTER_rd_only | 0x1 | <p>the function of this bit is handled by bit 12 of IC_TAR register, and becomes a read-only copy called IC_10BITADDR_MASTER_rd_only.</p> <p>0: 7-bit addressing 1: 10-bit addressing</p> |
| 3 | R/W | IC_10BITADDR_SLAVE | 0x1 | <p>When acting as a slave, this bit controls whether the I2C responds to 7- or 10-bit addresses.</p> <p>0: 7-bit addressing. The I2C ignores transactions that involve 10-bit addressing; for 7-bit addressing, only the lower 7 bits of the IC_SAR register are compared.</p> <p>1: 10-bit addressing. The I2C responds to only 10-bit addressing transfers that match the full 10 bits of the IC_SAR register.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------|-----------|--|
| 2:1 | R/W | SPEED | 0x3 | <p>These bits control at which speed the I2C operates.</p> <p>Hardware protects against illegal values being programmed by software. register These bits must be programmed appropriately for slave mode also, as it is used to capture correct value of spike filter as per the speed mode.</p> <p>This register should be programmed only with a value in the range of 1 to IC_MAX_SPEED_MODE; otherwise, hardware updates this register with the value of IC_MAX_SPEED_MODE.</p> <ul style="list-style-type: none"> • 1: standard mode (0 to 100 Kb/s) • 2: fast mode (\leq 400 Kb/s) or fast mode plus (\leq 1000 Kb/s) • 3: high speed mode (\leq 3.4 Mb/s) |
| 0 | R/W | MASTER_MODE | 0x1 | <p>This bit controls whether the I2C master is enabled.</p> <p>0: master disabled</p> <p>1: master enabled</p> <p>NOTE: Software should ensure that if this bit is written with '1,' then bit 6 should also be written with a '1'.</p> |

707 . IC_CON Description

IC_TAR

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------|-----------|---|
| 31:14 | N/A | Reserved | | Reserved |
| 13 | R/W | Device_ID | 0 | <p>If bit 11 (SPECIAL) is set to 1, then this bit indicates whether a Device-ID of a particular slave mentioned in IC_TAR[6:0] is to be performed by the I2C Master.</p> <ul style="list-style-type: none"> • 0: Device-ID is not performed and checks ic_tar[10] to perform either general call or START byte command. • 1: Device-ID transfer is performed and bytes based on the number of read commands in the Tx-FIFO are received from the targeted slave and put in the Rx-FIFO. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|---|
| 12 | R/W | IC_10BITADDR_MAST_ER | 1 | This bit controls whether the I2C starts its transfers in 7-or 10-bit addressing mode when acting as a master. <ul style="list-style-type: none">• 0: 7-bit addressing• 1: 10-bit addressing |
| 11 | R/W | SPECIAL | 0 | This bit indicates whether software performs a Device-ID, General Call or START BYTE command. <ul style="list-style-type: none">• 0: ignore bit 10 GC_OR_START and use IC_TAR normally• 1: perform special I2C command as specified in Device-ID or GC_OR_START bit |
| 10 | R/W | GC_OR_START | 0 | If bit 11 (SPECIAL) is set to 1 and bit 13 (Device-ID) is set to 0, then this bit indicates whether a General Call or START byte command is to be performed by the I2C. <ul style="list-style-type: none">• 0: General Call Address – after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register. The I2C remains in General Call mode until the SPECIAL bit value (bit 11) is cleared.• 1: START BYTE |
| 9:0 | R/W | IC_TAR | 0x0A0 | This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. If the IC_TAR and IC_SAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave. |

708 .IC_TAR Description

IC_SAR

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:10 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 9:0 | R/W | IC_SAR | 0x055 | <p>The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR[6:0] is used.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>NOTE: The default values cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7f.</p> <p>The correct operation of the device is not guaranteed if the IC_SAR or IC_TAR is programmed to a reserved value.</p> |

709 .IC_SAR Description

IC_HS_MADDR

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|--|
| 31:3 | N/A | Reserved | | Reserved |
| 2:0 | R/W | IC_HS_MAR | 0x1 | <p>This bit field holds the value of the I2C HS mode master code. HS-mode master codes are reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code;</p> <p>up to eight high speed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7. This register goes away and becomes read-only returning 0's if the IC_MAX_SPEED_MODE configuration parameter is set to either Standard(1) or Fast (2).</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> |

710 .IC_HS_MADDR Description

IC_DATA_CMD

This is the register the CPU writes to when filling the TX FIFO and the CPU reads from when retrieving bytes from RX FIFO.

In order for the i2c to continue acknowledging reads, a read command should be written for every byte that is to be received; otherwise i2c will stop acknowledging.

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------|-----------|--|
| 31:12 | N/A | Reserved | | Reserved |
| 11 | R/W | FIRST_DATA_BYTE | 0 | Indicates the first data byte received after the address phase for receive transfer in Master receiver or Slave receiver mode. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 10 | W | RESTART | 0 | <p>This bit controls whether a RESTART is issued before the byte is sent or received.</p> <ul style="list-style-type: none"> • 1 – If IC_RESTART_EN is 1, a RESTART is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command; if IC_RESTART_EN is 0, a STOP followed by a START is issued instead. • 0 – If IC_RESTART_EN is 1, a RESTART is issued only if the transfer direction is changing from the previous command; if IC_RESTART_EN is 0, a STOP followed by a START is issued instead. |
| 9 | W | STOP | 0 | <p>This bit controls whether a STOP is issued after the byte is sent or received.</p> <ul style="list-style-type: none"> • 1 – STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. • 0 – STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 8 | W | CMD | 0 | <p>This bit controls whether a read or a write is performed. This bit does not control the direction when the I2C acts as a slave. It controls only the direction when it acts as a master.</p> <ul style="list-style-type: none"> • 1 – Read • 0 – Write <p>When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a “don’t care” because writes to this register are not required.</p> <p>In slave-transmitter mode, a “0” indicates that the data in IC_DATA_CMD is to be transmitted.</p> <p>When programming this bit, the following should be remembered: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT interrupt</p> <p>(bit 6 of the IC_RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the IC_TAR register has been cleared.</p> <p>If a “1” is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs.</p> |
| 7:0 | R/W | DAT | 0 | <p>This register contains the data to be transmitted or received on the I2C bus. If this register is being written and a read is to be performed, bits 7:0 (DAT) are ignored by the I2C.</p> <p>However, when this register is read, these bits return the value of data received on the I2C interface.</p> |

711 .IC_DATA_CMD Description

IC_SS_SCL_HCNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:16 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:0 | R/W | IC_SS_SCL_HCNT | 0x01F4 | <p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed.</p> <p>This register can be written only when the I2C interface is disabled which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.</p> <p>NOTE: This register must not be programmed to a value higher than 65525, because I2C uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10.</p> |

712 .IC_SS_SCL_HCNT Description

IC_SS_SCL_LCNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------|-----------|---|
| 31:16 | N/A | Reserved | | Reserved |
| 15:0 | R/W | IC_SS_SCL_LCNT | 0x024C | <p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed.</p> <p>This register can be written only when the I2C interface is disabled which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted, results in 8 being set.</p> |

713 .IC_SS_SCL_LCNT Description

IC_FS_SCL_HCNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:16 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:0 | R/W | IC_FS_SCL_HCNT | 0x004B | <p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast mode or fast mode plus.</p> <p>It is used in high-speed mode to send the Master Code and START BYTE or General CALL.</p> <p>This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.</p> |

714 .IC_FS_SCL_HCNT Description

IC_FS_SCL_LCNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------|-----------|--|
| 31:16 | N/A | Reserved | | Reserved |
| 15:0 | R/W | IC_FS_SCL_LCNT | 0x00A3 | <p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for fast mode or fast mode plus.</p> <p>It is used in high-speed mode to send the Master Code and START BYTE or General CALL.</p> <p>This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE = standard.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set.</p> |

715 .IC_FS_SCL_LCNT Description

IC_HS_SCL_HCNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:16 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:0 | R/W | IC_HS_SCL_HCNT | 0x000F | <p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high period count for high speed.</p> <p>The SCL High time depends on the loading of the bus. For 100pF loading, the SCL High time is 60ns; for 400pF loading, the SCL High time is 120ns.</p> <p>This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE != high.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.</p> |

716 . IC_HS_SCL_HCNT Description

IC_HS_SCL_LCNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------|-----------|--|
| 31:16 | N/A | Reserved | | Reserved |
| 15:0 | R/W | IC_HS_SCL_LCNT | 0x0028 | <p>This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for high speed.</p> <p>The SCL low time depends on the loading of the bus. For 100pF loading, the SCL low time is 160ns; for 400pF loading, the SCL low time is 320ns.</p> <p>This register goes away and becomes read-only returning 0s if IC_MAX_SPEED_MODE != high.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set.</p> |

717 . IC_HS_SCL_LCNT Description

IC_INTR_STAT

| Bit | Access | Function | POR Value | Description |
|-------|----------|--------------------------|-----------|--|
| 31:15 | N/A | Reserved | | Reserved |
| 14 | R or R/W | M_SCL_STUCK_AT_L 0 OW | 0 | See IC_RAW_INTR_STAT for a detailed description of this bit. |
| 13 | R or R/W | R_MST_ON_HOLD | 0 | See “IC_RAW_INTR_STAT” for a detailed description of this bit. |
| 12 | R | R_RESTART_DET | 0 | See “IC_RAW_INTR_STAT” for a detailed description of these bits. |
| 11 | | R_GEN_CALL | | |
| 10 | | R_START_DET | | |
| 9 | | R_STOP_DET | | |
| 8 | | R_ACTIVITY | | |
| 7 | | R_RX_DONE | | |
| 6 | | R_TX_ABRT | | |
| 5 | | R_RD_REQ | | |
| 4 | | R_TX_EMPTY | | |
| 3 | | R_TX_OVER | | |
| 2 | | R_RX_FULL | | |
| 1 | | R_RX_OVER | | |
| 0 | | R_RX_UNDER | | |

718 .IC_INTR_STAT Description

IC_INTR_MASK

| Bit | Access | Function | POR Value | Description |
|-------|----------|--------------------------|-----------|---|
| 31:15 | N/A | Reserved | | Reserved |
| 14 | R or R/W | R_SCL_STUCK_AT_L 1 OW | 1 | This bit masks the R_SCL_STUCK_AT_LOW interrupt bit in the IC_INTR_STAT register |
| 13 | R/W | M_MST_ON_HOLD | 0 | This bit masks the R_MST_ON_HOLD interrupt bit in the IC_INTR_STAT register. |
| 12 | R/W | M_RESTART_DET | | This bit masks the R_RESTART_DET interrupt status bit in the IC_INTR_STAT register. |

| Bit | Access | Function | POR Value | Description |
|-----|----------|-------------|-----------|-------------|
| 11 | R or R/W | M_GEN_CALL | | |
| 10 | | M_START_DET | | |
| 9 | | M_STOP_DET | | |
| 8 | | M_ACTIVITY | | |
| 7 | | M_RX_DONE | | |
| 6 | | M_TX_ABRT | | |
| 5 | | M_RD_REQ | | |
| 4 | | M_TX_EMPTY | | |
| 3 | | M_TX_OVER | | |
| 2 | | M_RX_FULL | | |
| 1 | | M_RX_OVER | | |
| 0 | | M_RX_UNDER | | |

719 .IC_INTR_MASK Description

IC_RAW_INTR_STAT

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------|-----------|--|
| 31:15 | N/A | Reserved | | Reserved |
| 14 | R | SCL_STUCK_AT_LO_W | 0 | Indicates whether the SCL Line is stuck at low for the IC_SCL_STUCK_LOW_TIMOUT number of ic_clk periods. |
| 13 | R | MST_ON_HOLD | 0 | Indicates whether a master is holding the bus and the Tx FIFO is empty. |
| 12 | R | RESTART_DET | 0 | Indicates whether a RESTART condition has occurred on the I2C interface when I2C is operating in slave mode and the slave is the addressed slave. NOTE: However, in high-speed mode or during a START BYTE transfer, the RESTART comes before the address field as per the I2C protocol. In this case, the slave is not the addressed slave when the RESTART is issued, therefore I2C does not generate the RESTART_DET interrupt. |
| 11 | R | GEN_CALL | 0 | Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C or when the CPU reads bit 0 of the IC_CLR_GEN_CALL register. I2C stores the received data in the Rx buffer. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|--|
| 10 | R | START_DET | 0 | Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode. |
| 9 | R | STOP_DET | 0 | <p>Indicates whether a STOP condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.</p> <p>In Slave Mode:</p> <ul style="list-style-type: none"> If IC_CON[7]=1'b1 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is generated only if the slave is addressed. <p>Note: During a general call address, this slave does not issue a STOP_DET interrupt if STOP_DET_IF_ADDRESSED=1'b1, even if the slave responds to the general call address by generating ACK.</p> <p>The STOP_DET interrupt is generated only when the transmitted address matches the slave address (SAR).</p> <ul style="list-style-type: none"> If IC_CON[7]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is issued irrespective of whether it is being addressed. <p>In Master Mode:</p> <ul style="list-style-type: none"> If IC_CON[10]=1'b1 (STOP_DET_IF_MASTER_ACTIVE), the STOP_DET interrupt is issued only if the master is active. If IC_CON[10]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is issued irrespective of whether the master is active. |
| 8 | R | ACTIVITY | 0 | <p>This bit captures I2C activity and stays set until it is cleared. There are four ways to clear it:</p> <ul style="list-style-type: none"> Disabling the I2C Reading the IC_CLR_ACTIVITY register Reading the IC_CLR_INTR register System reset <p>Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I2C module is idle, this bit remains set until cleared, indicating that there was activity on the bus.</p> |
| 7 | R | RX_DONE | 0 | <p>When the I2C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte.</p> <p>This occurs on the last byte of the transmission, indicating that the transmission is done.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 6 | R | TX_ABRT | 0 | <p>This bit indicates if I2C, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO.</p> <p>This situation can occur both as an I2C master or an I2C slave and is referred to as a “transmit abort”.</p> <p>When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places.</p> <p>NOTE: The I2C flushes/resets/empties only the TX_FIFO whenever there is a transmit abort caused by any of the events tracked by the IC_TX_ABRT_SOURCE register.</p> <p>The Tx FIFO remains in this flushed state until the register IC_CLR_TX_ABRT is read. Once this read is performed, the Tx FIFO is then ready to accept more data bytes from the APB interface.</p> |
| 5 | R | RD_REQ | 0 | <p>This bit is set to 1 when I2C is acting as a slave and another I2C master is attempting to read data from I2C. The I2C holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced,</p> <p>which means that the slave has been addressed by a remote master that is asking for data to be transferred.</p> <p>The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register.</p> <p>This bit is set to 0 just after the processor reads the IC_CLR_RD_REQ register.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 4 | R | TX_EMPTY | 0 | <p>The behavior of the TX_EMPTY interrupt status differs based on the TX_EMPTY_CTRL selection in the IC_CON register.</p> <ul style="list-style-type: none"> When TX_EMPTY_CTRL = 0: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register. When TX_EMPTY_CTRL = 1: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register and the transmission of the address/data from the internal shift register for the most recently popped command is completed. It is automatically cleared by hardware when the buffer level goes above the threshold. <p>When IC_ENABLE[0] is set to 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer any activity, then with ic_en=0, this bit is set to 0.</p> |
| 3 | R | TX_OVER | 0 | <p>Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the IC_DATA_CMD register.</p> <p>When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> |
| 2 | R | RX_FULL | 0 | <p>Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold.</p> <p>If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full.</p> <p>So this bit is cleared once IC_ENABLE[0] is set to 0, regardless of the activity that continues.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 1 | R | RX_OVER | 0 | <p>Set if the receive buffer is completely filled to IC_RX_BUFFER_DEPTH and an additional byte is received from an external I2C device.</p> <p>The I2C acknowledges this, but any data bytes received after the FIFO is full are lost.</p> <p>If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.</p> |
| 0 | R | RX_UNDER | 0 | <p>Set if the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register.</p> <p>If the module is disabled (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared</p> |

720 .IC_RAW_INTR_STAT Description

IC_RX_TL

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:8 | N/A | Reserved | | Reserved |
| 7:0 | R/W | RX_TL | 0x06 | <p>Receive FIFO Threshold Level</p> <p>Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in IC_RAW_INTR_STAT register).</p> <p>The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer.</p> <p>If an attempt is made to do that, the actual value set will be the maximum depth of the buffer.</p> <p>A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.</p> |

721 .IC_RX_TL Description

IC_TX_TL

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 7:0 | R/W | TX_TL | 0x02 | <p>Receive FIFO Threshold Level</p> <p>Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in IC_RAW_INTR_STAT register).</p> <p>The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer.</p> <p>If an attempt is made to do that, the actual value set will be the maximum depth of the buffer.</p> <p>A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.</p> |

722 . IC_TX_TL Description

IC_CLR_INTR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_INTR | 0 | <p>Read this register to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register.</p> <p>This bit does not clear hardware clear-able interrupts but software clear-able interrupts.</p> <p>Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE.</p> |

723 . IC_CLR_INTR Description

IC_CLR_RX_UNDER

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_RX_UNDER | 0 | Read this register to clear the RX_UNDER interrupt (bit 0) of the IC_RAW_INTR_STAT register. |

724 . IC_CLR_RX_UNDER Description

IC_CLR_RX_OVER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------|-----------|---|
| 0 | R | CLR_RX_OVER | 0 | Read this register to clear the RX_OVER interrupt (bit 1) of the IC_RAW_INTR_STAT register. |

725 .IC_CLR_RX_OVER Description

IC_CLR_TX_OVER

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------|-----------|---|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_TX_OVER | 0 | Read this register to clear the TX_OVER interrupt (bit 3) of the IC_RAW_INTR_STAT register. |

726 .IC_CLR_TX_OVER Description

IC_CLR_RD_REQ

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_RD_REQ | 0 | Read this register to clear the RD_REQ interrupt (bit 5) of the IC_RAW_INTR_STAT register. |

727 .IC_CLR_RD_REQ Description

IC_CLR_TX_ABRT

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_TX_ABRT | 0 | Read this register to clear the TX_ABRT interrupt (bit 6) of the IC_RAW_INTR_STAT register, and the IC_TX_ABRT_SOURCE register. This also releases the Tx FIFO from the flushed/reset state, allowing more writes to the Tx FIFO. Refer to Bit 9 of the IC_TX_ABRT_SOURCE register for an exception to clearing IC_TX_ABRT_SOURCE. |

728 .IC_CLR_TX_ABRT Description

IC_CLR_RX_DONE

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------|-----------|---|
| 0 | R | CLR_RX_DONE | 0 | Read this register to clear the RX_DONE interrupt (bit 7) of the IC_RAW_INTR_STAT register. |

729 .IC_CLR_RX_DONE Description

IC_CLR_ACTIVITY

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_ACTIVITY | 0 | <p>Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set.</p> <p>It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus.</p> <p>The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the IC_RAW_INTR_STAT register.</p> |

730 .IC_CLR_ACTIVITY Description

IC_CLR_STOP_DET

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_STOP_DET | 0 | Read this register to clear the STOP_DET interrupt (bit 9) of the IC_RAW_INTR_STAT register. |

731 .IC_CLR_STOP_DET Description

IC_CLR_START_DET

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_START_DET | 0 | Read this register to clear the START_DET interrupt (bit 10) of the IC_RAW_INTR_STAT register. |

732 .IC_CLR_START_DET Description

IC_CLR_GEN_CALL

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 0 | R | CLR_GEN_CALL | 0 | Read this register to clear the GEN_CALL interrupt (bit 11) of the IC_RAW_INTR_STAT register. |

733 .IC_CLR_START_DET Description

IC_ENABLE

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-------------------------------|------------------|--|
| 31:16 | N/A | Reserved | | Reserved |
| 15:4 | N/A | Reserved | | Reserved |
| 3 | R/W | SDA_STUCK_RECOVER Y_ENABLE | 0 | If SDA is stuck at low indicated through the TX_ABORT interrupt (IC_TX_ABRT_SOURCE[17]), then this bit is used as a control knob to initiate the SDA Recovery Mechanism (that is, send at most 9 SCL clocks and STOP to release the SDA line) and then this bit gets auto clear. |
| 2 | R/W | TX_CMD_BLOCK | 0 | In Master mode <ul style="list-style-type: none"> • 1'b1: Blocks the transmission of data on I2C bus even if Tx FIFO has data to transmit. • 1'b0: The transmission of data starts on I2C bus automatically, as soon as the first data is available in the Tx FIFO. Note: To block the execution of Master commands, set the TX_CMD_BLOCK bit only when Tx FIFO is empty (IC_STATUS[2]=1) and the master is in the Idle state (IC_STATUS[5] == 0). Any further commands put in the Tx FIFO are not executed until TX_CMD_BLOCK bit is unset. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 1 | R/W | ABORT | 0 | <p>When set, the controller initiates the transfer abort.</p> <ul style="list-style-type: none"> • 0: ABORT not initiated or ABORT done • 1: ABORT operation in progress <p>The software can abort the I2C transfer in master mode by setting this bit.</p> <p>The software can set this bit only when ENABLE is already set; otherwise, the controller ignores any write to ABORT bit. The software cannot clear the ABORT bit once set.</p> <p>In response to an ABORT, the controller issues a STOP and flushes the Tx FIFO after completing the current transfer, then sets the TX_ABORT interrupt after the abort operation.</p> <p>The ABORT bit is cleared automatically after the abort operation.</p> |
| 0 | R/W | ENABLE | 0 | <p>Controls whether the I2C is enabled.</p> <ul style="list-style-type: none"> • 0: Disables I2C (TX and RX FIFOs are held in an erased state) • 1: Enables I2C <p>Software can disable I2C while it is active. However, it is important that care be taken to ensure that I2C is disabled properly.</p> <p>When I2C is disabled, the following occurs:</p> <ul style="list-style-type: none"> • The TX FIFO and RX FIFO get flushed. • Status bits in the IC_INTR_STAT register are still active until I2C goes into IDLE state. <p>If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete.</p> <p>If the module is receiving, the I2C stops the current transfer at the end of the current byte and does not acknowledge the transfer.</p> <p>In systems with asynchronous pclk and ic_clk when IC_CLK_TYPE parameter set to asynchronous (1), there is a two ic_clk delay when enabling or disabling the I2C.</p> |

734 .IC_ENABLE Description

IC_STATUS

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------------------|-----------|--|
| 31:12 | N/A | Reserved | | Reserved |
| 11 | R | SDA_STUCK_NOT_REC OVERED | 0 | This bit indicates that an SDA stuck at low is not recovered after the recovery mechanism. |
| 10 | R | SLV_HOLD_RX_FIFO_F ULL | 0 | This bit indicates the BUS Hold in Slave mode due to the Rx FIFO being Full and an additional byte being received. |
| 9 | R | SLV_HOLD_TX_FIFO_E MPTY | 0 | This bit indicates the BUS Hold in Slave mode for the Read request when the Tx FIFO is empty. The Bus is in hold until the Tx FIFO has data to Transmit for the read request. |
| 8 | R | MST_HOLD_RX_FIFO_F ULL | 0 | This bit indicates the BUS Hold in Master mode due to Rx FIFO is Full and additional byte has been received. |
| 7 | R | MST_HOLD_TX_FIFO_E MPTY | 0 | The I2C master stalls the write transfer when Tx FIFO is empty, and the the last byte does not have the Stop bit set. This bit indicates the BUS hold when the master holds the bus because of the Tx FIFO being empty, and the the previous transferred command does not have the Stop bit set. |
| 6 | R | SLV_ACTIVITY | 0 | Slave FSM Activity Status. When the Slave Finite State Machine(FSM) is not in the IDLE state, this bit is set. <ul style="list-style-type: none"> • 0: Slave FSM is in IDLE state so the Slave part of I2C is not Active • 1: Slave FSM is not in IDLE state so the Slave part of I2C is Active |
| 5 | R | MST_ACTIVITY | 0 | Master FSM Activity Status. When the Master Finite State Machine(FSM) is not in the IDLE state, this bit is set. <ul style="list-style-type: none"> • 0: Master FSM is in IDLE state so the Master part of I2C is not Active • 1: Master FSM is not in IDLE state so the Master part of I2C is Active <p>NOTE: IC_STATUS[0]—that is, ACTIVITY bit—is the OR of SLV_ACTIVITY and MST_ACTIVITY bits.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 4 | R | RFF | 0 | <p>Receive FIFO Completely 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.</p> <ul style="list-style-type: none"> • 0: Receive FIFO is not full • 1: Receive FIFO is full |
| 3 | R | RFNE | 0 | <p>Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty.</p> <ul style="list-style-type: none"> • 0: Receive FIFO is empty • 1: Receive FIFO is not empty |
| 2 | R | TFE | 1 | <p>Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared.</p> <p>This bit field does not request an interrupt.</p> <ul style="list-style-type: none"> • 0: Transmit FIFO is not empty • 1: Transmit FIFO is empty |
| 1 | R | TFNF | 1 | <p>Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full.</p> <ul style="list-style-type: none"> • 0: Transmit FIFO is full • 1: Transmit FIFO is not full |
| 0 | R | ACTIVITY | 0 | I2C Activity Status |

735 .IC_STATUS Description

IC_TXFLR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:4 | N/A | Reserved | | Reserved |
| 3:0 | R | TXFLR | 0 | <p>Transmit FIFO Level</p> <p>Contains the number of valid data entries in the transmit FIFO.</p> |

736 .IC_TXFLR Description

IC_RXFLR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:4 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 3:0 | R | RXFLR | 0 | <p>Receive FIFO Level</p> <p>Contains the number of valid data entries in the receive FIFO.</p> |

737 .IC_RXFLR Description

IC_SDA_HOLD

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------|-----------|--|
| 31:24 | N/A | Reserved | | Reserved |
| 23:16 | R/W | IC_SDA_RX_HOLD | 0 | Sets the required SDA hold time in units of ic_clk period, when I2C acts as a receiver. |
| 15:0 | R/W | IC_SDA_TX_HOLD | 1 | Sets the required SDA hold time in units of ic_clk period, when I2C acts as a transmitter. |

738 .IC_SDA_HOLD Description

IC_TX_ABRT_SOURCE

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------------|-----------|--|
| 31:23 | R | TX_FLUSH_CNT | 0 | <p>This field indicates the number of Tx FIFO data commands that are flushed due to TX_ABRT interrupt. It is cleared whenever I2C is disabled.</p> <p>I2C can be in either Master-Transmitter or Slave-Transmitter mode.</p> |
| 22:21 | R | Reserved | | These bits are reserved. |
| 20 | R | ABRT_DEVICE_WRITE | 0 | <p>This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the Tx- FIFO consists of write commands.</p> <p>I2C is in Master mode.</p> |
| 19 | R | ABRT_DEVICE_SLVADD R_NOACK | 0 | <p>This is a master-mode-only bit. Master is initiating the DEVICE_ID transfer and the slave address sent was not acknowledged by any slave.</p> <p>I2C is in Master mode.</p> |
| 18 | R | ABRT_DEVICE_NOACK | 0 | <p>This is a master-mode-only bit. Master initiates the DEVICE_ID transfer and the device ID sent is not acknowledged by any slave.</p> <p>I2C is in Master mode.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------|-----------|--|
| 17 | R | ABRT_SDA_STUCK_AT_LOW | 0 | This is a master-mode-only bit. Master detects the SDA is Stuck at low for the IC_SDA_STUCK_AT_LOW_TIMEOUT value of ic_clks. I2C is in Master mode. |
| 16 | R | ABRT_USER_ABRT | 0 | This is a master-mode-only bit. Master has detected the transfer abort (IC_ENABLE[1]). I2C is in Master mode. |
| 15 | R | ABRT_SLVRD_INTX | 0 | 1: When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD (bit 8) of IC_DATA_CMD register. I2C is in Master-Transmitter mode. |
| 14 | R | ABRT_SLV_ARBLOST | 0 | 1: Slave lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time. NOTE: Even though the slave never “owns” the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then I2C no longer own the bus. I2C is in Slave-Transmitter mode. |
| 13 | R | ABRT_SLVFLUSH_TXFIFO | 0 | 1: Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO. I2C is in Slave-Transmitter mode. |
| 12 | R | ARB_LOST | 0 | 1: Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration. I2C can be either Master-Transmitter or Slave-Transmitter mode. |
| 11 | R | ABRT_MASTER_DIS | 0 | 1: User tries to initiate a Master operation with the Master mode disabled. I2C can be either Master-Transmitter or Master-Receiver mode. |
| 10 | R | ABRT_10B_RD_NORST | 0 | 1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the master sends a read command in 10-bit addressing mode. I2C is in Master-Receiver mode. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|--|
| 9 | R | ABRT_SBYTE_NORSTRT | 0 | To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets re-asserted. 1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to send a START Byte. I2C is in Master mode. |
| 8 | R | ABRT_HS_NORSTRT | 0 | 1: The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to use the master to transfer data in High Speed mode. I2C can be either Master-Transmitter or Master-Receiver mode. |
| 7 | R | ABRT_SBYTE_ACKDET | 0 | 1: Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). I2C is in Master mode. |
| 6 | R | ABRT_HS_ACKDET | 0 | 1: Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). I2C is in Master mode. |
| 5 | R | ABRT_GCALL_READ | 0 | 1: I2C in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1). I2C is in Master-Transmitter mode. |
| 4 | R | ABRT_GCALL_NOACK | 0 | 1: I2C in master mode sent a General Call and no slave on the bus acknowledged the General Call. I2C is in Master-Transmitter mode. |
| 3 | R | ABRT_TXDATA_NOACK | 0 | 1: This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledge from the remote slave(s). I2C is in Master-Transmitter mode. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|--|
| 2 | R | ABRT_10ADDR2_NOACK | 0 | 1: Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. I2C can be either Master-Transmitter or Slave-Receiver mode. |
| 1 | R | ABRT_10ADDR1_NOACK | 0 | 1: Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. I2C can be either Master-Transmitter or Master-Receiver mode. |
| 0 | R | ABRT_7B_ADDR_NOACK | 0 | 1: Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. I2C can be in either Master-Transmitter or Master-Receiver mode |

739 .IC_TS_ABRT_SOURCE Description

IC_SLV_DATA_NACK_ONLY

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R/W | NACK | 0 | Generate NACK. This NACK generation only occurs when I2C is a slavereceiver. If this register is set to a value of 1, it can only generate a NACK after a data byte is received; hence, the data transfer is aborted and the data received is not pushed to the receive buffer. When the register is set to a value of 0, it generates NACK/ACK, depending on normal criteria. <ul style="list-style-type: none"> • 1 - generate NACK after data byte received • 0 - generate NACK/ACK normally |

740 .IC_SLV_DATA_NACK_ONLY Description

IC_DMA_CR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:2 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 1 | R/W | TDMAE | 0 | <p>Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel.</p> <ul style="list-style-type: none"> • 0 - Transmit DMA disabled • 1 - Transmit DMA enabled |
| 0 | R/W | RDMAE | 0 | <p>Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel.</p> <ul style="list-style-type: none"> • 0 - Receive DMA disabled • 1 - Receive DMA enabled |

741 .IC_DMA_CR Description

IC_DMA_TDLR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:4 | N/A | Reserved | | Reserved |
| 3:0 | R | DMATDL | 0 | <p>Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic.</p> <p>It is equal to the watermark level; that is, the <code>dma_tx_req</code> 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.</p> |

742 .IC_DMA_TDLR Description

IC_DMA_RDLR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:4 | N/A | Reserved | | Reserved |
| 3:0 | R | DMARDL | 0 | <p>Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic.</p> <p>The watermark level = DMARDL+1; that is, <code>dma_rx_req</code> is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and RDMAE = 1.</p> <p>For instance, when DMARDL is 0, then <code>dma_rx_req</code> is asserted when 1 or more data entries are present in the receive FIFO.</p> |

743 .IC_DMA_RDLR Description

IC_SDA_SETUP

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|---|
| 31:8 | N/A | Reserved | | Reserved |
| 7:0 | R/W | SDA_SETUP | 0x64 | SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. IC_SDA_SETUP must be programmed with a minimum value of 2. |

744 .IC_SDA_SETUP Description

IC_ACK_GENERAL_CALL

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|-----------|---|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R/W | ACK_GEN_CALL | 0x1 | ACK General Call. When set to 1, I2C responds with a ACK (by asserting ic_data_oe) when it receives a General Call. When set to 0, the I2C does not generate General Call interrupts. |

745 .IC_ACK_GENERAL_CALL Description

IC_ENABLE_STATUS

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:3 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------|-----------|---|
| 2 | R | SLV_RX_DATA_LOST | 0 | <p>Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to setting IC_ENABLE[0] from 1 to 0.</p> <p>When read as 1, I2C is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK.</p> <p>NOTE: If the remote I2C master terminates the transfer with a STOP condition before the I2C has a chance to NACK a transfer, and IC_ENABLE[0] has been set to 0, then this bit is also set to 1.</p> <p>When read as 0, I2C is deemed to have been disabled without being actively involved in the data phase of a Slave- Receiver transfer.</p> <p>NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0.</p> |
| 1 | R | SLV_DISABLED WHILE 0 _BUSY | | <p>Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to setting bit 0 of the IC_ENABLE register from 1 to 0.</p> <p>This bit is set when the CPU writes a 0 to bit 0 of IC_ENABLE while:</p> <ul style="list-style-type: none"> (a) I2C is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master. <p>When read as 1, I2C is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in I2C (IC_SAR register) OR if the transfer is completed before bit 0 of IC_ENABLE is set to 0, but has not taken effect.</p> <p>NOTE: If the remote I2C master terminates the transfer with a STOP condition before the i2c has a chance to NACK a transfer and bit 0 of IC_ENABLE has been set to 0, then this bit will also be set to 1.</p> <p>When read as 0, I2C is deemed to have been disabled when there is master activity, or when the I2C bus is idle.</p> <p>NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------|-----------|---|
| 0 | R/W | ACK_GEN_CALL | 0x1 | <p>ACK General Call. When set to 1, I2C responds with a ACK (by asserting ic_data_oe) when it receives a General Call.</p> <p>When set to 0, the I2C does not generate General Call interrupts.</p> |

746 .IC_ENABLE_STATUS Description

IC_FS_SPKLEN

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|-----------|---|
| 31:8 | N/A | Reserved | | Reserved |
| 7:0 | R/W | IC_FS_SPKLEN | 0x07 | <p>This register must be set before any I2C bus transaction can take place to ensure stable operation.</p> <p>This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 1; hardware prevents values less than this being written, and if attempted, results in 1 being set.</p> |

747 .IC_FS_SPKLEN Description

IC_HS_SPKLEN

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | N/A | Reserved | | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------|-----------|--|
| 7:0 | R/W | IC_HS_SPKLEN | 0x02 | <p>This register must be set before any I2C bus transaction can take place to ensure stable operation.</p> <p>This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic.</p> <p>This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.</p> <p>The minimum valid value is 1; hardware prevents values less than this being written, and if attempted, results in 1 being set.</p> <p>This register is implemented only if the component is configured to support HS mode; that is, if the IC_MAX_SPEED_MODE parameter is set to 3.</p> |

748 .IC_HS_SPKLEN Description

IC_CLR_RESTART_DET

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_RESTART_DET | 0 | Read this register to clear the RESTART_DET interrupt (bit 12) of the IC_RAW_INTR_STAT register. |

749 .IC_CLR_RESTART_DET Description

IC_COMP_PARAM_1

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------|-----------|--|
| 31:24 | N/A | Reserved | | Reserved |
| 23:16 | R | TX_BUFFER_DEPTH | 0x08 | <ul style="list-style-type: none"> • 0x00 = Reserved • 0x01 = 2 • 0x02 = 3 ... • 0xFF = 256 |
| 15:8 | R | RX_BUFFER_DEPTH | 0x08 | <ul style="list-style-type: none"> • 0x00 = Reserved • 0x01 = 2 • 0x02 = 3 ... • 0xFF = 256 |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|--|
| 7 | R | ADD_ENCODED_PAR_AMS | 0x1 | <p>Reading 1 in this bit means that the capability of reading these encoded parameters via software has been included.</p> <p>Otherwise, the entire register is 0 regardless of the setting of any other parameters that are encoded in the bits.</p> |
| 6 | R | HAS_DMA | 0x1 | Configures the inclusion of DMA handshaking interface signals. |
| 5 | R | INTR_IO | 0x1 | <p>Controls which interrupt outputs are present.</p> <ul style="list-style-type: none"> • 0: Individual - each interrupt source has its own output • 1: Combined - all interrupt sources are combined in to a single output |
| 4 | R | HC_COUNT_VALUES | 0x0 | <p>Setting this parameter to 1 will cause the CNT registers to read only.</p> <p>Setting this parameter to 0 will allow the CNT registers to be writeable.</p> <p>Regardless of the setting, the CNT registers are always readable and have reset values from the corresponding configuration parameters, which may be user defined or else derived.</p> |
| 3:2 | R | MAX_SPEED_MODE | 0x3 | <ul style="list-style-type: none"> • 0x0 = Reserved • 0x1 = Standard • 0x2 = Fast • 0x3 = High |
| 1:0 | R | CLR_RESTART_DET | 0 | Read this register to clear the RESTART_DET interrupt (bit 12) of the IC_RAW_INTR_STAT register. |

750 .IC_COMP_PARAM_1 Description

IC_COMP_VERSION

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------|------------|---------------------------------------|
| 31:0 | R | IC_COMP_VERSION | 0x3230302a | Specific I2C component version number |

751 .IC_COMP_VERSION Description

IC_COMP_TYPE

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|------------|---|
| 31:0 | R | IC_COMP_TYPE | 0x44570140 | This assigned unique hex value is constant. |

752 .IC_COMP_TYPE Description

IC_SCL_STUCK_AT_LOW_TIMEOUT

This register is used to store the duration, measured in ic_clk cycles, used to generate an Interrupt (SCL_STUCK_AT_LOW) if SCL is held low for the IC_SCL_STUCK_LOW_TIMEOUT duration.

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|------------|---|
| 31:0 | R/W | IC_SCL_STUCK_LOW_TIMEOUT | 0xFFFFFFFF | I2C generates the interrupt to indicate SCL stuck at low if it detects the SCL stuck at low for the IC_SCL_STUCK_LOW_TIMEOUT in units of ic_clk period. |

753 .IC_SCL_STUCK_AT_LOW_TIMEOUT Description

IC_SDA_STUCK_AT_LOW_TIMEOUT

This register is used to store the duration, measured in ic_clk cycles, used to recover the Data (SDA) line through sending SCL pulses if SDA is held low for the mentioned duration.

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|------------|--|
| 31:0 | R/W | IC_SDA_STUCK_LOW_TIMEOUT | 0xFFFFFFFF | I2C initiates the recovery of SDA line through enabling the SDA_STUCK_RECOVERY_EN (IC_ENABLE[3]) register bit, if it detects the SDA stuck at low for the IC_SDA_STUCK_LOW_TIMEOUT in units of ic_clk period. |

754 .IC_SDA_STUCK_AT_LOW_TIMEOUT Description

IC_CLR_SCL_STUCK_DET

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | CLR_SCL_STUCK | 0 | Read this register to clear the SCL_STUCK_DET interrupt (bit 14) of the IC_RAW_INTR_STAT register. |

755 .IC_SDA_STUCK_AT_LOW_TIMEOUT Description

IC_DEVICE_ID

This register contains the Device-ID of the component, which includes 12 bits of manufacturer name, 9 bits of part identification and 3 bits of die-version.

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|--|
| 31:1 | N/A | Reserved | | Reserved |
| 0 | R | DEVICE-ID | 0x1 | Contains the Device-ID of the component. |

756 .IC_DEVICE_ID Description

IC_SMBUS_CLOCK_LOW_SEXT

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------------|-------------|---|
| 31:0 | R/W | SMBUS_CLK_LOW_SE XT_TIMEOUT | 32'hFFFFFFF | <p>This field is used to detect the Slave Clock Extend timeout (tLOW:SEXT) in master mode extended by the slave device in one message from the initial START to the STOP.</p> <p>The values in this register are in units of ic_clk period.</p> |

757 .IC_SMBUS_CLOCK_LOW_SEXT Description

IC_SMBUS_CLOCK_LOW_MEXT

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------------|-------------|---|
| 31:0 | R/W | SMBUS_CLK_LOW_ME XT_TIMEOUT | 32'hFFFFFFF | <p>This field is used to detect the Master extend SMBus clock (SCL) timeout defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP in Master mode.</p> <p>The values in this register are in units of ic_clk period.</p> |

758 .IC_SMBUS_CLOCK_LOW_MEXT Description

IC_SMBUS_THIGH_MAX_IDLE_COUNT

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------------|-----------|--|
| 31:16 | N/A | Reserved | | Reserved |
| 15:0 | R/W | SMBUS_THIGH_MAX_ BUS_IDLE_CNT | 16'hFFFF | <p>This field is used to set the required Bus-Idle time period used when a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait to ensure that a transfer is not currently in progress.</p> <p>The values in this register are in units of ic_clk period.</p> |

759 .IC_SMBUS_THIGH_MAX_IDLE_COUNT Description

IC_SMBUS_INTR_STAT

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:0 | N/A | Reserved | - | Reserved |

760 .IC_SMBUS_INTR_STAT Description

IC_SMBUS_INTR_MASK

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:0 | N/A | Reserved | | Reserved |

761 .IC_SMBUS_INTR_MASK Description

IC_SMBUS_INTR_RAW_STATUS

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:0 | N/A | Reserved | 0 | Reserved |

762 .IC_SMBUS_INTR_RAW_STATUS Description

IC_CLR_SMBUS_INTR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:0 | N/A | Reserved | 0 | Reserved |

763 .IC_CLR_SMBUS_INTR Description

IC_OPTIONAL_SAR

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 15:0 | N/A | Reserved | | Reserved |

764 .IC_OPTIONAL_SAR Description

IC_SMBUS_UDID_LSB

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------|-------------|---|
| 31:0 | R/W | IC_SMBUS_ARP_UDID_LSB | 32'hFFFFFFF | This field is used to store the LSB 32 bit value of slave unique device identifier used in Address Resolution Protocol. |

765 .IC_SMBUS_ARP_UDID_LSB Description

16.10 I2S/PCM Master and Slave

16.10.1 General Description

There are three I²S controllers - one in the MCU HP peripherals (I2S_2CH), one in the MCU ULP subsystem (ULP_I2S) and one in the security/NWP subsystem. Each I²S controller supports PCM mode of operation also.

I2S is a protocol used for digital stereo audio. It is used in systems that process digital audio signals, such as:

- A/D and D/A converters
- digital signal processors
- error correction for compact disc and digital recording
- digital filters

- digital input/output interfaces

16.10.2 Features

I²S

The each I²S controllers support the following features:

- The I²S_2CH supports two stereo channels while the ULP_I²S and the NWP/Security subsystem I²S support one stereo channel
- Programmable Audio data resolutions of 12, 16, 20 and 24 bits
- Supported audio sampling rates are 8, 11.025, 16, 22.05, 24, 32, 44.1, 48, 88.2, 96 and 192 kHz
- Support for master and slave modes
- Full duplex communication due to the independence of transmitter and receiver
- The PCM mode of operation supports the following additional features
 - Mono audio data is supported
 - Supports two modes for data transmission with respect to the Frame Synchronization signal – the MS bit is transmitted in the same clock cycle as the Frame Synchronization signal is asserted or one clock cycle after the Frame Synchronization signal is asserted
- Programmable FIFO thresholds with maximum FIFO depth of 8 and support for DMA
- Supports generation of interrupts for different events

The I²S in the MCU ULP subsystem supports the following additional power-save features:

- After the DMA is programmed in PS2 state for I²S transfers, the MCU can switch to PS1 state (processor is shutdown) while the I²S controller continues with the data transfer
- In PS1 state (ULP Peripheral mode) the I²S controller completes the data transfer and, triggered by the Peripheral Interrupt, shifts either to the sleep state (without processor intervention) or the active state

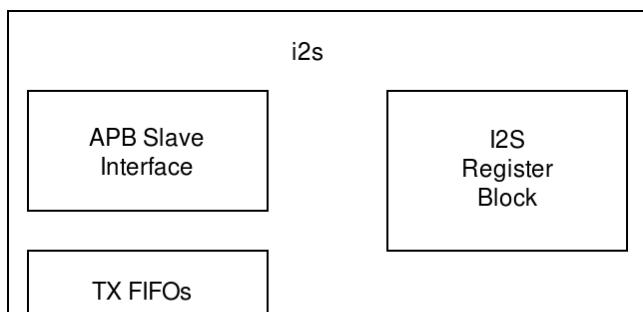
The NWP/Security subsystem's I²S controller allows direct bridging between audio data and the wireless link without the MCU's intervention (e.g., in a wireless headset application)

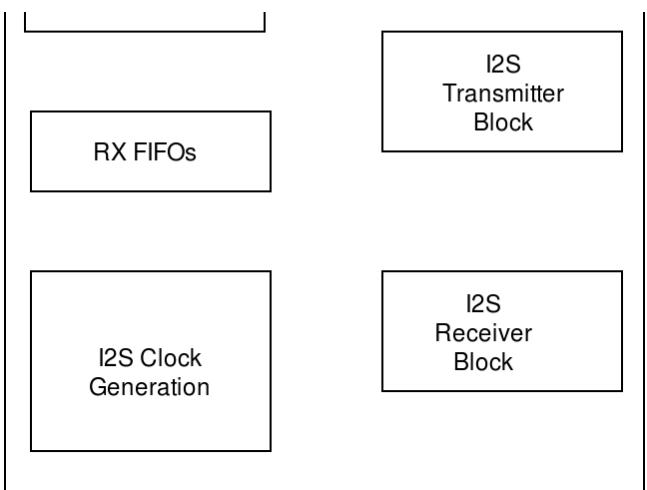
PCM

- The PCM interface works in slave mode which can transmit and receive serial data to and from an off-chip PCM master.
- Supports full-duplex data transfer with a PCM master.
- Supports 12, 16, 20 and 24-bit frame sizes.
- Data is driven at the rising edge of clock and sampled at the falling edge of clock.
- Maximum operating frequency is 24MHz.
- PCM Master/slave mode can be configured (can be configured through the register named i2s_master_slave_mode 1-master, 0-slave (present in misc config registers))

16.10.3 Functional Description

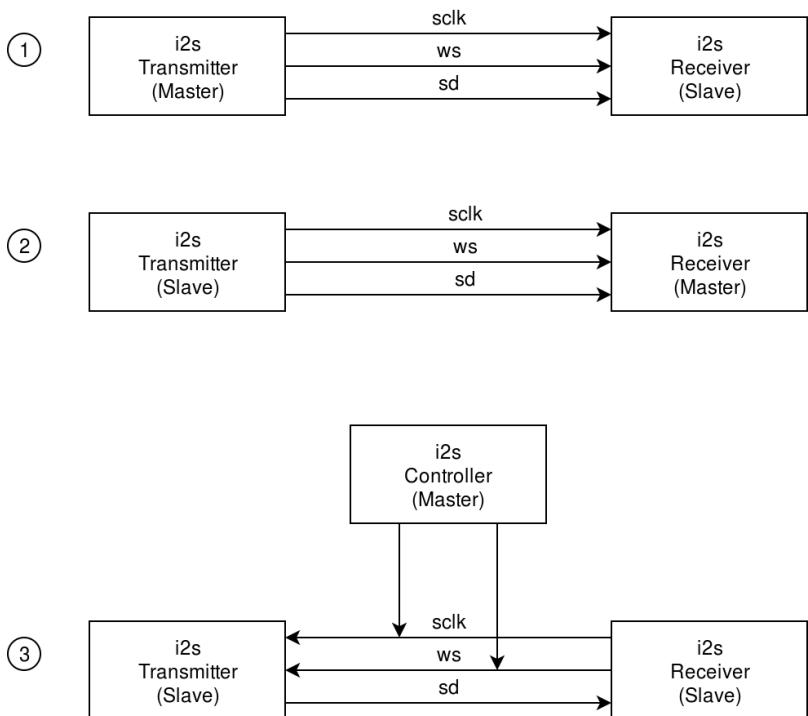
Figure 1 illustrates a block diagram of the I²S.





I2S Block diagram

The bus consists of a serial data line (sd), a word select line (ws), and a serial clock (sclk). The serial data line is time multiplexed to allow the transfer of two data streams (such as, left and right stereo data). It supports up to two data channels for both transmit and receive operations. Figure 2 illustrates simple system configurations for the I2S component.



Simple System Configurations for I2S

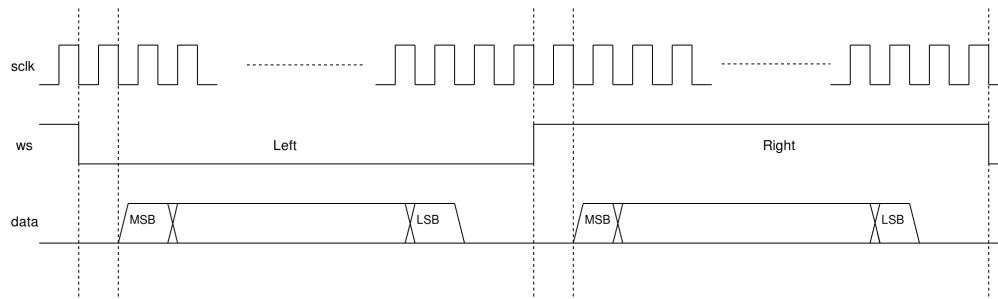
The master is responsible for generating the shared sclk and ws clocking signals. In complex systems where there may be several transmitters and receivers, a separate system master can be used.

As illustrated in the figure 2, this system master can also be combined with one of the transmitters or receivers in the system. The “controller” in this example is enabled and disabled by configuring the component to act as a master and by programming the clock enable and clock configuration registers.

The serial data is transmitted in two's complement format with the most significant bit (MSB) first. This means that the transmitter and receiver can have different word lengths, and neither the transmitter nor receiver needs to know what size words the other can handle. If the word being transferred is too large for the receiver, the least significant bits (LSB) are truncated. Similarly, if the word size is less than what the receiver can handle, the data is zero padded.

The word select line is used to time the multiplexed data streams. For instance, when ws is low, the word

being transferred is left stereo data; when ws is high, the word being transferred is right stereo data. This format is illustrated in Figure3. For standard I2S formats, the MSB of a word is sent one sclk cycle after a ws change. Serial data sent by the transmitter can be synchronized with either the negative edge or positive edge of the sclk signal. However, the receiver must latch the serial data on the rising edge of sclk.



I2S Stereo Frame Format

The I2S component can be configured to support up to two stereo I2S transmit (TX) channels. These channels can operate in either master or slave mode. By default, I2S is configured in slave mode. Stereo data pairs (such as, left and right audio data) written to a TX channel via the APB bus are shifted out serially on the appropriate serial data out line (sdo0, sdo1, sdo2, sdo3). The shifting is timed with respect to the serial clock (sclk) and the word select line (ws). The instantiation of the I2S transmitter block and the number of TX channels is determined by the two configuration parameters: Transmitter Block Enabled (I2S_TRANSMITTER_BLOCK) and Number of Transmit Channels (I2S_TX_CHANNELS), respectively.

Each TX channel is initially configured with a maximum audio data resolution as set by the Maximum Audio Resolution parameter (I2S_TX_WORDSIZE_x, where x is the channel number). A TX channel can be reprogrammed during operation to any supported audio data resolution that is less than I2S_TX_WORDSIZE_x.

I2S can be configured to support up to four stereo I2S receive (RX) channels. These channels can operate in either master or slave mode. By default, I2S is configured in slave mode. Stereo data pairs (such as, left and right audio data) are received serially from a data input line (sdi0, sdi1, sdi2, sdi3). These data words are stored in RX FIFOs until they are read via the APB bus. The receiving is timed with respect to the serial clock (sclk) and the word select line (ws).

I2S master PLL must be able to output following clock frequencies. The clock frequency is related to the sampling frequency of I2S module according to the relation given below.

$$\text{Clock frequency} = 2 * \text{bit_width} * \text{Sampling_freq.}$$

| Sampling freq(kHz) | Clock Frequency (MHz) | | | |
|---------------------------|------------------------------|---------|---------|---------|
| | 16 Bits | 32 Bits | 48 Bits | 64 bits |
| 8 | 0.256 | 0.512 | 0.768 | 1.024 |
| 16 | 0.512 | 1.024 | 1.536 | 2.048 |
| 32 | 1.024 | 2.048 | 4.2336 | 5.6448 |
| 44.1 | 1.4112 | 2.8224 | 4.2336 | 5.6448 |
| 48 | 1.536 | 3.072 | 4.608 | 6.144 |
| 88.2 | 2.8224 | 5.6448 | 8.4672 | 11.2896 |
| 96 | 3.072 | 6.144 | 9.216 | 12.288 |
| 192 | 6.144 | 12.288 | 18.432 | 24.576 |

766 .I2S Clock Frequencies

16.10.4 Programming Sequence of I2S

I2S Enabling

enable the I2S component before any data can be received or transmitted into the FIFOs.

To enable the component, set the I2S Enable (IEN) bit of the I2S Enable Register (IER) to 1. When you disable the device, it acts as a global disable. To disable I2S, set IER[0] to 0. After disable, the following events occur:

- TX and RX FIFOs are cleared, and read/write pointers are reset;
- Any data in the process of being transmitted or received is lost;
- All other programmable enables (such as transmitter/receiver block enables and individual TX/RX channel enables) in the component are overridden;
- Generation of master mode clock signals sclk_en, ws_out and sclk_gate are disabled

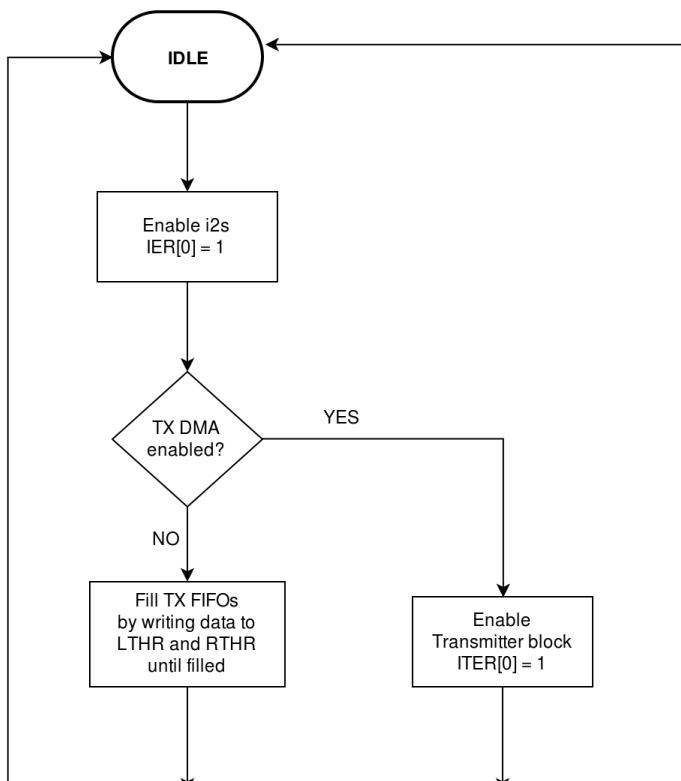
When I2S is enabled and configured as a master, the device always starts in the left stereo data cycle ($ws = 0$), and one sclk cycle later transitions to the right stereo data cycle ($ws = 1$). This allows for half a frame of sclks to write data to the TX FIFOs and to ensure that any connected slave receivers do not miss the start of the data frame (for instance, the ws 1-to-0 transition) once the sclk restarts.

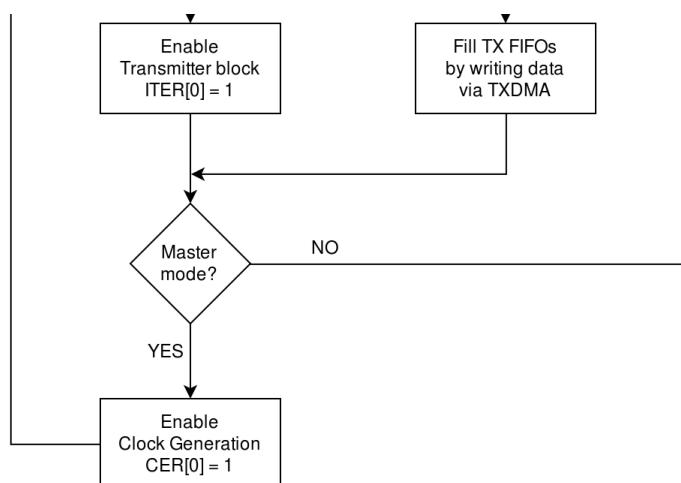
I2S as Transmitter

The I2S component can be configured to support up to two stereo I2S transmit (TX) channels. These channels can operate in either master or slave mode. By default, I2S is configured in slave mode. Stereo data pairs (such as, left and right audio data) written to a TX channel via the APB bus are shifted out serially on the appropriate serial data out line (sdo0, sdo1, sdo2, sdo3). The shifting is timed with respect to the serial clock (sclk) and the word select line (ws).

Figure 4 illustrates the basic usage flow for I2S when it acts as a transmitter.

Software Flow





Basic Usage Flow for I2S as Transmitter

Transmitter Block Enable

The Transmitter Block Enable (TXEN) bit of the I2S Transmitter Enable Register (ITER) globally turns on and off all of the configured TX channels. To enable the transmitter block, set ITER[0] to 1. To disable the block, set ITER[0] to 0. When the transmitter block is disabled, the following events occur:

- Outgoing data is lost and the channel outputs are held low;
- Data in the TX FIFOs are preserved and the FIFOs can be written to;
- Any previous programming (like changes in word size, threshold levels, and so on) of the TX channels is preserved.
- Any individual TX channel enables are overridden.

When the transmitter block is enabled, if there is data in the TX FIFOs, the channel resumes transmission on the next left stereo data cycle.

When the block is disabled, perform any of the following procedures:

- Program (or further program) TX channel registers
- Flush the TX FIFOs by programming the Transmitter FIFOs Reset bit of the Transmitter FIFO Flush Register (TXFFR[0] = 1)
- Flush an individual channel's TX FIFO by programming the Transmit Channel FIFO Reset (TXCHFR) bit of the Transmit FIFO Flush Register (TFFx[0] = 1, where x is the channel number)

Transmit Channel Enable

Each transmit channel has its own enable/disable that can be set independently of the other channels to allow the reprogramming of a channel and to flush the channel's TX FIFOs while other TX channels are transmitting. This enable/disable is controlled by bit 0 of the Transmitter Enable Register (TERx, where x is the channel number). For example, to enable TX Channel 1, write a 1 to TER1[0]. To disable this channel, write a 0 to TER1[0]. When a TX channel is disabled, the following occurs:

- Outgoing stereo data is lost;
- Channel output is held low;
- Data in the TX FIFO is preserved, and the FIFO can be written to; and
- Any previous programming of the TX channel's registers is preserved, and the registers can be further reprogrammed.

When a TX channel is disabled, flush the channel's TX FIFO by programming the Transmit Channel FIFO Reset (TXCHFR) bit of the Transmit FIFO Flush (TFFx[0] = 1, where x is the channel number). When the TX channel is enabled, if there is data in the TX FIFO, the channel resumes transmission on the next left stereo data cycle (such as, when the ws line goes low)

Transmit Channel Audio Data Resolution

Each TX channel is initially configured with a maximum audio data resolution as set by the Maximum Audio Resolution parameter. A TX channel can be reprogrammed during operation to any supported audio data resolution that is less than Maximum Audio Resolution. Changes to the resolution are programmed via the Word Length (WLEN) bits of the Transmitter Configuration Registers (TCRx[2:0], where x is the channel number). The channel must be disabled prior to any resolution changes. On reset or if an invalid resolution is selected, the TX channel's audio data resolution defaults back to the initial value.

Transmit Channel Interrupts

All interrupts in I2S are active interrupts. Each TX channel generates two interrupts: TX FIFO Empty and Data Overrun.

- TX FIFO Empty interrupt – This interrupt is asserted when the empty trigger threshold level for the TX FIFO is reached. When this interrupt is included on the I/O, it appears on the outputs tx_emp_x_intr (where x is the channel number). A TX FIFO Empty interrupt is cleared by writing data to the TX FIFO to bring its level above the empty trigger threshold level for the channel.
- Data Overrun interrupt – This interrupt is asserted when an attempt is made to write to a full TX FIFO (any data being written is lost while data in the FIFO is preserved). When this interrupt is included on the I/O, it appears on the outputs tx_or_x_intr (where x is the channel number). A Data Overrun interrupt is cleared by reading the Transmit Channel Overrun (TXCHO) bit [0] of the Transmit Overrun Register (TORx, where x is the channel number).

The interrupt status of any TX channel can be determined by polling the Interrupt Status Register (ISR x , where x is the channel number). The TXFE bit [4] indicates the status of the TX FIFO Empty interrupt, while the TXFO bit [5] indicates the status of the Data Overrun interrupt. Both the TX FIFO Empty and Data Overrun interrupts can be masked off by writing a 1 in the Transmit Empty Mask (TXFEM) and Transmit Overrun Mask (TXFOM) bits of the Interrupt Mask Register (IMR x , where x is the channel number), respectively. This prevents the interrupts from driving their output lines, however, the ISR x always shows the current status of the interrupts regardless of any masking.

Writing to a Transmit Channel

The stereo data pairs to be transmitted by a TX channel are written to the TX FIFOs via the Left Transmit Holding Register (LTHR x , where x is the channel number) and the Right Transmit Holding Register (RTHR x , where x is the channel number). All stereo data pairs must be written using the following two-stage process:

1. Write left stereo data to LTHR x
2. Write right stereo data to RTHR x .

All enabled TX channels starting from the lowest-numbered enabled channel. After a stereo data pair is transmitted, the component will point to the next enabled channel.

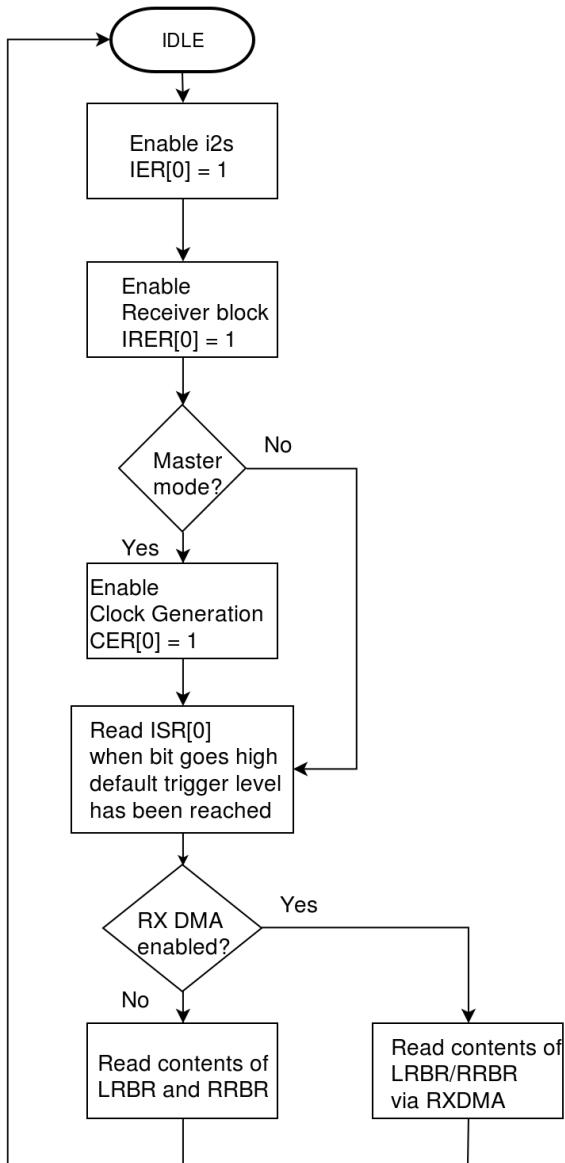
When I2S is enabled, if the TX FIFO is empty and data is not written to the FIFOs before the next left cycle, the channel outputs zeros for a full frame (left and right cycle). Transmission only commences if there is data in the TX FIFO prior to the transition to the left data cycle. In other words, if the start of the frame is missed, the channel output idles until the next available frame.

I2S as Receiver

I2S supports up to two stereo I2S receive (RX) channels. These channels can operate in either master or slave mode. By default, I2S is configured in slave mode. Stereo data pairs (such as, left and right audio data) are received serially from a data input line (sdi0, sdi1, sdi2, sdi3). These data words are stored in RX FIFOs until they are read via the APB bus. The receiving is timed with

respect to the serial clock (sclk) and the word select line (ws). Figure 5 illustrates the basic usage flow for I2S when it acts as a receiver.

Software Flow



Basic Usage Flow for I2S as Receiver

Receiver Block Enable

The Receiver Block Enable (RXEN) bit of the I2S Receiver Enable Register (IRER) enables/disables all configured RX channels. To enable the receiver block, set IRER[0] to '1.' To disable the block, set this bit to '0.' When the receiver block is disabled, the following events occur:

- Incoming data is lost
- Data in the RX FIFOs is preserved and the FIFOs can be read.
- Any previous programming (such as changes in word size, threshold levels, and so on) of the RX channels is preserved

- Any individual RX channel enable is overridden. Enabling the channel resumes receiving on the next left stereo data cycle (for instance, when ws goes low).

When the block is disabled, you can perform any of the following procedures:

- Program (or further program) the RX channel registers;
- Flush the RX FIFOs by programming the Receiver FIFOs Reset (RXFR) bit of the Receiver FIFO Flush Register (RXFFR[0] = 1).
- Flush an individual channel's RX FIFO by programming the Receive Channel FIFO Reset (RXCHFR) bit of the Receive FIFO Flush Register (RFFx [0] = 1, where x is the channel number).

Receive Channel Enable

Each RX channel has its own enable/disable that can be set independently of the other channels to allow programming of the channel and to clear the channel's RX FIFO while other RX channels are still receiving data. This enable/disable is controlled by bit 0 of the Receiver Enable Register (RERx[0], where x is the channel number). For example, to enable RX Channel 1, write a 1 to RER1[0]. To disable this channel, write a 0 to RER1[0]. When the RX channel is disabled, the following occurs:

- Incoming data is lost
- Data in the RX FIFO is preserved
- FIFO can be read
- Previous programming of the RX channel is preserved.
- RX channel can be further programmed.

When the RX channel or block is disabled, flush the channel's RX FIFO by writing 1 in bit 0 of the Receive FIFO Flush Register (RFFx, where x is the channel number). When the channel is enabled, it resumes receiving on the next left stereo data cycle (for instance. when ws line goes low).

Receive Channel Audio Data Resolution

Each RX channel is initially configured with a maximum audio data resolution. A RX channel can be reprogrammed during operation to any supported audio data resolution that is less than Maximum Audio Resolution. Changes to the resolution are programmed via the Word Length (WLEN) bits of the Transmitter Configuration Registers (RCRx[2:0], where x is the channel number). The channel must be disabled prior to any resolution changes. On reset or if an invalid resolution is selected, the RX channel's audio data resolution defaults back to the initial value.

Receive Channel Interrupts

Each RX channel generates two interrupts: RX FIFO Data Available and Data Overrun.

- RX FIFO Data Available interrupt – This interrupt is asserted when the trigger level for the RX FIFO is reached. When this interrupt is included on the I/O, it appears on the outputs rx_da_x_intr (where x is the channel number). This interrupt is cleared by reading data from the RX FIFO until its level drops below the data available trigger level for the channel.
- Data Overrun interrupt – This interrupt is asserted when an attempt is made to write received data to a full RX FIFO (any data being written is lost while data in the FIFO is preserved). When this interrupt is included on the I/O, it appears on the outputs rx_or_x_intr (where x is the channel number). This interrupt is cleared by reading the Receive Channel Overrun (RXCHO) bit [0] of the Receive Overrun Register (RORx, where x is the channel number).

The interrupt status of any RX channel can be determined by polling the Interrupt Status Register (ISR_x, where x is the channel number). The RXDA bit [0] indicates the status of the RX FIFO Data Available interrupt. the RXFO bit [1] indicates the status of the RX FIFO Data Overrun interrupt.

Both the Receive Empty Threshold and Data Overrun interrupts can be masked by writing a 1 in the Receive Empty Threshold Mask (RDM) and Receive Overrun Mask (ROM) bits of the Interrupt Mask Register (IMRx, where x is the channel number), respectively. This prevents the interrupts from driving their output lines, however, the ISR_x always shows the current status of the interrupts regardless of any masking.

Reading from a Receive Channel

The stereo data pairs received by a RX channel are written to the left and right RX FIFOs. These FIFOs can be read via the Left Receive Buffer Register (LBRRx, where x is the channel number) and the Right Receive Buffer Register (RRBRx, where x is the channel number). All stereo data pairs must be read using the following two-stage process:

1. Read the left stereo data from LBRRx.
2. Read the right stereo data from RRBRx.

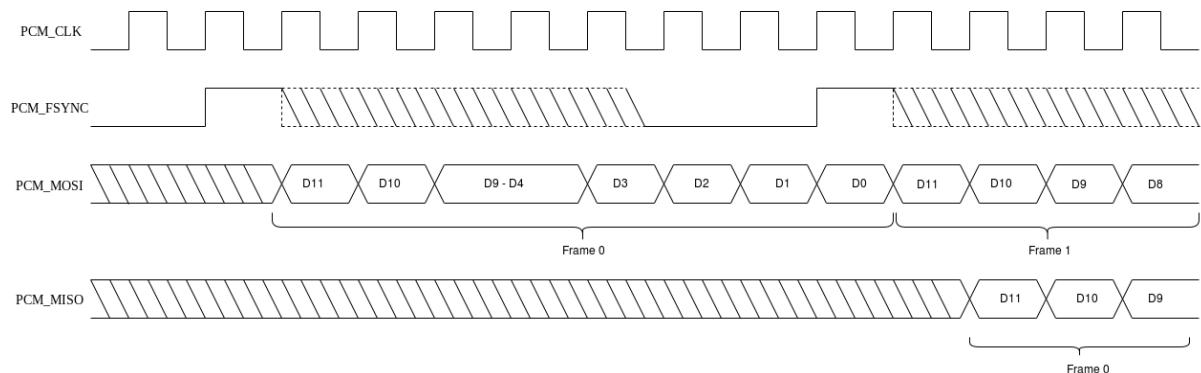
16.10.5 PCM

The PCM interface can transmit and receive serial data to and from an off-chip PCM master/slave. It uses the I2S Master/slave engine for data transmission and reception.

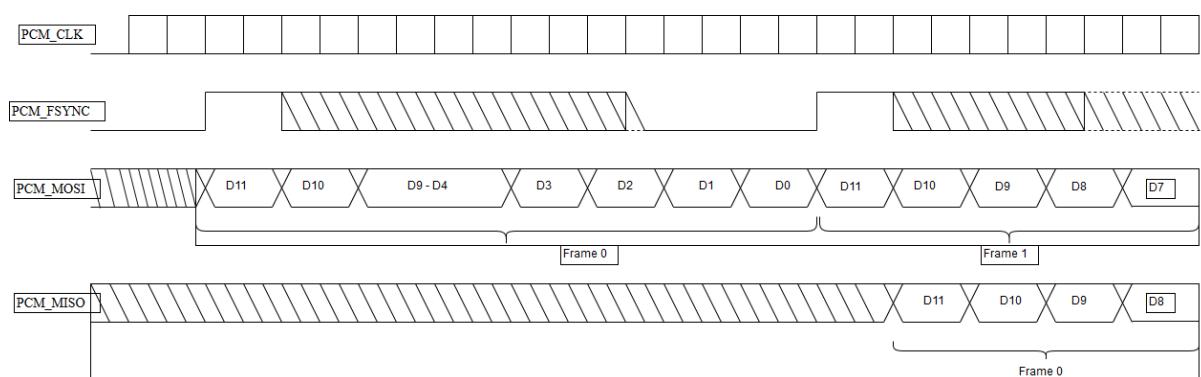
As a result of this, there is a delay of one frame while transmitting data. The following are the features supported by the PCM interface.

- Supports full-duplex data transfer with a PCM master/slave.
- Supports two modes for data transmission with respect to the Frame Synchronization signal – the MS bit is transmitted in the same clock cycle as the Frame Synchronization signal is asserted or one clock after the Frame Synchronization signal is asserted.
- Supports 12, 16, 20 and 24-bit frame sizes.
- Data is driven at the rising edge of clock and sampled at the falling edge of clock.

Figures 6 and 7 shows the timing diagrams of the PCM interface module. The diagrams are shown for the 12-bit frame size. The same diagrams apply for 12, 16, 20 and 24-bit frame sizes. If there are extra clock cycles between each pcm_fsync signal assertion, the data during those clock cycles is ignored. The PCM_FSYNC signal may be asserted for 1 clock (short FSYNC) or more clocks (long FSYNC) but should be deasserted before the last bit is transmitted.



Timing Diagram for 12-bit Frame Size and FSYNC Asserted with LS bit



Timing Diagram for 12-bit Frame Size and FSYNC Asserted with MS bit

16.10.6 Register Summary

Base Address: 0x4705_0000

ULP_I2S Base Address: 0x2404_0400

| Register Name | Offset | Description |
|---------------|--------|--|
| I2S_IER | 0x000 | i2s Enable Register |
| I2S_IRER | 0x004 | I2S Receiver Block Enable Register |
| I2S_ITER | 0x008 | I2S Transmitter Block Enable Register |
| I2S_CER | 0x0C | Clock Enable Register |
| I2S_CCR | 0x010 | Clock Configuration Register |
| I2S_RXFFR | 0x014 | Receiver Block FIFO Register |
| I2S_TXFFR | 0x018 | Transmitter Block FIFO Register |
| I2S_LRBR0 | 0x020 | Left Receive Buffer Register for Channel 0 |
| I2S_LTHR0 | 0x020 | Left Transmit Holding Register for Channel 0 |
| I2S_RRBR0 | 0x024 | Right Receive Buffer Register for Channel 0 |
| I2S_RTHR0 | 0x024 | Right Transmit Holding Register for Channel 0 |
| I2S_RERO | 0x028 | Receive Enable Register for Channel 0 |
| I2S_TER0 | 0x02C | Transmit Enable Register for Channel 0 |
| I2S_RCR0 | 0x030 | Receive Configuration Register for Channel 0 |
| I2S_TCR0 | 0x034 | Transmit Configuration Register for Channel 0 |
| I2S_ISR0 | 0x038 | Interrupt Status Register for Channel 0 |
| I2S_IMR0 | 0x03C | Interrupt Mask Register for Channel 0 |
| I2S_ROR0 | 0x040 | Receive Overrun Register for Channel 0 |
| I2S_TOR0 | 0x044 | Transmit Overrun Register for Channel 0 |
| I2S_RFCR0 | 0x048 | Receive FIFO Configuration Register for Channel 0 |
| I2S_TFCR0 | 0x04C | Transmit FIFO Configuration Register for Channel 0 |
| I2S_RFF0 | 0x050 | Receive FIFO Flush Register for Channel 0 |
| I2S_TFF0 | 0x054 | Transmit FIFO Flush Register for Channel 0 |
| I2S_LRBR1 | 0x060 | Left Receive Buffer Register for Channel 1 |
| I2S_LTHR1 | 0x060 | Left Transmit Holding Register for Channel 1 |
| I2S_RRBR1 | 0x064 | Right Receive Buffer Register for Channel 1 |
| I2S_RTHR1 | 0x064 | Right Transmit Holding Register for Channel 1 |
| I2S_RER1 | 0x068 | Receive Enable Register for Channel 1 |
| I2S_TER1 | 0x06C | Transmit Enable Register for Channel 1 |
| I2S_RCR1 | 0x070 | Receive Configuration Register for Channel 1 |

| Register Name | Offset | Description |
|------------------|--------|--|
| I2S_TCR1 | 0x074 | Transmit Configuration Register for Channel 1 |
| I2S_ISR1 | 0x078 | Interrupt Status Register for Channel 1 |
| I2S_IMR1 | 0x07C | Interrupt Mask Register for Channel 1 |
| I2S_ROR1 | 0x080 | Receive Overrun Register for Channel 1 |
| I2S_TOR1 | 0x084 | Transmit Overrun Register for Channel 1 |
| I2S_RFCR1 | 0x088 | Receive FIFO Configuration Register for Channel 1 |
| I2S_TFCR1 | 0x08C | Transmit FIFO Configuration Register for Channel 1 |
| I2S_RFF1 | 0x090 | Receive FIFO Flush Register for Channel 1 |
| I2S_TFF1 | 0x094 | Transmit FIFO Flush Register for Channel 1 |
| I2S_RXDMA | 0x1C0 | Receiver Block DMA Register |
| I2S_RRXDMA | 0x1C4 | Reset Receiver Block DMA Register |
| I2S_TXDMA | 0x1C8 | Transmitter Block DMA Register |
| I2S_RTXDMA | 0x1CC | Reset Transmitter Block DMA Register |
| I2S_COMP_PARAM_2 | 0x1F0 | Component Parameter 2 Register |
| I2S_COMP_PARAM_1 | 0x1F4 | Component Parameter 1 Register |
| I2S_COMP_VERSION | 0x1F8 | Component Version ID |
| I2S_COMP_TYPE | 0x1FC | DesignWare Component Type |

767 . Register Summary Table

16.10.7 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, N/A = Reserved

I2S ENABLE REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | R/W | IEN | 0 | i2s enable. A disable on this bit overrides any other block or channel enables and flushes all FIFOs. 1: enable i2s 0: disable i2s |

768 . i2s Enable Register Description

I2S RECEIVER BLOCK ENABLE REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | R/W | RXEN | 0 | Receiver block enable. A disable on this bit overrides any individual receive channel enables. 1: enable receiver 0: disable receiver |

769 .I2S Receiver Block Enable Register Description

I2S TRANSMITTER BLOCK ENABLE REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | R/W | TXEN | 0 | Transmitter block enable. A disable on this bit overrides any individual transmit channel enables. 1: enable transmitter 0: disable transmitter |

770 .I2S Transmitter Block Enable Register Description

I2S CLOCK ENABLE REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | R/W | CLKEN | 0 | Clock generation enable/disable. This bit enables/disables the clock generation signals when i2s is a master: sclk_en, ws_out, and sclk_gate. 1: enable 0: disable |

771 .Clock Enable Register Description

I2S CLOCK CONFIGURATION REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|----------------------------|
| 31:5 | N/A | Reserved | 0 | Reserved and read as zero. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|--|
| 4:3 | R/W | WSS | 0x01 | <p>These bits are used to program the number of sclk cycles for which the word select line (ws_out) stays in the left or right sample mode:</p> <p>0: 16 clock cycles 1: 24 clock cycles 2: 32 clock cycles</p> <p>The I2S Clock Generation block must be disabled (CER[0] = 0) prior to any changes in this value.</p> |
| 2:0 | R/W | SCLKG | 0x04 | <p>These bits are used to program the gating of sclk:</p> <p>0: No clock gating 1: Gate after 12 clock cycles 2: Gate after 16 clock cycles 3: Gate after 20 clock cycles 4: Gate after 24 clock cycles</p> <p>The programmed gating value should be greater than or equal to the largest programmed audio resolution to prevent the truncating of RX/TX data.</p> <p>The I2S Clock Generation block must be disabled (CER[0] = 0) prior to any changes in this value.</p> |

772 . Clock Configuration Register Description

I2S RECEIVER BLOCK FIFO REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | W | RXFFR | 0 | <p>Receiver FIFO Reset. Writing a 1 to this register flushes all the RX FIFOs (this is a self clearing bit).</p> <p>Receiver Block must be disabled prior to writing this bit.</p> |

773 . Receiver Block FIFO Register Description

I2S TRANSMITTER BLOCK FIFO REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|----------------------------|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|--|
| 0 | W | TXFFR | 0 | <p>Transmitter FIFO Reset. Writing a 1 to this register flushes all the TX FIFOs (this is a self clearing bit).</p> <p>Transmitter Block must be disabled prior to writing this bit.</p> |

774 . Transmitter Block FIFO Register Description

I2S LEFT RECEIVE BUFFER REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|--|
| 31:24 | N/A | Reserved | 0 | Reserved and read as zero. |
| 23:0 | R | LRBR | 0x0 | <p>The left stereo data received serially from the receive channel input (sdix) is read through this register.</p> <p>If the RX FIFO is full and the two-stage read operation (for instance, a read from LRBRx followed by a read from RRBRx) is not performed before the start of the next stereo pair, then the new data is lost and an overrun interrupt occurs. (Data already in the RX FIFO is preserved.)</p> <p>NOTE: Before reading this register again, the right stereo data MUST be read from RRBRx, or the status/interrupts will not be valid.</p> |

775 . Left Receive Buffer Register Description

I2S LEFT TRANSMIT HOLDING REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|----------------------------|
| 31:24 | N/A | Reserved | 0 | Reserved and read as zero. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 23:0 | W | LTHR _x | 0x0 | <p>The left stereo data to be transmitted serially through the transmit channel output (sdox) is written through this register.</p> <p>Writing is a two-stage process:</p> <ul style="list-style-type: none"> (1) A write to this register passes the left stereo sample to the transmitter. (2) This MUST be followed by writing the right stereo sample to the RTHR_x register. <p>Data should only be written to the FIFO when it is not full. Any attempt to write to a full FIFO results in that data being lost and an overrun interrupt being generated.</p> |

776 . Left Transmit Holding Register Description

I2S RIGHT RECEIVE BUFFER REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|--|
| 31:24 | N/A | Reserved | 0 | Reserved and read as zero. |
| 23:0 | R | RRBR _x | 0x0 | <p>The right stereo data received serially from the receive channel input (sdix) is read through this register. If the RX FIFO is full and the two-stage read operation (for instance, read from LRBR_x followed by a read from RRBR_x) is not performed before the start of the next stereo pair, then the new data is lost and an overrun interrupt occurs. (Data already in the RX FIFO is preserved.)</p> <p>NOTE: Prior to reading this register, the left stereo data MUST be read from LRBR_x, or the status/interrupts will not be valid.</p> |

777 . Right Receive Buffer Register Description

I2S RIGHT TRANSMIT HOLDING REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|----------------------------|
| 31:24 | N/A | Reserved | 0 | Reserved and read as zero. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 23:0 | W | RTHRx | 0x0 | <p>The right stereo data to be transmitted serially through the transmit channel output (sdox) is written through this register. Writing is a two-stage process:</p> <p>(1) A left stereo sample MUST first be written to the LTHRx register.</p> <p>(2) A write to this register passes the right stereo sample to the transmitter.</p> <p>Data should only be written to the FIFO when it is not full. Any attempt to write to a full FIFO results in that data being lost and an overrun interrupt being generated.</p> |

778 . Right Transmit Holding Register Description

I2S RECEIVE ENABLE REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | R/W | RXCHENx | 1 | <p>Receive channel enable. This bit enables/disables a receive channel, independently of all other channels.</p> <p>On enable, the channel begins receiving on the next left stereo cycle. A global disable of i2s (IER[0] = 0) or the Receiver block (IRER[0] = 0) overrides this value.</p> <p>1: Enable 0: Disable</p> |

779 . Receive Enable Register Description

I2S TRANSMIT ENABLE REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|----------------------------|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|---|
| 0 | R/W | TXCHENx | 1 | <p>Transmit channel enable. This bit enables/disables a transmit channel, independently of all other channels.</p> <p>On enable, the channel begins transmitting on the next left stereo cycle. A global disable of i2s (IER[0] = 0) or Transmitter block (ITER[0] = 0) overrides this value.</p> <p>0: Disable 1: Enable</p> |

780 . Transmit Enable Register Description

I2S RECEIVE CONFIGURATION REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:3 | N/A | Reserved | 0 | Reserved and read as zero. |
| 2:0 | R/W | WLEN | 0x4 | <p>These bits are used to program the desired data resolution of the receiver and enables the LSB of the incoming left (or right) word to be placed in the LSB of the LRBRx (or RRBRx) register.</p> <p>100 implies 24 bit resolution</p> <p>Programmed data resolution must be less than or equal to 24 bits. If the selected resolution is greater than the 24 bits, the receive channel defaults back to 24 bits.</p> <p>The channel must be disabled prior to any changes in this value (RERx[0] = 0).</p> |

781 . Receive Configuration Register Description

I2S TRANSMIT CONFIGURATION REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|----------------------------|
| 31:3 | N/A | Reserved | 0 | Reserved and read as zero. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|--|
| 2:0 | R/W | WLEN | 0x4 | <p>These bits are used to program the data resolution of the transmitter and ensures the MSB of the data is transmitted first.</p> <p>100 implies 24 bit resolution</p> <p>Programmed resolution must be less than or equal to 24 bits. If the selected resolution is greater than 24 bits, the transmit channel defaults back to 24 bits.</p> <p>The channel must be disabled prior to any changes in this value (TERx[0] = 0).</p> |

782 . Transmit Configuration Register Description

I2S INTERRUPT STATUS REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:6 | N/A | Reserved | 0 | Reserved and read as zero. |
| 5 | R | TXFO | 0 | <p>Status of Data Overrun interrupt for the TX channel. Attempt to write to full TX FIFO.</p> <p>0: TX FIFO write valid 1: TX FIFO write overrun</p> |
| 4 | R | TXFE | 1 | <p>Status of Transmit Empty Trigger interrupt. TX FIFO is empty.</p> <p>1: trigger level reached 0: trigger level not reached</p> |
| 3:2 | N/A | Reserved | | Reserved and read as zero. |
| 1 | R | RXFO | 0 | <p>Status of Data Overrun interrupt for the RX channel. Incoming data lost due to a full RX FIFO.</p> <p>0: RX FIFO write valid 1: RX FIFO write overrun</p> |
| 0 | R | RXDA | 0 | <p>Status of Receive Data Available interrupt. RX FIFO data available.</p> <p>1: trigger level reached 0: trigger level not reached</p> |

783 . Interrupt Status Register Description

I2S INTERRUPT MASK REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:6 | N/A | Reserved | 0 | Reserved and read as zero. |
| 5 | R/W | TXFOM | 1 | Masks TX FIFO Overrun interrupt. 1: masks interrupt 0: unmasks interrupt |
| 4 | R/W | TXFEM | 1 | Masks TX FIFO Empty interrupt. 1: masks interrupt 0: unmasks interrupt |
| 3:2 | N/A | Reserved | | Reserved and read as zero. |
| 1 | R/W | RXFOM | 1 | Masks RX FIFO Overrun interrupt. 1: masks interrupt 0: unmasks interrupt |
| 0 | R/W | RXDAM | 1 | Masks RX FIFO Data Available interrupt. 1: masks interrupt 0: unmasks interrupt |

784 . Interrupt Mask Register Description

I2S RECEIVE OVERRUN REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | R | RXCHO | 0 | Read this bit to clear the RX FIFO Data Overrun interrupt. 0: RX FIFO write valid 1: RX FIFO write overrun |

785 . Receive Overrun Register Description

I2S TRANSMIT OVERRUN REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|----------------------------|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| 0 | R | TXCHO | 0 | <p>Read this bit to clear the TX FIFO Data Overrun interrupt.</p> <p>0: TX FIFO write valid 1: TX FIFO write overrun</p> |

786 . Trasmit Overrun Register Description

I2S RECEIVE FIFO CONFIGURATION REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 31:4 | N/A | Reserved | 0 | Reserved and read as zero. |
| 3:0 | R/W | RXCHDT | 0x3 | <p>These bits program the trigger level in the RX FIFO at which the Received Data Available interrupt is generated.</p> <p>Trigger Level = Programmed Value + 1 . Valid RXCHDT values: 0 to 7</p> <p>If an illegal value is programmed, these bits saturate to 7. The channel must be disabled prior to any changes in this value (that is, RERx[0] = 0).</p> |

787 . Receive FIFO Configuration Register Description

I2S TRANSMIT FIFO CONFIGURATION REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| 31:4 | N/A | Reserved | 0 | Reserved and read as zero. |
| 3:0 | R/W | TXCHET | 0x3 | <p>Transmit Channel Empty Trigger. These bits program the trigger level in the TX FIFO at which the Empty Threshold Reached Interrupt is generated.</p> <p>Trigger Level = TXCHET. TXCHET values: 0 to 7</p> <p>If an illegal value is programmed, these bits saturate to 7. The channel must be disabled prior to any changes in this value (that is, TERx[0] = 0).</p> |

788 . Transmit FIFO Configuration Register Description

I2S RECEIVE FIFO FLUSH REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | W | RXCHFR | 0 | Receive Channel FIFO Reset. Writing a 1 to this register flushes an individual RX FIFO. (This is a self clearing bit.) RX channel or block must be disabled prior to writing to this bit. |

789 . Receive FIFO Flush Register Description

I2S TRANSMIT FIFO FLUSH REGISTER x

x is the channel number ; x = 0,1

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:1 | N/A | Reserved | 0 | Reserved and read as zero. |
| 0 | W | TXCHFR | 0 | Transmit Channel FIFO Reset. Writing a 1 to this register flushes channel's TX FIFO. (This is a self clearing bit.) TX channel or block must be disabled prior to writing to this bit. |

790 . Transmit FIFO Flush Register Description

I2S RECEIVER BLOCK DMA REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | R | RXDMA | 0x00 | Receiver Block DMA Register. Used to cycle repeatedly through the enabled receive channels (from lowest numbered to highest), reading stereo data pairs. |

791 . Receiver Block DMA Register Description

I2S RESET RECEIVER BLOCK DMA REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:1 | N/A | Reserved | N/A | Reserved |
| 0 | W | RRXDMA | 0x0 | Reset Receiver Block DMA Register. Writing a 1 to this self-clearing register resets the RXDMA register mid-cycle to point to the lowest enabled Receive channel. Note: Writing to this register has no effect if the component is performing a stereo pair read (such as, when left stereo data has been read but not right stereo data). |

792 . Reset Receiver Block DMA Register Description

I2S TRANSMITTER BLOCK DMA REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | W | TXDMA | 0x00 | Transmitter Block DMA Register. This register can be used to cycle repeatedly through the enabled Transmit channels (from lowest numbered to highest) to allow writing of stereo data pairs. |

793 . Transmitter Block DMA Register Description

I2S RESET TRANSMITTER BLOCK DMA REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:1 | N/A | Reserved | N/A | Reserved |
| 0 | W | RTXDMA | 0x0 | <p>Reset Transmitter Block DMA Register. Writing a 1 to this self-clearing register resets the TXDMA register mid-cycle to point to the lowest enabled Transmit channel.</p> <p>Note:</p> <p>This register has no effect in the middle of a stereo pair write (such as, when left stereo data has been written but not right stereo data).</p> |

794 . Reset Transmitter Block DMA Register Description

I2S COMPONENT PARAMETER 2 REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------|-------------|--|
| 31:6 | N/A | Reserved | | Reserved |
| 5:3 | R | I2S_RX_WORDSIZE_ 1 | 0x3 | <p>Sets the maximum audio data resolution (word size) of the left and right data for Receive Channel 1.</p> <p>0x3 = 24 bit resolution</p> |
| 2:0 | R | I2S_RX_WORDSIZE_ 0 | 0x3 | <p>Sets the maximum audio data resolution (word size) of the left and right data for Receive Channel 0.</p> <p>0x3 = 24 bit resolution</p> |

795 . Component Parameter Register 2 Description

I2S COMPONENT PARAMETER 1 REGISTER

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:22 | N/A | Reserved | | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------|-------------|---|
| 21:19 | R | I2S_TX_WORDSIZE_1 | 0x3 | Sets the maximum audio data resolution (word size) of the left and right data for Transmit Channel 1. 0x3 implies 24 bit resolution |
| 18:16 | R | I2S_TX_WORDSIZE_0 | 0x3 | Sets the maximum audio data resolution (word size) of the left and right data for Transmit Channel 0. 0x3 - 24 bit resolution |
| 15:11 | N/A | Reserved | | Reserved |
| 10:9 | R | I2S_TX_CHANNELS | 0x2 | Controls the number of transmit channels for this i2s component. The range of values is 1 to 2 and is enabled only if parameter I2S_TRANSMITTER_BLOCK is also enabled. 0x2 implies 2 channels are present. |
| 8:7 | R | I2S_RX_CHANNELS | 0x2 | Controls the number of receive channels for this i2s component. The range of values is 1 to 2 and is enabled only if parameter I2S_RECEIVER_BLOCK is also enabled. 0x2 implies 2 channels are present. |
| 6 | R | I2S_RECEIVER_BLOCK | 0x1 | Controls whether the i2s component has I2S receiver block(s) or not. This must be enabled to be able to set the number of RX channels (I2S_RX_CHANNELS). |
| 5 | R | I2S_TRANSMITTER_BLOCK | 0x1 | Controls whether the i2s component has I2S transmitter block(s) or not. This must be enabled to be able to set the number of TX channels (I2S_TX_CHANNELS). |
| 4 | R | I2S_MODE_EN | 0x1 | Determines whether the component acts as the I2S bus master or slave. 0x1 = TRUE |
| 3:2 | R | I2S_FIFO_DEPTH_GLOBAL | 0x2 | 0x2 implies I2S_FIFO_DEPTH_GLOBAL = 8 |
| 1:0 | R | APB_DATA_WIDTH | 0x2 | 32 bit width |

796 . Component Parameter Register 1 Description

I2S COMPONENT VERSION REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------|-------------|--|
| 31:0 | R | I2S_COMP_VERSION | 0x3130392a | Specific I2S component version number. |

797 . I2S Component Version Register Description

I2S COMPONENT TYPE REGISTER

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------|-------------|---|
| 31:0 | R | I2S_COMP_TYPE | 0x445701a0 | Specific I2S component Type.This assigned unique hex value is constant. |

798 .I2S Component Type Register Description

16.11 MCU Configuration Registers

16.11.1 General Description

This block provides programming support for configuration of MCU blocks. Various features in the chip are enabled using this.

16.11.2 Features

Here is a list of features controlled by this block.

- Supports programming of crystal ON/OFF and clock cleaner controls for 40MHz crystal.
- Provides a scratch pad register for software.
- Supports invalid AHB access detection and provides acknowledgment.
- Provides package select information.
- Provides information related to trap detection and supports trap handling for 64K,128K and NWP memories.
- Supports configuration of Ethernet mode.
- Provides information about device and version of the chip.
- Provides registers for programming of USB PHY and reference clock.
- Supports programmable usage of DDR Pads for SPI Flash Controller, CCI and SD memory controller.
- Supports host SPI interrupt handling.
- Provides option to select required host interfaces.
- Indicates host detection to firmware.
- Supports programmable CCI modes(master and slave).
- Provides enable for registering AHB data path between MCU and NWP.
- Provides SDIO slave status to firmware.
- Supports I2S and PCM interface control information.
- Provides enable for registering data path from MCU ROM and NWP RAM.
- Provides enable for Cortex M4 I-port prefetching from NWP memory.
- Provides Cortex M4 control and status information like sleep and deepsleep.
- Provides programmable option to connect MCU HP peripherals to either UDMA or GPDMA.
- Provides enables for memory low power option.
- Provides registers for inter-processor communication between MCU and NWP.
- Provides registers to wakeup NWP from sleep and indicates active/sleep status of MCU and NWP.
- Provides enables for MCU HP peripheral interrupts connected to NWP.
- Supports programming of SPI Flash Controller DLLs in test mode.

16.11.3 Functional Description

- Provides firmware controls to turn ON/OFF the 40 MHz crystal and the respective clock cleaners by using [MCR_XTAL_ON_CTRL](#) register.
- Provides a scratch pad register ([MCR_SW_SCRATCHPAD_SET_REG](#) , [MCR_SW_SCRATCHPAD_CLEAR_REG](#)) for software to be used as storage space. There are set and clear registers.
- Supports invalid AHB access detection using [MCR_AHB_DUMMY_SLAVE_SELECTED_MASTER](#) and [MCR_AHB_ERROR_PER_MASTER_STATUS](#) registers.
- Provides package select information on 10th and 12th bits of [MCR_RST_LATCH_STATUS](#) register.

- Provides information related to trap detection and handling. For more information on this, refer to "Trap Generation and handling" in [Memory Architecture](#).
- Supports configuration of Ethernet via [MCR_ETH_CONFIG](#) register.
- Device and version information of the chip are present in [MCR_CHIP_DEVICE_ID](#) and [MCR_CHIP_VER_NO](#) registers respectively.
- Provides I2S interface control information on 23rd bit of [MCR_GENERIC_CTRL_1_REG](#), 14th bit of [MCR_GENERIC_CTRL_1_REG](#).
- Provides PCM interface control information using [MCR_PCM_CTRL_SET](#)/ [MCR_PCM_CTRL_CLEAR](#) registers.
- USB PHY can be programmed using [MCR_USB_CONFIG_ENABLE_SET_REG_1](#), [MCR_USB_CONFIG_ENABLE_SET_REG_2](#) and respective [CLEAR](#) registers.
- DDR Pads for SPI Flash Controller, CCI and SD memory controller can be programmed using [MCR_DDR_PADS_CTRL](#) (see [table 839](#)) register.
- Supports Host SPI interrupt handling using [MCR_HOST_SPI_INTR_MASK](#), [MCR_HOST_SPI_INTR_SET](#), [MCR_HOST_SPI_INTR_CLR](#) registers.
- Provides option to select required host interfaces using [MCR_HOST_CTRL_REG](#) and [MCR_RST_LATCH_STATUS_REG](#).
- Firmware can read the host detection status from [MCR_RST_LATCH_STATUS](#) register.
- Provides SDIO slave status to firmware via [MCR_SDIO_STATE](#) and [MCR_SDIO_STATE_CTRL](#) registers.
- 7th bit of [MCR_QUASI_SYNC_MODE_REG](#) serves as an enable for Cortex M4 I-port prefetching from NWP memory.
- Programmable option to connect MCU HP peripherals to either UDMA or GPDMA is present in [MCR_PERIPHERAL_UDMA_DMA_SEL](#) register.
- Light sleep for MCU memories and FIFOs can be enabled using [MCR_MEM_LS_ENABLE](#) register. Memory RME and RM can be asserted by using [MCR_MEM_RM_RME](#) register.
- Inter processor communication from MCU to NWP can be done through [MCR_MCU_P2P_INTR_SET](#), [MCR_MCU_P2P_INTR_CLR](#) registers.
- Inter processor communication from NWP to MCU can be done through [MCR_NWP_P2P_INTR_MASK_SET](#), [MCR_NWP_P2P_INTR_MASK_CLR](#), [MCR_NWP_P2P_INTR_CLR](#) registers.
- Active/sleep status of MCU and NWP is present in [MCR_MCU_P2P_COMM_STATUS_REG](#).
- For registers supporting handling of MCU HP peripheral interrupts to NWP refer [MCU HP Peripheral Interrupt Handling](#).
- Supports programming of SPI Flash Controller transmit and receive DLLs in test mode using [MCR_SFC_TX_DLL_TEST](#), [MCR_SFC_RX_DLL_TEST](#) registers respectively.
- Miscellaneous functions provided by [MCR_GENERIC_CTRL](#) register:
 - Enables AHB invalid access trap when 6th bit of the register is set.
 - Enables daisy chaining in JTAG when 10th bit is set.
 - Supports CCI programming on bits 15, 16 and 17.
- Provides enable for registering AHB data path between MCU and NWP using 4th bit of [MCR_AHB_BRIDGE_CTRL](#) register.
- Provides enable for registering data path from MCU ROM and NWP RAM using 4th bit of [MCR_GENERIC_CTRL_1_REG](#) and 6th bit of [MCR_QUASI_SYNC_MODE_REG](#) respectively.
- Cortex M4 can be kept under soft reset using 0th bit of [MCR_CM_CTRL](#) register.
- Sleep and Deepsleep status of MCU can be obtained from [MCR_CM_STATUS](#) register.

16.11.4 Register Summary

Base Address: 0x4600_8000

| Register Name | Offset | Description |
|--|--------|--|
| MCR_HOST_SPI_INTR_MASK_REG | 0x00 | Host SPI interface interrupt mask register |

| Register Name | Offset | Description |
|---|--------|--|
| MCR_HOST_SPI_INTR_SET_REG | 0x04 | Host SPI interface interrupt set register |
| MCR_HOST_SPI_INTR_CLR_REG | 0x08 | Host SPI interface interrupt clear register |
| MCR_HOST_CTRL_REG | 0x0C | Host Control Register |
| MCR_RST_LATCH_STATUS_REG | 0x10 | Reset Latch Status Register |
| MCR_GENERIC_CTRL_REG | 0x14 | MCU Generic Control Register |
| MCR_AHB_BRIDGE_CTRL_REG | 0x18 | AHB bridge Control Register |
| MCR_USB_CONFIG_REG | 0x1C | USB configuration register |
| MCR_SDIO_STATE_CTRL_REG | 0x20 | SDIO State Control Register |
| MCR_SDIO_STATE_REG | 0x24 | SDIO State status Register |
| MCR_XTAL_ON_CTRL_REG | 0x28 | Crystal Control Register |
| MCR_SW_SCRATCHPAD_SET_REG | 0x2C | Software Scratch Pad Set Register |
| MCR_SW_SCRATCHPAD_CLEAR_REG | 0x30 | Software Scratch Pad Clear Register |
| MCR_PCM_CTRL_SET_REG | 0x34 | PCM Interface Control Set Register |
| MCR_PCM_CTRL_CLEAR_REG | 0x38 | PCM Interface Control Clear Register |
| MCR_ETH_CONFIG_REG | 0x40 | Ethernet Mode Configuration Register |
| MCR_GENERIC_CTRL_1_REG | 0x44 | MCU Generic Control Register1 |
| MCR_CM_CTRL_REG | 0x48 | Cortex M4 Control Register |
| MCR_CM_STATUS_REG | 0x4C | Cortex M4 Status Register |
| MCR_CHIP_DEVICE_ID_REG | 0x50 | Chip Device ID Register |
| MCR_CHIP_VER_NO_REG | 0x54 | Chip Version ID Register |
| MCR_PERIPHERAL_UDMA_DMA_SEL_REG | 0x58 | DMA Peripheral Selection Register |
| MCR_AHB_DUMMY_SLAVE_SELECTED_MASTER_REG | 0x5C | AHB Dummy slave selection register |
| MCR_AHB_ERROR_PER_MASTER_STATUS_REG | 0x60 | AHB Master Error status register |
| MCR_I2S_LOOP_BACK_REG | 0x68 | MCU I2S Loopback Enable Register |
| MCR_QUASI_SYNC_MODE_REG | 0x84 | Quasi sync mode register |
| MCR_MEM_LS_ENABLE_REG | 0x8C | Memory Light Sleep Enable Register |
| MCR_USB_CONFIG_ENABLE_SET_REG_1 | 0xF0 | USB PHY Control Set Register1 |
| MCR_USB_CONFIG_ENABLE_CLEAR_REG_1 | 0xF4 | USB PHY Control Clear Register1 |
| MCR_USB_CONFIG_ENABLE_SET_REG_2 | 0xF8 | USB PHY Control Set Register2 |
| MCR_USB_CONFIG_ENABLE_CLEAR_REG_2 | 0xFC | USB PHY Control Clear Register2 |
| MCR_USB_FSEL_FROM_EFUSE_REG | 0x10C | USB Frequency Selection Read from eFuse or flash |
| MCR MCU_P2P_INTR_SET_REG | 0x16C | MCU to NWP P2P Interrupt Set Register |

| Register Name | Offset | Description |
|---|--------|---|
| MCR MCU_P2P_INTR_CLR_REG | 0x170 | MCU to NWP P2P Interrupt Clear Register |
| MCR MCU_P2P_COMM_STATUS_REG | 0x174 | MCU to NWP P2P Communication Status Register |
| MCR_NWP_P2P_INTR_MASK_SET_REG | 0x178 | NWP to MCU P2P Interrupt Mask Register |
| MCR_NWP_P2P_INTR_MASK_CLR_REG | 0x17C | NWP to MCU P2P Interrupt Unmask Register |
| MCR_NWP_P2P_INTR_CLR_REG | 0x180 | NWP to MCU P2P Interrupt Clear Register |
| MCR_PERI_INTR_MASK_SET_TH0_REG | 0x184 | Mask Register for MCU HP Peripheral Interrupts going to NWP on Thread 0 |
| MCR_PERI_INTR_MASK_CLR_TH0_REG | 0x188 | Unmask Register for MCU HP Peripheral Interrupts going to NWP on Thread 0 |
| MCR_PERI_INTR_MASK_SET_TH1_REG | 0x18C | Mask Register for MCU HP Peripheral Interrupts going to NWP on Thread 1 |
| MCR_PERI_INTR_MASK_CLR_TH1_REG | 0x190 | Unmask Register for MCU HP Peripheral Interrupts going to NWP on Thread 1 |
| MCR_PERI_INTR_MASK_SET_TH2_REG | 0x194 | Mask Register for MCU HP Peripheral Interrupts going to NWP on Thread 2 |
| MCR_PERI_INTR_MASK_CLR_TH2_REG | 0x198 | Unmask Register for MCU HP Peripheral Interrupts going to NWP on Thread 2 |
| MCR_PERI_INTR_MASK_SET_TH3_REG | 0x19C | Mask Register for MCU HP Peripheral Interrupts going to NWP on Thread 3 |
| MCR_PERI_INTR_MASK_CLR_TH3_REG | 0x1A0 | Unmask Register for MCU HP Peripheral Interrupts going to NWP on Thread 3 |
| MCR_PERI_INTR_STS_TH0_REG | 0x1A4 | Status Register for MCU HP Peripheral Interrupts going to NWP on Thread 0 |
| MCR_PERI_INTR_STS_TH1_REG | 0x1A8 | Status Register for MCU HP Peripheral Interrupts going to NWP on Thread 1 |
| MCR_PERI_INTR_STS_TH2_REG | 0x1AC | Status Register for MCU HP Peripheral Interrupts going to NWP on Thread 2 |
| MCR_PERI_INTR_STS_TH3_REG | 0x1B0 | Status Register for MCU HP Peripheral Interrupts going to NWP on Thread 3 |
| MCR_DDR_PADS_CTRL_REG (see table 839) | 0x1B4 | DDR Pads Control Register |
| MCR_MEM_RM_RME_REG | 0x1B8 | Memory RM and RME Control Register |
| MCR_SFC_TX_DLL_TEST_REG | 0x1BC | SFC Transmit DLL Test Register |
| MCR_SFC_RX_DLL_TEST_REG | 0x1C0 | SFC Receive DLL Test Register |
| MCR_ULP_AHB_BRIDGE_CLK_ENABLE_REG (see table 843) | 0x1FC | ULP AHB-AHB bridge static clock enable register |

799 . Register Summary

16.11.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, NA = Reserved

MCR_HOST_SPI_INTR_MASK_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------------|-------------|--|
| 31:10 | NA | NA | 0x0 | Reserved and read as zero. |
| 9 | R/W | HOST_SPI_INTR_ACTIV_E_LOW_MODE | 0x0 | Writing '1' to this bit configures the host SPI interrupt in active low mode. By default, it will be active high. |
| 8 | R/W | HOST_SPI_INTR_OPEN_DRAIN_MODE | 0x1 | Writing '1' to this bit configures the host SPI interrupt in open drain mode. When open drain mode is enabled and interrupt is configured in active high mode, external PULL DOWN has to be used on the board. |
| 7:0 | R/W | HOST_SPI_INTR_MSK | 0x0 | Writing '1' in any bit masks the corresponding interrupt in HOST_SPI_INTR_STATUS |

800 .MCR_HOST_SPI_INTR_MASK_REG Description

MCR_HOST_SPI_INTR_SET_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------|-------------|---|
| 31:8 | NA | NA | 0x0 | Reserved and read as zero |
| 7:0 | R/W | HOST_SPI_INTR_STA_TUS | 0x0 | Writing '1' to any bit raises an interrupt to SPI host. Writing '1' at the corresponding bit position in MCR_HOST_SPI_INTR_CLR clears the interrupt. Performing read gives HOST_SPI_INTR_STATUS always. Note: Interrupt will be raised only if the corresponding interrupt is unmasked in the MCR_HOST_SPI_INTR_MASK register. |

801 .MCR_HOST_SPI_INTR_SET_REG Description

MCR_HOST_SPI_INTR_CLR_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------------|-------------|--|
| 31:8 | NA | NA | 0x0 | Reserved and read as zero |
| 7:0 | R/WC | HOST_SPI_INTR_CLE_AR | 0x0 | Writing '1' to this bit clears the MCR_HOST_SPI_INTR_SET_REG . This register gets cleared in the next clock cycle. Performing read gives HOST_SPI_INTR_STATUS always |

802 .MCR_HOST_SPI_INTR_CLR_REG Description

MCR_HOST_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|--|
| 31:16 | NA | NA | 0x0 | Reserved and read as zero |
| 15 | R/W | sdio_spi_prog_sel | 0x1 | SDIO/SPI registers share the same based address. Based on this bit MCU can access either SDIO register or SPI registers 1 - MCU can access SDIO registers 0 - MCU can access SPI registers |
| 14 | R/W | load_host_mode_2 | 0x0 | Overrides the Hardware detected Host on 2nd interface. 0 - H/W based Host detection 1 - S/W based Host (Depends on host_sel_2 mentioned below) |
| 13:12 | R/W | host_sel_2 | 0x0 | Selects the Host on 2nd Interface 0 - No HOST Mode 1 - USB is selected 2, 3 - Reserved |
| 11 | NA | NA | 0x0 | Reserved |
| 10 | R/W | load_host_mode | 0x0 | Overrides the Hardware detected Host on 1st interface. 0 - H/W based Host detection 1 - S/W based Host (Depends on host_sel mentioned below) |
| 9:8 | R/W | host_sel | 0x0 | Selects the Host on 1st Interface 0 - No HOST Mode 1 - Reserved 2 - Host SPI is selected 3 - SDIO slave is selected |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------|-------------|--|
| 7 | R/W | usb_phy_clk_sel | 0x0 | <p>Selects the USB PHY clock between Internal & External source</p> <p>0 - Internal source</p> <p>1 - External source</p> <p>When this is set, clock will be taken from GPIO_25. When this option is selected, USB_XTAL_ON can be used to disable the XO during LP power save. USB_XTAL_ON is mapped to multiple GPIOs.</p> <p>But all these GPIOs will be in GPIO mode at power-up. So we need to keep a weak pull up(~50K-100K) on USB_XTAL_ON so that clock will come at power-up. Because of this pull up there will be leakage during sleep</p> |
| 6:5 | NA | NA | 0x0 | Reserved |
| 4 | R | host_spi_bus_err_in | 0x0 | <p>Host SPI Bus error input.</p> <p>0 means there is no error on Host SPI transmit transaction</p> <p>1 means there is a error on Host SPI transmit transaction</p> |
| 3 | R/W | host_spi_bus_err_out | 0x0 | <p>Host SPI Bus error output.</p> <p>0 means there is no error on Host SPI receive transaction</p> <p>1 means there is error in Host SPI receive transaction</p> |
| 2 | R/W | host_spi_bus_err_oe | 0x1 | <p>Host SPI Bus error output enable. It is active low signal.</p> <p>0 means SPI Bus error output is enabled.</p> <p>1 means SPI Bus error output is disabled.</p> |
| 1 | R/W | host_spi_poweron_rst | 0x0 | This is an active high reset which is used to generate host reset. This has to be enabled only in SPI Host Mode. |
| 0 | R/W | ready_from_core | 0x0 | <p>Indication to the host that bootloading is done. When the reset latch bootload_en is '0', firmware sets this bit.</p> <p>When hardware bootloading is enabled , this gets set only after bootloading is done.</p> <p>When hardware bootloading is enabled and bootloader is programmed to release the PC from soft reset, firmware sets this bit.</p> |

803 .MCR_HOST_CTRL_REG Description

MCR_RST_LATCH_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------------|-------------|--|
| 31:13 | NA | NA | 0x0 | Reserved and read as zero |
| 12 | R | package_sel[1] | NA | This pad gives information about DDR Flash presence. 1. package_sel[1] = 0: DDR Flash is not present 2. package_sel[1] = 1: Check on DDR Flash |
| 11 | R | Reserved | 0x0 | Reserved |
| 10 | R | package_sel[0] | NA | If GPIO 0:6 doesn't have Flash Magic number then check 1. package_sel[0] = 0 ; Check on GPIO 6:11 2. package_sel[0] = 1 ; Check on GPIO 46:51 |
| 9 | R | Reserved | 0x0 | Reserved |
| 8 | R | ulp_wakeup | 0x0 | This bit differentiates between normal power up and ULP wakeup state '1' – ULP based wakeup '0' – Not ULP based wakeup (first power up) |
| 7 | R | mcu_first_powerup_p or | 0x0 | This bit indicates MCU first power up status |
| 6 | R | ram_retention_status | 0x0 | This bit indicates whether RAM is retained or not. '0' – Ram not retained '1' – Ram retained |
| 5 | R | usb_sel | 0x0 | This bit indicates the USB host select '0' – USB not selected '1' – USB selected |
| 4 | R | spi_sel | 0x0 | This bit indicates the SPI slave host select '0' – SPI is not selected '1' – SPI is selected |
| 3 | R | sdio_sel | 0x1 | This bit indicates the SDIO slave host select '0' – SDIO is not selected '1' – SDIO is selected |
| 2:1 | R | boot_mode | 0x3 | Reserved |
| 0 | R | Boot_mode_en | 0x0 | This bit is used to indicate if boot mode is enabled. |

804 .MCR_RST_LATCH_STATUS_REG Description

MCR_GENERIC_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:18 | R | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------------|-------------|--|
| 17 | R/W | Connect_mcu_intr_to_CCI | 0x0 | This bit indicates MCU peripheral interrupt is connection to CCI. 1'b1 - MCU peripheral interrupts are connected to CCI system interrupt 1'b0 - MCU peripheral interrupts are not connected to CCI system interrupt |
| 16 | R/W | CCI_sync_mode | 0x0 | When this bit is set, CCI clock and AHB clock are synchronous with each other in master mode. By default they are asynchronous |
| 15 | R/W | cci_master_en | 0x0 | This bit indicates mode of CCI. This is read from eFuse or flash at the time of bootloading. 1: master mode 0: slave mode |
| 14 | R/W | jtag_daisy_chain_en | 0x0 | This bit has to be set to enable daisy chaining in JTAG |
| 13:10 | R/W | Reserved | 0x0 | Reserved |
| 9 | R/W | hspi_ssi_sel | 0x0 | When this bit is set, it indicates that the host SPI interface pins are intended to be connected to the SSI slave and not to Host SPI. Host SPI chip select is made inactive in this case. '1' – Host SPI pins used for SSI slave. Host SPI inactive. '0' – Host SPI active. |
| 8 | R/W | i2s_ssi_gpio_mode_sel | 0x1 | When set, GPIO 11,12,13 & 14 are used for SSI in GPIO mode 2. '1' – GPIO 11,12,13,14 are used for SSI when configured in GPIO mode 2. '0' - GPIO 11,12,13,14 are used for I2S when configured in GPIO mode 2. |
| 7 | R/W | Reserved | 0x0 | Reserved |
| 6 | R/W | AHB invalid access trap enable | 0x0 | When this bit is set, trap will be generated to processor in the case an invalid access is done on AHB. |
| 5:0 | R/W | Reserved | 0x0 | Reserved |

805 .MCR_GENERIC_CTRL_REG Description

MCR_AHB_BRIDGE_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|-------------|
| 31:5 | R | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------------------------|-------------|---|
| 4 | R/W | Bypass_registering_for_0x0_AHB_bridge | | When this bit is set, bypass the AHB bus registering in AHB bridge, which is present in between MCU and NWP subsystems. It should be asserted when MCU clock is less than 100MHz. |
| 3:0 | R/W | Reserved | 0x0 | Reserved |

806 .MCR_AHB_BRIDGE_CTRL_REG Description

MCR_USB_CONFIG_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------------|-------------|--|
| 31:8 | R/W | Reserved | 0x0 | Reserved |
| 7 | R | utmi_iddig | 0x1 | This signal provides ID pin status 1'b0 => USB acts as Host 1'b1 => USB acts as Device |
| 6 | R | usb_pwrctl_wakeup | 0x1 | This bit will go low, when USB is out of suspend state. |
| 5 | R/W | pwrctl_based_wakeup_en | 0x0 | When set, wakeup is given to sleep FSM whenever there is activity on USB lines. |
| 4 | R/W | utmi_valid | 0x0 | When set valid coming from PHY to controller will be masked and 1 will be passed to controller. |
| 3 | R/W | usb_assert_suspendm | 0x0 | When set USB PHY will enter into suspend state. |
| 2 | R/W | usb_enable_pwr_ctrl | 0x0 | This has to be set before entering into suspend state. This has to be reset after resuming. |
| 1 | R/W | usb_usr_wakeup | 0x0 | Firmware can set this bit to resume USB PHY from suspend state. |
| 0 | R/W | usb_xtal_stable | 0x1 | This has to be set by firmware after xtal(or PLL) clock when PLL clock is used for feeding USB PHY) is stable. |

807 .MCR_USB_CONFIG_REG Description

MCR_SDIO_STATE_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|---|
| 31:2 | NA | Reserved | 0x0 | Reserved |
| 1 | W | Load_sdio_state | 0x0 | When this bit is set, hardware loads the value in MCR_SDIO_STATE register to SDIO block. The state to be restored has to be loaded to MCR_SDIO_STATE register before setting this. This is done by firmware after coming out of sleep. It is used for ULP Mode State Retention. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------|-------------|---|
| 0 | W | latch_sdio_state | 0x0 | When this bit is set, hardware latches the SDIO state into MCR_SDIO_STATE register. This is done by firmware while going to sleep. It is used for ULP Mode State Retention. |

808 . MCR_SDIO_STATE_CTRL_REG Description

MCR_SDIO_STATE_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|---|
| 31:16 | R/W | sdio_state_upper | 0x0 | Upper 16-bits of the SDIO state that is to be loaded into the SDIO block. Upon reading, it represents the upper 16-bits of the state read from SDIO block. |
| 15:0 | R/W | sdio_state_lower | 0x0 | Lower 16-bits of the SDIO state that is to be loaded into the SDIO block. Upon reading, it represents the lower 16-bits of the state read from SDIO block. |

809 . MCR_SDIO_STATE_REG Description

MCR_XTAL_ON_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------|-------------|---|
| 31:4 | R | Reserved | 0x009 | Reserved |
| 3 | R/W | 40Mhz_xtal_on_fw | 0x0 | This bit drives the 40MHz crystal ON indication. This bit is considered only if the firmware based driving is enabled using BIT(0) of this register. '1' – crystal ON is set '0' – crystal ON is reset |
| 2:1 | R/W | Reserved | 0x1 | Reserved |
| 0 | R/W | 40Mhz_xtal_on_fw_s | 0x0 | This bit determines the source of 40MHz CRYSTAL ON indication. '1' – CRYSTAL ON is controlled by firmware through BIT(3) of this register. '0' – CRYSTAL ON is controlled by the sleep state machine and the XTAL_ON_IN coming to the chip. It is asserted whenever there is an indication from either of the clients. |

810 . MCR_XTAL_ON_CTRL_REG Description

MCR_SW_SCRATCHPAD_SET_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:16 | R | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------------|-------------|---|
| 15:0 | R/W | Software_scratchpad_0x0 set | | <p>This register is used by software for storing information. It does not affect anything in the hardware.</p> <p>Writing a '1' to any of the bits sets the corresponding bit in the soft register. Writing '0' has no effect on this register.</p> |

811 .MCR_SW_SCRATCHPAD_SET_REG Description

MCR_SW_SCRATCHPAD_CLEAR_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------------|-------------|---|
| 31:16 | R | Reserved | 0x0 | Reserved |
| 15:0 | R/W | Software_scratchpad_0x0 clear | | <p>This register is used by software for storing information. It does not affect anything in the hardware.</p> <p>Writing a '1' to any of the bits clears the corresponding bit in the soft register. Writing '0' has no effect on this register.</p> |

812 .MCR_SW_SCRATCHPAD_CLEAR_REG Description

MCR_PCM_CTRL_SET_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:4 | R/W | Reserved | 0x0 | Reserved |
| 3:2 | R/W | pcm_bit_res | 0x0 | <p>The bit-resolution of the data on PCM. 2'b00 - 8-bit 2'b01 - 12-bit 2'b1x - 16-bit, 24-bit, 32-bit</p> <p>Writing a '1' to any of the bits sets the corresponding bit in the soft register. Writing '0' has no effect on this register.</p> |
| 1 | R/W | pcm_fsync_start_m | 0x0 | <p>This bit has to be programmed according to when the MS bit of the PCM data is driven w.r.t. the fsync signal of PCM. '1' - The MS bit of data is driven in the same clock cycle as fsync going high. '0' - The MS bit of data is driven one clock cycle after fsync goes high.</p> <p>Writing a '1' to any of the bits sets the corresponding bit in the soft register. Writing '0' has no effect on this register.</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------|-------------|--|
| 0 | R/W | pcm_enable_m | 0x0 | <p>Enable/disable PCM mode of I2S interface. When PCM is enabled, I2S is disabled and vice versa.</p> <p>'1' - PCM mode is enabled and I2S mode is disabled. This programming is valid only when the GPIO signals are programmed for I2S mode.</p> <p>'0' - PCM mode is disabled and I2S mode is enabled. This programming is in addition to the other GPIO level programming to enable I2S mode.</p> <p>Writing a '1' to any of the bits sets the corresponding bit in the soft register. Writing '0' has no effect on this register.</p> |

813 . MCR_PCM_CTRL_SET_REG Description

MCR_PCM_CTRL_CLEAR_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:4 | R/W | Reserved | 0x0 | Reserved |
| 3:2 | R/W | pcm_bit_res | 0x0 | Setting the bits clear the pcm_bit_res. Writing '0' has no effect on this register. |
| 1 | R/W | pcm_fsync_start_m | 0x0 | Setting the bit 1 clear the pcm_fsync_start. Writing '0' has no effect on this register. |
| 0 | R/W | pcm_enable_m | 0x0 | Setting the bit clears the pcm_enable_m. Writing '0' has no effect on this register. |

814 . MCR_PCM_CTRL_CLEAR_REG Description

MCR_ETH_CONFIG_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------------------|-------------|--|
| 31:5 | R/W | Reserved | 0x0 | Reserved |
| 4 | R/W | mac_speed_i | 0x0 | 1 – Ethernet 100mbps operation is selected 0 – Ethernet10mbps operation is selected |
| 3:1 | R/W | eth_phy_intf_sel_i | 0x4 | 3'b100 – RMII Other combinations are reserved |
| 0 | R/W | use_intf_pll_for_eth_r_ef_clk | 0x0 | This bit has to be set to use interface PLL for Ethernet reference clock generation. otherwise, external clock to be used as Ethernet reference clock. |

815 . MCR_ETH_CONFIG_REG Description

MCR_GENERIC_CTRL_1_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------------------|-------------|--|
| 31:24 | R/W | Reserved | 0x0 | Reserved |
| 23 | R | i2s_master_slave_mode | 0x0 | 0 – I2S/I2S PCM act as slave 1 -- I2S/I2S PCM act as master |
| 22 | R/W | Reserved | 0x0 | Reserved |
| 21 | R/W | enable_icache_seq_access_ps_mode | 0x0 | When this bit is set, power save is enabled and a icache read that is sequential to the previous cache read of the same line is saved in local buffer and accessed |
| 20 | R/W | icache_dram_power_save_mode | 0x0 | When this bit is set, only half performance is valid with 1x clock.(Data two cycles after clock). Full performance is valid with 2x clock to icache dram |
| 19 | R/W | Reserved | 0x0 | Reserved |
| 18:14 | R/W | host_pads_gpio_mode | 0x0 | Control bit for 5 pins to use it either as host pin or gpio pin. Pins from 30 to 26 are controlled by these bits respectively. 0 – HOST mode 1 – GPIO mode |
| 13:8 | R | Reserved | 0x0 | Reserved |
| 7 | R | Provide_soc_clk_2x_to_icache_dram | 0x0 | When set, twice the frequency of soc clk will be provided to icache dram., When zero, normal soc_clk is given. |
| 6:5 | R | Reserved | 0x0 | Reserved |
| 4 | R/W | register_rom_output | 0x0 | When set, the ready and read data from ROM will be registered. It should be asserted when MCU clock is greater than 100MHz. |
| 3:0 | R/W | Reserved | 0x0 | Reserved |

816 .MCR_GENERIC_CTRL_1_REG Description

MCR_CM_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------|-------------|---|
| 31:2 | R/W | Reserved | 0x0 | Reserved |
| 1 | R | cm_reset_por | 0x0 | Status of cm_reset_por can be seen on this bit. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|--|
| 0 | R/W | cm_reset | 0x0 | <p>When 1'b1 is written in this bit, both por and non regions of M4 will be under reset.</p> <p>When 1'b0 is written in this bit, por and non regions of M4 will be out of reset.</p> <p>For JTAG and host resets, cm_reset_por will not get reset and only cm_reset gets reset.</p> |

817 .MCR_CM_CTRL_REG Description

MCR_CM_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------|-------------|---|
| 31:3 | R | Reserved | 0x0 | Reserved |
| 2 | R | SLEEPING | 0x0 | When high, indicates the Cortex M4 is in sleeping state |
| 1 | R | SLEEPDEEP | 0x0 | When high, indicates the Cortex M4 is in deep sleep state |
| 0 | R | cm_lockup | 0x0 | When high, indicates the Cortex M4 is in LOCKUP state |

818 .MCR_CM_STATUS_REG Description

MCR_CHIP_DEVICE_ID_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------|-------------|--|
| 31:16 | R | Reserved | 0x0 | Reserved |
| 15:0 | R | device_id | 0x9116 | It gives the device ID of the chip. Device ID is 9116. |

819 .MCR_CHIP_DEVICE_ID_REG Description

MCR_CHIP_VER_NO_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:8 | R | Reserved | 0x0 | Reserved |
| 7:0 | R | ver_no | 0x1 or 0x2 | <p>Indicates the revision number of the chip.</p> <p>0x1 indicates Rev1.0 of the chip.</p> <p>0x2 indicates Rev1.1 of the chip.</p> |

820 .MCR_CHIP_VER_NO_REG Description

MCR_PERIPHERAL_UDMA_DMA_SEL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------|-------------|---|
| 31:8 | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | I2C | 0x0 | When set, ack from udma will go to the I2C, else ack from GPDMA will go. |
| 6 | R/W | Reserved | 0x0 | Reserved |
| 5 | R/W | SSI master | 0x0 | When set, ack from udma will go to the SSI master, else ack from GPDMA will go. |
| 4 | R/W | SSI slave | 0x0 | When set, ack from udma will go to the SSI, else ack from GPDMA will go. |
| 3:2 | R/W | Reserved | 0x0 | Reserved |
| 1 | R/W | uart2 | 0x0 | When set, ack from udma will go to the uart2, else ack from GPDMA will go. |
| 0 | R/W | uart1 | 0x0 | When set, ack from udma will go to the uart1, else ack from GPDMA will go. |

821 .MCR_PERIPHERAL_UDMA_DMA_SEL_REG Description

MCR_AHB_DUMMY_SLAVE_SELECTED_MASTER_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:16 | R/W | Reserved | 0x0 | Reserved |
| 15 | R/W | NWP AHB-AHB master | 0x0 | Hardware set this bit, When the NWP AHB-AHB bridge master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 14 | R/W | USB | 0x0 | Hardware set this bit, When the USB AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 13 | R/W | GPDMA M2 | 0x0 | Hardware set this bit, When the GPDMA AHB master2 is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 12 | R/W | Reserved | 0x0 | Reserved |
| 11 | R/W | CCI M1 | 0x0 | Hardware set this bit, When the CCI AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 10 | R/W | ULP AHB bridge | 0x0 | Hardware set this bit, When the ULP AHB-AHB bridge master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------|-------------|--|
| 9 | R/W | SD memory Controller | 0x0 | Hardware set this bit, When the SD memory controller's AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 8 | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | Ethernet | 0x0 | Hardware set this bit, When the Ethernet AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 6 | R/W | UDMA | 0x0 | Hardware set this bit, When the uDMA AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 5 | R/W | Icache | 0x0 | Hardware set this bit, When the MCU Icache AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 4 | R/W | M4 S port | 0x0 | Hardware set this bit, When the Cortex M4 S-port AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 3 | R/W | M4 D port | 0x0 | Hardware set this bit, When the Cortex M4 D-port AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 2 | R/W | M4 I port | 0x0 | Hardware set this bit, When the Cortex M4 I-port AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 1 | R/W | HIF | 0x0 | Hardware set this bit, When the HIF AHB master is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |
| 0 | R/W | GPDMA M1 | 0x0 | Hardware set this bit, When the GPDMA AHB master1 is trying to access the wrong slave address. Firmware reset this bit by writing zero in this register. |

822 .MCR_AHB_DUMMY_SLAVE_SELECTED_MASTER_REG Description

MCR_AHB_ERROR_PER_MASTER_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:16 | R/W | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------|-------------|--|
| 15 | R/W | NWP AHB-AHB master | 0x0 | Hardware set this bit, When the NWP AHB-AHB bridge master is getting error response. Firmware reset this bit by writing zero in this register. |
| 14 | R/W | USB | 0x0 | Hardware set this bit, When the USB AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 13 | R/W | GPDMA M2 | 0x0 | Hardware set this bit, When the GPDMA AHB master2 is getting error response. Firmware reset this bit by writing zero in this register. |
| 12 | R/W | Reserved | 0x0 | Reserved |
| 11 | R/W | CCI M1 | 0x0 | Hardware set this bit, When the CCI AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 10 | R/W | ULP AHB bridge | 0x0 | Hardware set this bit, When the ULP AHB-AHB bridge master is getting error response. Firmware reset this bit by writing zero in this register. |
| 9 | R/W | SD memory Controller | 0x0 | Hardware set this bit, When the SD memory controller's AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 8 | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | Ethernet | 0x0 | Hardware set this bit, When the Ethernet AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 6 | R/W | UDMA | 0x0 | Hardware set this bit, When the uDMA AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 5 | R/W | Icache | 0x0 | Hardware set this bit, When the MCU Icache AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 4 | R/W | M4 S port | 0x0 | Hardware set this bit, When the Cortex M4 S-port AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 3 | R/W | M4 D port | 0x0 | Hardware set this bit, When the Cortex M4 D-port AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 2 | R/W | M4 I port | 0x0 | Hardware set this bit, When the Cortex M4 I-port AHB master is getting error response. Firmware reset this bit by writing zero in this register. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|--|
| 1 | R/W | HIF | 0x0 | Hardware set this bit, When the HIF AHB master is getting error response. Firmware reset this bit by writing zero in this register. |
| 0 | R/W | GPDMA M1 | 0x0 | Hardware set this bit, When the GPDMA AHB master1 is getting error response. Firmware reset this bit by writing zero in this register. |

823 .MCR_AHB_ERROR_PER_MASTER_STATUS_REG Description

MCR_I2S_LOOP_BACK_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--------------------------------|
| 31:15 | R/W | Reserved | 0x0 | Reserved |
| 14 | R/W | i2s_loop_back_mode | 0x0 | Enables MCU I2S loop back mode |
| 13:8 | R/W | Reserved | 0x0 | Reserved |
| 7:0 | R/W | Reserved | 0x0 | Reserved |

824 .MCR_I2S_LOOP_BACK_REG Description

MCR_QUASI_SYNC_MODE_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------------------|-------------|---|
| 31:8 | R/W | Reserved | 0x0 | Reserved |
| 7 | R/W | Prefetch_Enable_for_M4_I_port | 0x0 | When set, Prefetch operation will be enabled for Cortex M4 I port fetching from NWP memory. |
| 6 | R/W | Register_NWP_RAM_port | 0x0 | When set, the RAM port from MCU to NWP to access NWP memories is registered. It should be asserted when MCU clock is greater than 100MHz. |
| 5:0 | R/W | Reserved | 0x0 | Reserved |

825 .MCR_QUASI_SYNC_MODE_REG Description

MCR_MEM_LS_ENABLE_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------------------------|-------------|--|
| 31:4 | R | Reserved | 0x0 | Reserved |
| 3 | R/W | gen_spi_master_mem_lightsleep_enable | 0x0 | Light sleep enable signal for GSPI master FIFOs. This bit has to be set to make GSPI FIFO memories enter into light sleep. Prior to this, Bit(0) of this register has to be set to enable light sleep for entire MCU. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------------|-------------|--|
| 2:1 | R/W | Reserved | 0x0 | Reserved |
| 0 | R/W | MCU_mem_lightsleep_enable | 0x0 | Light sleep enable signal for all MCU memories. This bit must be set to enable Light sleep mode for any memory in MCU. |

826 .MCR_MEM_LS_ENABLE_REG Description

MCR_USB_CONFIG_ENABLE_SET_REG_1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------------|-------------|---|
| 31 | R/W | usb_testburnin | 0x0 | Connected to TESTBURNIN port of USB PHY |
| 30:29 | R/W | usb_txhsxvtune0 | 0x3 | Connected to TXHSXVTUNE0 port of USB PHY |
| 28:25 | R/W | usb_txvreftune0 | 0x3 | Connected to TXVREFTUNE0 port of USB PHY |
| 24:23 | R/W | usb_txrisetune0 | 0x1 | Connected to TXRISETUNE0 port of USB PHY |
| 22 | R/W | usb_txpreemppulsetu ne0 | 0x0 | Connected to TXPREEMPPULSETUNE0 port of USB PHY |
| 21:20 | R/W | usb_txpreempampstu ne0 | 0x0 | Connected to TXPREEMPAMPSTUNE0 port of USB PHY |
| 19:16 | R/W | usb_txfslstune0 | 0x3 | Connected to TXFSLSTUNE0 port of USB PHY |
| 15:13 | R/W | usb_sqrxtune0 | 0x3 | Connected to SQRXTUNE0 port of USB PHY |
| 12:10 | R/W | usb_otgtune0 | 0x4 | Connected to OTGTUNE0 port of USB PHY |
| 9:7 | R/W | usb_compdistune0 | 0x4 | Connected to COMPDISTUNE0 port of USB PHY |
| 6 | R/W | usb_commononnn | 0x1 | Connected to COMMONONNN port of USB PHY |
| 5 | R/W | usb_sleepm0 | 0x1 | Connected to SLEEPM0 port of USB PHY |
| 4 | R/W | usb_otgdisable0 | 0x0 | Connected to OTGDISABLE0 port of USB PHY |
| 3:2 | R | Reserved | 0x0 | Reserved |
| 1 | R/W | sfc_rx_dll_calib_clk | 0x0 | SFC receive DLL clock calibration enable. It is asserted by firmware in calibration mode for calibrating the input clock. |
| 0 | R/W | usb_phy_soft_reset | 0x0 | Soft Reset used to Reset ATRESET and POR of USB PHY |

827 .MCR_USB_CONFIG_ENABLE_SET_REG_1 Description

MCR_USB_CONFIG_ENABLE_SET_REG_2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:13 | NA | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|--|
| 12:10 | R/W | usb_fsel_fw | 0x4 | <p>Connected to FSEL port of USB PHY if usb_phy_clk_sel is set to zero.</p> <p>These bits specify the frequency of the clock fed to the USB PHY from the internal source, either from the PLLs or RF ref_clk</p> <ul style="list-style-type: none"> • 000 => 9.6 MHz • 001 => 10 MHz • 010 => 12 MHz • 011 => 19.2 MHz • 100 => 20 MHz • 101 => 24 MHz • 111 => 50 MHz |
| 9:8 | R/W | usb_xtal_fw_ctrl | 0x3 | <p>0th bit of usb_xtal_fw_ctrl is used to select the either firmware controlled USB crystal ON or hardware controlled USB crystal ON.</p> <p>1st bit of usb_xtal_fw_ctrl is used as firmware controlled USB crystal ON</p> |
| 7 | R/W | usb_xtal_hw_ctrl | 0x0 | <p>This bit is used to select one of the hardware controller USB crystal ON.</p> <p>1 - Controlled by USB power control wake up, which is asserted after coming out-of suspend</p> <p>0 - Controlled by chip sleep state machine</p> |
| 6 | R/W | usb_txbitstuffen | 0x0 | Connected to TXBITSTUFFEN0 port of USB PHY |
| 5 | R/W | usb_txbitstuffenh | 0x0 | Connected to TXBITSTUFFENH0 port of USB PHY |
| 4 | R/W | usb_portreset0 | 0x0 | Connected to PORTRESET0 port of USB PHY |
| 3 | R/W | usb_vbusvldext0 | 0x1 | Connected to VBUSVLDEXT0 port of USB PHY |
| 2 | R/W | usb_vbusvldeextsel0 | 0x0 | Connected to VBUSVLDEXTSEL0 port of USB PHY |
| 1:0 | R/W | usb_txrestune0 | 0x1 | Connected to TXRESTUNE0 port of USB PHY |

828 . MCR_USB_CONFIG_ENABLE_SET_REG_2 Description

MCR_USB_CONFIG_ENABLE_CLEAR_REG_n

n = 1 to 2; Offset Address: 0xF4 + (n-1)*8

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------|-------------|---|
| 31:0 | R/W | Clear Register | | Clear Register for each of the MCR_USB_CONFIG_ENABLE_SET_REG_n register |

829 . MCR_USB_CONFIG_ENABLE_CLEAR_REG_n Description

MCR_USB_FSEL_FROM_EFUSE_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------|-------------|---|
| 31:3 | R | Reserved | 0x0 | Reserved |
| 2:0 | R/W | usb_fsel_efuse | 0x4 | <p>Connected to FSEL port of USB PHY if usb_phy_clk_sel is set to one.</p> <p>These bits specify the frequency of the clock fed to USB PHY from internal sources like either from the PLLs or RF ref_clk. This value is read from either eFuse or flash at the time of bootloading.</p> <ul style="list-style-type: none"> • 000 => 9.6 MHz • 001 => 10 MHz • 010 => 12 MHz • 011 => 19.2 MHz • 100 => 20 MHz • 101 => 24 MHz • 111 => 50 MHz |

830 . MCR_USB_FSEL_FROM_EFUSE_REG Description

MCR MCU_P2P_INTR_SET_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|--|
| 31:16 | R | Reserved | 0x0 | Reserved for future use. |
| 15:0 | R/W | MCU_P2P_INTR_SET | 0x0 | <p>There are 16 interrupts for P2P(processor to processor) communication from MCU to NWP.</p> <p>Each bit is used to raise the interrupt to NWP</p> <p>For write operation, '1'- Raises the Interrupt '0'- Writing a zero into this has no effect.</p> <p>For read operation, '1' - Interrupt is raised '0' - Not raised</p> |

831 . MCR MCU_P2P_INTR_SET_REG Description

MCR MCU_P2P_INTR_CLR_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|--------------------------|
| 31:16 | R | Reserved | 0x0 | Reserved for future use. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 15:0 | R/W | MCU_P2P_I_NTR_CLR | 0x0 | <p>There are 16 interrupts for P2P(processor to processor) communication from MCU to NWP.</p> <p>Each bit is used to clear the interrupt to NWP</p> <p>For write operation, '1'- Clears the Interrupt '0'- Writing a zero into this has no effect.</p> <p>For read operation, '1' – Interrupt is raised '0' – Not raised</p> |

832 .MCR MCU_P2P_INTR_CLR_REG Description

MCR MCU_P2P_COMM_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:4 | R | Reserved | 0x0 | Reserved. |
| 3 | R | NWP active status | 0x0 | <p>This bit is used to indicate MCU that NWP is active '1'- NWP is active '0'- NWP is sleeping</p> |
| 2 | R | NWP wakeup MCU | 0x0 | <p>This bit is used to indicate MCU that it should wakeup from sleep '1'- NWP wakes up MCU '0'- No operation</p> |
| 1 | R/W | MCU active status | 0x0 | <p>This bit is used to indicate NWP that MCU is active '1'- MCU is active '0'- MCU is sleeping</p> |
| 0 | R/W | MCU wakeup NWP | 0x0 | <p>This bit is used to wakeup NWP from sleep. '1'- MCU wakes up NWP '0'- No operation</p> |

833 .MCR MCU_P2P_COMM_STATUS_REG Description

MCR_NWP_P2P_INTR_MASK_SET_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|--------------------------|
| 31:16 | R | Reserved | 0x0 | Reserved for future use. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 15:0 | R/W | NWP_P2P_INTR_MASK | 0xFFFF | <p>There are 16 interrupts from NWP to MCU for P2P communication.</p> <p>Each bit is used to mask the NWP P2P interrupt</p> <p>For write operation, '1' - masks the NWP P2P Interrupt '0' - Writing a zero into this has no effect.</p> <p>For read operation, '1' - Interrupt is masked '0' - Not raised</p> |

834 . MCR_NWP_P2P_INTR_MASK_SET_REG Description

MCR_NWP_P2P_INTR_MASK_CLR_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|--|
| 31:16 | R | Reserved | 0x0 | Reserved for future use. |
| 15:0 | R/W | NWP_P2P_INTR_UNMASK | 0xFFFF | <p>There are 16 interrupts from NWP to MCU for P2P communication.</p> <p>Each bit is used to unmask the NWP P2P interrupt</p> <p>For write operation, '1' - unmask the Interrupt '0' - Writing a zero into this has no effect.</p> <p>For read operation, '1' - Interrupt is masked '0' - Not raised</p> |

835 . MCR_NWP_P2P_INTR_MASK_CLR_REG Description

MCR_NWP_P2P_INTR_CLR_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|--------------------------|
| 31:16 | R | Reserved | 0x0 | Reserved for future use. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|---|
| 15:0 | R/W | NWP_P2P_I_NTR_CLR | 0x0 | <p>There are 16 interrupts from NWP to MCU for P2P communication.</p> <p>Each bit is used to clear the interrupt</p> <p>For write operation, '1'- Clears the Interrupt for MCU instantly '0'- Writing a zero into this has no effect.</p> <p>For read operation, '1' – Interrupt is raised '0' – Not raised</p> |

836 .MCR_NWP_P2P_INTR_CLR_REG Description

MCR_PERI_INTR_MASK_SET_TH0/1/2/3

Offset Address: (0x184 + (0x8 * n)) ; n = 0,1,2,3; n indicates the thread number

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|---------------|---|
| 31:29 | R | Reserved | 0x0 | Reserved |
| 28:0 | R/W | MCU_PERI_MSK_SET | 0x1FF F_FFF F | <p>There are 29 MCU HP peripheral interrupts connected to NWP. These interrupts are shown in MCU HP Peripheral Interrupts to NWP table.</p> <p>Each bit is used to mask the respective MCU HP peripheral interrupt.</p> <p>For write operation, '1'- Mask Interrupt '0'- Writing a zero into this has no effect.</p> <p>For read operation, '1' – Interrupt masked '0' – Not masked</p> |

837 .MCR_PERI_INTR_MASK_TH0/1/2/3 Description

MCR_PERI_INTR_MASK_CLR_TH0/1/2/3

Offset Address: (0x188 + (0x8 * n)) ; n = 0,1,2,3; n indicates the thread number

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:29 | R | Reserved | 0x0 | Reserved. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------|---------------|---|
| 28:0 | R/W | MCU_PERI_MSK_CLR | 0x1FF F_FFF F | <p>There are 29 MCU HP peripheral interrupts connected to NWP. These interrupts are shown in MCU HP Peripheral Interrupts to NWP table.</p> <p>Each bit is used to unmask the respective MCU HP peripheral interrupt.</p> <p>For write operation, '1'- Unmask Interrupt '0'- Writing a zero into this has no effect.</p> <p>For read operation, '1' - Interrupt masked '0' - Not masked</p> |

838 .MCR_PERI_INTR_MASK_CLR_TH0/1/2/3 Description

MCR_PERI_INTR_STS_TH0/1/2/3

Offset Address: (0x1A4 + (0x8 * n)) ; n = 0,1,2,3; n indicates the thread number

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------|-------------|--|
| 31:29 | R | Reserved | 0x0 | Reserved for future use. |
| 28:0 | R | MCU_PERI_INTR_STATUS | 0x0 | <p>There are 29 MCU HP peripheral interrupts connected to NWP. These interrupts are shown in MCU HP Peripheral Interrupts to NWP table.</p> <p>If bit m of (28:0) is 1, then the mth MCU HP peripheral interrupt is not masked and is been raised in thread n(0/1/2/3).</p> |

839 .MCR_PERI_INTR_STS_TH0/1/2/3 Description

MCR_DDR_PADS_CTRL_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------------------------------|-------------|--|
| 31:7 | R | Reserved | 0x0 | Reserved |
| 6 | R/W | cci_sfc_smih_mux_sel_f or_ddr_pads | 0x0 | <p>Used to mux CCI demuxed pins and SMIH/SPI flash Controller demuxed pins</p> <p>0 – SPI flash controller/SMIH is selected for DDR pads mode. 2nd bit of this register is used to select between SPI flash controller and SMIH.</p> <p>1 – CCI is selected for DDR pads mode.</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|-------------------------------|-------------|--|
| 5 | R/W | cci_ddr_pads_mode | 0x0 | Used to mux CCI signals for DDR pads mode or GPIO pads mode. 1- CCI working in DDR pads mode 0- CCI working in GPIO pads mode |
| 4:3 | R/W | NWP MCU sfc_sel_for_ddr_pads | 0x0 | Mux at soc_modem level to give the ports from MCU and NWP SPI flash controller/SMIH to DDR pads. Below are the combinations for these two bits 00 / 11 – Connect MCU SPI flash controller/SMIH/CCI signals in OCTA mode 01 – Connect NWP SPI signals in OCTA mode 10 – Connect NWP and MCU SPI signals in QUAD mode |
| 2 | R/W | smih_sfc_mux_sel_for_ddr_pads | 0x0 | Used to mux SMIH demuxed pins and SPI flash controller demuxed pins 0 – SPI flash controller is selected 1 – SMIH is selected |
| 1 | R/W | smih_ddr_pads_mode | 0x0 | Used to mux SMIH signals for DDR pads mode or GPIO pads mode. 1- SMIH working in DDR pads mode 0- SMIH working in GPIO pads mode |
| 0 | R/W | sfc_ddr_pads_mode | 0x0 | Used as enable for MCU SPI flash controller ddr pads mode. 1- SPI flash controller working in DDR pads mode 0- SPI flash controller working in GPIO pads mode |

840 . MCR_DDR_PADS_CTRL_REG Description

MCR_MEM_RM_RME_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|---|
| 31:19 | R | Reserved | 0x0 | Reserved |
| 18:16 | R/W | MCU_fifo_rm_rm_e | 0x2 | MCU_fifo_rm_rm[2:1] bits are used as RM ports for fifo memories which are internal to peripherals. MCU_fifo_rm_rm[0] bit is used as RM enable (RME) for fifo memories which are internal to peripherals. |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------------|-------------|---|
| 15:3 | R | Reserved | 0x0 | Reserved |
| 2:0 | R/W | MCU_ram_rom_rm_rme[2:1] | 0x2 | MCU_ram_rom_rm_rme[2:1] bits are used as RM ports for SRAM memories. MCU_ram_rom_rm_rme[0] bit is used as RM enable (RME) for SRAM memories. |

841 .MCR_MEM_RM_RME_REG Description

MCR_SFC_TX_DLL_TEST_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------------------|-------------|---|
| 31:30 | R | Reserved | 0x0 | Reserved |
| 29:23 | R | Tv_polarity_tdc_tx | 0x0 | Read back signal (should come from SPI flash controller TX DLL) This is generated on posedge of sclk, used in estimating average buffer resolution of polarity_tdc in test mode. |
| 22:16 | R | Tv_dqs_tdc_tx | 0x0 | Read back signal (should come from SPI flash controller TX DLL) This is generated on posedge of dqs_in, used in estimating average buffer resolution of dqs_tdc in test mode. |
| 16:9 | R | dqs_mux_sel_tx | 0x0 | Used for SPI flash controller TX DLL dqs mux selection. |
| 8:6 | R | Reserved | 0x0 | Reserved |
| 5 | R/W | sfc_tx_dqs_dll_calib | 0x0 | To start calibration TX SPI flash controller DLL, enable this bit. |
| 4 | R/W | sfc_dqs_dll_loopback_en | 0x0 | SPI flash controller DLL loopback enable. It loopback the TX DLL DQS output to RX DLL DQS input. It is used for DLL test mode. |
| 3:1 | R/W | delay_val_offset_corr_tx | 0x0 | Offset correction applied on delay_val of TX DLL. |
| 0 | R/W | test_mux_sel_polarity_tdc_tx | 0x0 | Test mode signal (to SPI flash controller TX DLL) - used for testing TDC. |

842 .MCR_SFC_TX_DLL_TEST_REG Description

MCR_SFC_RX_DLL_TEST_REG

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|-------------|
| 31 | R | Reserved | 0x0 | Reserved |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------------------|-------------|---|
| 30 | R | ddr_clk_polarity | 0x0 | Indicates SPI flash controller RX DLL clock polarity. |
| 29:23 | R | Tv_polarity_tdc_rx | 0x0 | Read back signal (should come from SPI flash controller RX DLL) This is generated on posedge of sclk, used in estimating average buffer resolution of polarity_tdc in test mode. |
| 22:16 | R | Tv_dqs_tdc_rx | 0x0 | Read back signal (should come from SPI flash controller RX DLL) This is generated on posedge of dqs_in, used in estimating average buffer resolution of dqs_tdc in test mode. |
| 15:9 | R | dqs_mux_sel_rx | 0x0 | This is used for SPI flash controller RX DLL DQS mux selection. |
| 8:4 | R | Reserved | 0x0 | Reserved |
| 3:1 | R/W | delay_val_offset_cor_rx | 0x0 | Offset correction applied on delay_val of RX DLL. Default value of this register should be 4 for functional mode. |
| 0 | R/W | test_mux_sel_polarity_tdc_rx | 0x0 | Test mode signal (to SPI flash controller RX DLL) - used for testing TDC. |

843 . MCR_SFC_RX_DLL_TEST_REG Description

MCR_ULP_AHB_BRIDGE_CLK_ENABLE_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------------------|-------------|---|
| 31:1 | R | Reserved | 0x0 | Reserved |
| 0 | R/W | ULP_AHB_bridge_clk_enable | 0x1 | Used to enable static clock gating ULP AHB-AHB bridge. Only 32-bit write is allowed into this register. |

844 . MCR_ULP_AHB_BRIDGE_CLK_ENABLE_REG Description

16.11.6 MCU HP Peripheral Interrupt Handling

MCU HP Peripheral Interrupts to NWP

| Bit | Peripheral |
|-----|------------------------------------|
| 0 | Reserved |
| 1 | MCU GPDMA Interrupts |
| 2 | Reserved |
| 3 | MCU UDMA Interrupts |
| 4 | MCU Configurable Timers Interrupts |

| Bit | Peripheral |
|------------|--------------------------------|
| 5 | SIO interrupt |
| 6 | USART 1 Interrupt |
| 7 | USART 2 Interrupt |
| 8 | Reserved |
| 9 | I2C Interrupt |
| 10 | SSI Slave Interrupt |
| 11 | Reserved |
| 12 | GSPI Master 1 Interrupt |
| 13 | SSI Master Interrupt |
| 14 | MCPWM Interrupt |
| 15 | Quadrature Encoder Interrupt |
| 16 | GPIO Group Interrupt 0 |
| 17 | GPIO Group Interrupt 1 |
| 18 | GPIO Pin Interrupts |
| 19 | SPI Flash Controller Interrupt |
| 20 | Ethernet Interrupt |
| 21 | Ethernet PMT Interrupt |
| 22 | I2S master Interrupt |
| 23 | Reserved |
| 24 | Can Controller Interrupt |
| 25 | SDMEM Interrupt |
| 26 | PLL Clock Indication Interrupt |
| 27 | Reserved |
| 28 | CCI System Out Interrupt |

Interrupts 0-4 are multi-channel interrupts. They give single interrupt to NWP based on MCU multi-channel interrupt selection registers in [Interrupts](#).

845 . MCU HP Peripheral Interrupts to NWP Description

Programming Sequence

- All interrupts are masked by default on 4 threads.
- To unmask any interrupt for thread 'n', set the corresponding bit in the `MCR_PERI_INTR_MASK_CLR_THn` register.
- To mask the interrupts for thread 'n', set the corresponding bit in the `MCR_PERI_INTR_MASK_SET_THn` register.
- To mask multi-channel interrupts(0-4), use MCU multi-channel interrupt selection registers in [Interrupts](#).

- After clearing the interrupt at source of MCU HP peripheral, a dummy read has to be made to the source from the ISR to make sure that the interrupt is cleared.
- Masked status of all the interrupts are available in [MCR_PERI_INTR_STS_THn](#) for each thread.

16.11.7 Programming Sequence for P2P Interrupt Handling

- Raising interrupt from Cortex M4 to NWP
 - One of the 16 P2P Interrupts can be raised from Cortex M4 by setting the corresponding interrupt bit in [MCR MCU_P2P_INTR_SET_REG](#).
 - P2P Interrupts can be cleared from Cortex M4 by setting the corresponding interrupt bit in [MCR MCU_P2P_INTR_CLR_REG](#).
- Clearing interrupt raised from NWP to Cortex M4
 - One of the 16 P2P Interrupts can be masked and cleared from M4 by setting the corresponding interrupt bit in [MCR_NWP_P2P_INTR_MASK_SET_REG](#) and [MCR_NWP_P2P_INTR_CLR_REG](#) in MCU.
 - Then, unmask the respective interrupt bit in [MCR_NWP_P2P_INTR_MASK_CLR_REG](#).
 - P2P Interrupts can be cleared from M4 by setting the corresponding interrupt bit in [NWP_P2P_INTR_CLR_REG\(0x4105_0090\)](#) in NWP.
 - The interrupt is cleared in two steps to reduce interrupt latency

16.12 Motor Control PWM

16.12.1 General Description

The Motor Control PWM (MCPWM) controller is used to generate a periodic pulse waveform, which is useful in motor control and power control applications. The MCPWM controller acts as a timer to count up to a period count value. The time period and the duty cycle of the pulses are both programmable. It is present as part of the MCU HP peripherals.

16.12.2 Features

- Supports up to eight PWM outputs with four duty cycle generators
- Complementary and Independent output modes are supported
- Dead time insertion in Complementary mode
- Manual override option for PWM output pins. Output pin polarity is programmable
- Supports generation of interrupt for different events
- Supports two hardware fault input pins
- Special event trigger for synchronizing analog-to-digital conversions

16.12.3 Functional Description

The block diagram of the MCPWM controller is shown in figure 1. CSR, 16-bit Time base counter, PWM period generator, dead time generator and PWM override logic are shown in this diagram. This is capable of generating up to four output PWM signals, with the same period but different duty cycles. The Duty cycle generator is followed by the dead time generator. The outputs from this module go through the PWM Override Logic and finally to the output pins. The output pins are grouped in pairs, to facilitate driving the low side and high side of a power half-bridge. The safety fault feature is directly connected to the output stages in order to have the smallest possible response time. A/D Conversion trigger block generates event trigger for synchronizing A/D conversion with time base counter.

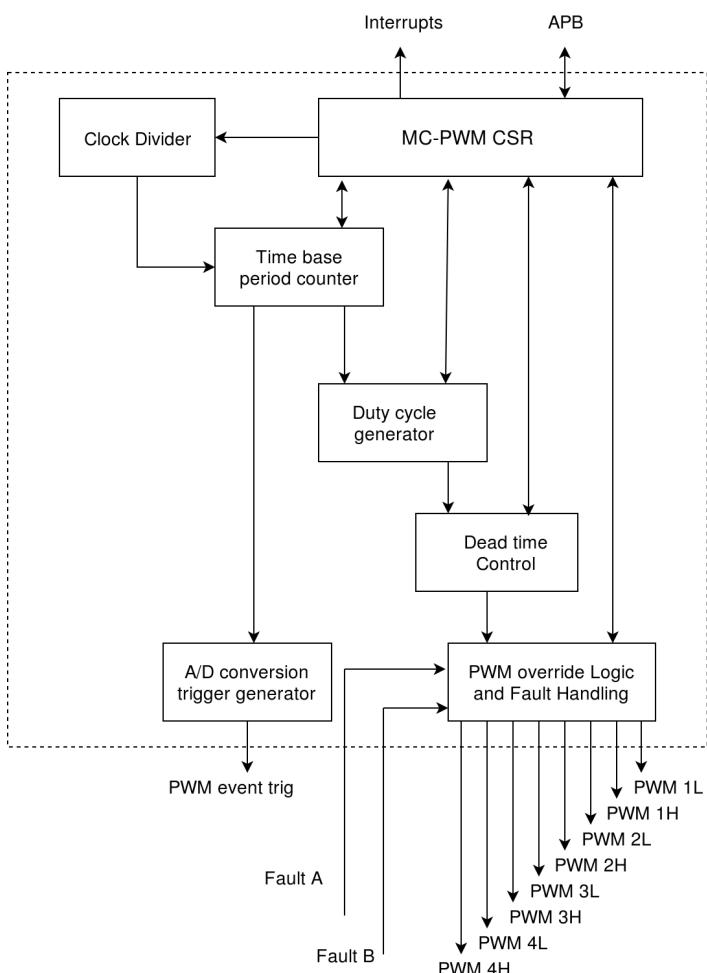
The divided clock is input to the time base period counter. This counter output depending on the time period match, zero match depending on time base mode selection. If a match occurs, a period match signal is generated. The counter direction is controlled by counter direction control bit. If the direction bit value is zero, timer is counting upward, and if it is one, timer is counting downward. The time base postscaler is useful when the PWM duty cycles need not be updated every PWM cycle. The interrupt control logic decides when to generate a PWM interrupt, depending on the postscale value and operation mode. The interrupt generation for each of the operating modes is described below:

Free running mode: An interrupt event is generated when the time base counter is reset to zero due to a match with the time base period register. The postscaler selection bits can be used in free running mode to reduce the frequency of the interrupt events.

Single event mode: An interrupt event is generated when the time base counter is reset to zero due to a match with the time base period register. The PWM time base counter enable bit is also cleared to inhibit further time base counter increments. The postscaler selection bits have no effect in single event mode

Up/Down counting mode: An interrupt event is generated each time the value of the time base counter is equal to zero and the time base counter begins to count upward. The postscale selection can be used to reduce the frequency of interrupt events in up/down counting mode.

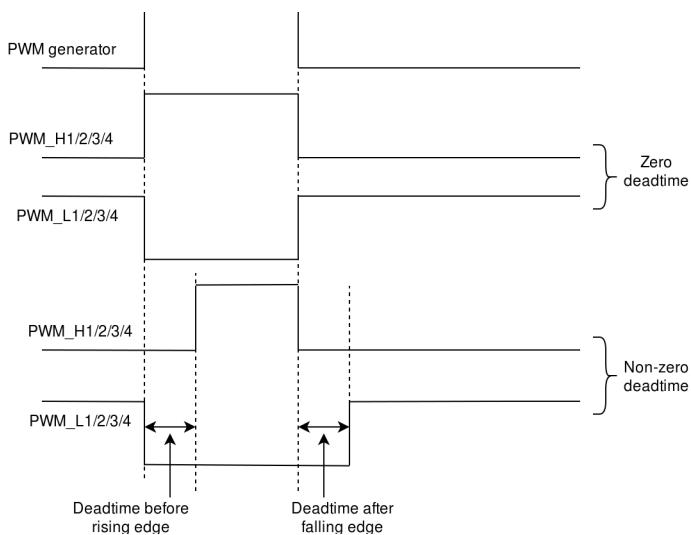
Up/down counting mode with double update of duty cycle: An interrupt event is generated each time the time base counter is equal to zero and each time a period match occurs. The postscale selection has no effect in up/down counting mode with double update of duty cycle. This mode allows the control loop bandwidth to be doubled because the PWM duty cycles can be updated twice per period. Every rising and falling edge of the PWM signal can be controlled using the double update mode.



Motor Control PWM block diagram

The PWM duty cycle generator has four duty cycle generators for eight PWM outputs. It generates duty cycle by comparing duty cycle register value with time base period counter. There are multiple modes for generating duty cycle. They are single event mode, Tail Edge-aligned mode, Lead Edge-aligned mode, Center-aligned mode and Center-aligned mode with double updates. Each complementary output pair from the PWM duty cycle generator has a dead time counter that is used to insert the dead time on both rising and falling edges. Deadtime applies only to PWM output pair , which are in complementary mode. This dead time insertion is shown in figure.2.





Dead time insertion diagram

There is an overdrive control bit for each PWM output pin. If this bit is cleared, the pin will output a PWM signal after dead time insertion. If this bit is set, the pin will be controlled manually. There is a manual out bit for each PWM pin that sets the state of the pin when the output is manually controlled. There are two fault pins, FLT_A and FLT_B, associated with the PWM. When it is asserted, these pins can optionally drive each of the PWM I/O pins to a defined state. Each fault pin is readable using its corresponding status register. Each fault pin has its own interrupt vector, interrupt flag bit, interrupt ack bit and interrupt mask bits. The function of the FLT_A pin is controlled by the fault_A configuration register, and the function of the FLT_B pin is controlled by the fault_B configuration register. The PWM module has a special trigger that allows the A/D converter to be synchronized to the PWM time base. This trigger will start the conversion phase of the A/D block, in order to minimize the delay between getting the A/D conversion result and updating the duty cycle value. The Special Event Compare Register is loaded with a number which is compared to the content of the time base timer, while this one is counting up or down. The counting direction is set by the Special Event Direction bit. As soon as there is a match between the two registers, the trigger is generated and the A/D conversion is started.

16.12.4 Programming sequence

1. Initializing and configuring MCPWM registers.
 - a. Reset deadtime counter for each channel by setting the corresponding bit in `deadtime_disable_frm_reg[11:8]` in `PWM_DEADTIME_CTRL_SET_REG`
 - b. Set deadtime counter for each channel by setting the corresponding bit in `deadtime_disable_frm_reg[11:8]` in `PWM_DEADTIME_CTRL_RESET_REG`.
 - c. Set the deadtime select bits for PWM going inactive by setting `deadtime_select_inactive[7:4]` in `PWM_DEADTIME_CTRL_SET_REG`. 0 implies counter A and 1 implies counter B.
 - d. Set the deadtime select bits for PWM going active by setting `deadtime_select_active[3:0]` in `PWM_DEADTIME_CTRL_SET_REG`. 0 implies counter A and 1 implies counter B.
 - e. If bits in `deadtime_select_active` are set to zero, then program the `deadtime_prescale_select_A` in `PWM_DEADTIME_PRESCALE_SELECT_A` corresponding to each channel and `deadtime_A` for the specific channel.
 - f. If bits in `deadtime_select_active` are set to one, then program the `deadtime_prescale_select_B` in `PWM_DEADTIME_PRESCALE_SELECT_B` corresponding to each channel and `deadtime_B` for the specific channel.
 - g. Reset deadtime counter for each channel by setting the corresponding bit in `deadtime_disable_frm_reg[11:8]` in `PWM_DEADTIME_CTRL_SET_REG`
2. Initializing interrupt status structure.
3. a. Set BIT(0) in `PWM_TIME_PRD_COMMON_REG` to zero to enable individual timers for each channel.
b. Set bits[2:1] in `PWM_TIME_PRD_COMMON_REG` to generate a special event.
c. Set BIT[3] in `PWM_TIME_PRD_COMMON_REG` to enable use of external trigger for base time counter increment or decrement.
d. Set the corresponding bits in `PWM_FLT_OVERRIDE_CTRL_SET_REG[15:12]` for the enable PWM I/O pin pair

is in the complementary output mode for the specific channels.

- e. Set the output polarity for low side and high side signals by setting BIT(3) and BIT(2) in PWM_FLT_OVERRIDE_CTRL_SET_REG
- f. For each channel n (n = 0,1,2,3) , program the pulse width.
- g. To program the pulse width
 - i. Program tmr_operating_mode i.e., bits[2:0] in PWM_TIME_PRD_PARAM_REG_CHn to one for a single event mode. Configure pre_scalar_value and post_scalar_value by programming bits[6:4],[11:8] in PWM_TIME_PRD_PARAM_REG_CHn (n = 0,1,2,3) for each channel.
 - ii. Program the initial value of base timer in PWM_TIME_PRD_CNTR_WR_REG_CH0[15:0] and base timer period in PWM_TIME_PRD_WR_REG_CHn[15:0]
 - h. Configure the dutycycle for each channel by disabling corresponding bit in [7:4] and enabling corresponding bit in [3:0] in PWM_DUTYCYCLE_CTRL_SET_REG and programming duty cycle value in PWM_DUTYCYCLE_REG_WR_VALUE_n
 - i. Enable PWM timer for each channel by setting BIT(1) in PWM_TIME_PRD_CTRL_REG_CH_n.
 - j. Wait till an interrupt is raised by reading BIT(8) , BIT(6), BIT(4), BIT(0) in PWM_INTR_STS.
 - k. Clear all interrupts by setting all the bits in PWM_INTR_ACK

16.12.5 Register Summary

Base Address: 0x4707_0000

| Register Name | Offset | Description |
|--------------------------------|--------|--|
| PWM_INTR_STS | 0x00 | Interrupt Status Register |
| PWM_INTR_UNMASK | 0x04 | Interrupt Unmask Register |
| PWM_INTR_MASK | 0x08 | Interrupt Mask Register |
| PWM_INTR_ACK | 0x0C | Interrupt Acknowledgment Register |
| PWM_TIME_PRD_WR_REG_ch0 | 0x28 | Base timer period register of channel 0 |
| PWM_TIME_PRD_CNTR_WR_REG_ch0 | 0x2C | Base time counter initial value register for channel 0 |
| PWM_TIME_PRD_PARAM_REG_ch0 | 0x30 | Base time period config parameter's register for channel0 |
| PWM_TIME_PRD_CTRL_REG_ch0 | 0x34 | Base time period control register for channel0 |
| PWM_TIME_PRD_STS_REG_ch0 | 0x38 | Base time period status register for channel0 |
| PWM_TIME_PRD_CNTR_VALUE_ch0 | 0x3C | Base Time period counter current value register for channel0 |
| PWM_DUTYCYCLE_CTRL_SET_REG | 0x50 | Duty cycle Control Set Register |
| PWM_DUTYCYCLE_CTRL_RESET_REG | 0x54 | Duty cycle Control Reset Register |
| PWM_DUTYCYCLE_REG_WR_VALUE_0 | 0x58 | Duty cycle Value Register for Channel 0 |
| PWM_DUTYCYCLE_REG_WR_VALUE_1 | 0x5C | Duty cycle Value Register for Channel 1 |
| PWM_DUTYCYCLE_REG_WR_VALUE_2 | 0x60 | Duty cycle Value Register for Channel 2 |
| PWM_DUTYCYCLE_REG_WR_VALUE_3 | 0x64 | Duty cycle Value Register for Channel 3 |
| PWM_DEADTIME_CTRL_SET_REG | 0x78 | Dead time Control Set Register |
| PWM_DEADTIME_CTRL_RESET_REG | 0x7C | Dead time Control Reset Register |
| PWM_DEADTIME_PRESCALE_SELECT_A | 0x80 | Dead time Prescale Select Register for A |
| PWM_DEADTIME_PRESCALE_SELECT_B | 0x84 | Dead time Prescale Select Register for B |

| Register Name | Offset | Description |
|----------------------------------|--------|---|
| PWM_DEADTIME_A_0 | 0x88 | Dead time A for Channel 0 Register |
| PWM_DEADTIME_B_0 | 0x8C | Dead time B for Channel 0 Register |
| PWM_DEADTIME_A_1 | 0x90 | Dead time A for Channel 1 Register |
| PWM_DEADTIME_B_1 | 0x94 | Dead time B for Channel 1 Register |
| PWM_DEADTIME_A_2 | 0x98 | Dead time A for Channel 2 Register |
| PWM_DEADTIME_B_2 | 0x9C | Dead time B for Channel 2 Register |
| PWM_DEADTIME_A_3 | 0xA0 | Dead time A for Channel 3 Register |
| PWM_DEADTIME_B_3 | 0xA4 | Dead time B for Channel 3 Register |
| PWM_OP_OVERRIDE_CTRL_SET_REG | 0xC8 | output override control set register |
| PWM_OP_OVERRIDE_CTRL_RESET_REG | 0xCC | output override control reset register |
| PWM_OP_OVERRIDE_ENABLE_SET_REG | 0xD0 | output override enable set register |
| PWM_OP_OVERRIDE_ENABLE_RESET_REG | 0xD4 | output override enable reset register |
| PWM_OP_OVERRIDE_VALUE_SET_REG | 0xD8 | output override value set register |
| PWM_OP_OVERRIDE_VALUE_RESET_REG | 0xDC | output override enable reset register |
| PWM_FLT_OVERRIDE_CTRL_SET_REG | 0xE0 | fault override control set register |
| PWM_FLT_OVERRIDE_CTRL_RESET_REG | 0xE4 | fault override control reset register |
| PWM_FLT_A_OVERRIDE_VALUE_REG | 0xE8 | fault A override value register |
| PWM_FLT_B_OVERRIDE_VALUE_REG | 0xEC | fault B override value register |
| PWM_SVT_CTRL_SET_REG | 0xF0 | Special event control set register |
| PWM_SVT_CTRL_RESET_REG | 0xF4 | Special event control reset register |
| PWM_SVT_PARAM_REG | 0xF8 | Special event parameter register |
| PWM_SVT_COMPARE_VALUE_REG | 0xFC | Special event compare value register |
| PWM_TIME_PRD_WR_REG_ch1 | 0x100 | Base timer period register of channel1 |
| PWM_TIME_PRD_CNTR_WR_REG_ch1 | 0x104 | Base time counter initial value register for channel1 |
| PWM_TIME_PRD_PARAM_REG_ch1 | 0x108 | Base time period config parameter's register for channel1 |
| PWM_TIME_PRD_CTRL_REG_ch1 | 0x10C | Base time period control register for channel1 |
| PWM_TIME_PRD_STS_REG_ch1 | 0x110 | Base time period status register for channel1 |
| PWM_TIME_PRD_CNTR_VALUE_ch1 | 0x114 | Time period counter current value for channel1 |
| PWM_TIME_PRD_WR_REG_ch2 | 0x118 | Base timer period register of channel2 |
| PWM_TIME_PRD_CNTR_WR_REG_ch2 | 0x11C | Base time counter initial value register for channal2 |

| Register Name | Offset | Description |
|------------------------------|--------|---|
| PWM_TIME_PRD_PARAM_REG_ch2 | 0x120 | Base time period config parameter's register for channel2 |
| PWM_TIME_PRD_CTRL_REG_ch2 | 0x124 | Base time period control register for channel2 |
| PWM_TIME_PRD_STS_REG_ch2 | 0x128 | Base time period status register for channel2 |
| PWM_TIME_PRD_CNTR_VALUE_ch2 | 0x12C | Time period counter current value register for channel2 |
| PWM_TIME_PRD_WR_REG_ch3 | 0x130 | Base timer period register of channel3 |
| PWM_TIME_PRD_CNTR_WR_REG_ch3 | 0x134 | Base time counter initial value register for channel3 |
| PWM_TIME_PRD_PARAM_REG_ch3 | 0x138 | Base time period config parameter's register for channel3 |
| PWM_TIME_PRD_CTRL_REG_ch3 | 0x13C | Base time period control register for channel3 |
| PWM_TIME_PRD_STS_REG_ch3 | 0x140 | Base time period status register for channel3 |
| PWM_TIME_PRD_CNTR_VALUE_ch3 | 0x144 | Time period counter current value register for channel3 |
| PWM_TIME_PRD_COMMON_REG | 0x148 | Time period common register |

846 . Register Summary Table

16.12.6 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, - = Reserved

PWM_INTR_STS

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------------------|-----------|--|
| 15:10 | R | Reserved | 0 | Reserved. |
| 9 | R | pwm_time_prd_match_in0_tr_ch3 | | This time base interrupt for 3rd channel, which considers postscaler value. |
| 8 | R | rise_pwm_time_period_0_match_intr_ch3 | | This time base interrupt for 3rd channel without considering postscaler value. |
| 7 | R | pwm_time_prd_match_in0_tr_ch2 | | This time base interrupt for 2nd channel, which considers postscaler value. |
| 6 | R | rise_pwm_time_period_0_match_intr_ch2 | | This time base interrupt for 2nd channel without considering postscaler value. |
| 5 | R | pwm_time_prd_match_in0_tr_ch1 | | This time base interrupt for 1 st channel, which considers postscaler value. |
| 4 | R | rise_pwm_time_period_0_match_intr_ch1 | | This time base interrupt for 1 st channel without considering postscaler value. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--|-----------|---|
| 3 | R | flt_B_intr | 0 | When the fault B pin is driven low, this interrupt is raised. The PWM outputs remain in this state until the fault B pin is driven high and this interrupt flag has been cleared in software. |
| 2 | R | flt_A_intr | 0 | When the fault A pin is driven low, this interrupt is raised. The PWM outputs remain in this state until the fault A pin is driven high and this interrupt flag has been cleared in software. |
| 1 | R | pwm_time_prd_match_in0 tr_ch0 | | This time base interrupt for 0th channel, which considers postscaler value. The generation of PWM interrupts depends on the mode of operation selected by the PWM Time Base Mode Select bits of the PWM Time Base Control register and. There is no effect of time base output postscaler. |
| 0 | R | rise_pwm_time_period_0 match_intr_ch0 | 0 | This time base interrupt for 0th channel without considering postscaler. The generation of PWM interrupts depends on the mode of operation selected by the PWM Time Base Mode Select bits of the PWM Time Base Control register and the time base output postscaler selected using the PWM Time Base Output Postscale Select bits of the register. |

847 . PWM_INTR_STS Description

PWM_INTR_UNMASK

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------|-----------|------------------|
| 15:0 | R/W | pwm_intr_unmask0 | | Interrupt Unmask |

848 . PWM_INTR_UNMASK Description

PWM_INTR_MASK

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|----------------|
| 15:0 | R/W | pwm_intr_mask0 | | Interrupt Mask |

849 . PWM_INTR_MASK Description

PWM_INTR_ACK

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 15:10 | R/W | Reserved | 0 | reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---|-----------|--|
| 9 | W | pwm_time_prd_match_intr_0 ch3_ack | 0 | 1 – pwm time period match interrupt for 3rd channel will be cleared. 0 – No effect. |
| 8 | W | rise_pwm_time_period_matc0 h_ch3_ack | 0 | 1 – pwm time period match interrupt for 3rd channel will be cleared. 0 – No effect. |
| 7 | W | pwm_time_prd_match_intr_0 ch2_ack | 0 | 1 – pwm time period match interrupt for 2nd channel will be cleared. 0 – No effect. |
| 6 | W | rise_pwm_time_period_matc0 h_ch2_ack | 0 | 1 – pwm time period match interrupt for 2nd channel will be cleared. 0 – No effect. |
| 5 | W | pwm_time_prd_match_intr_0 ch1_ack | 0 | 1 – pwm time period match interrupt for 1st channel will be cleared. 0 – No effect. |
| 4 | W | rise_pwm_time_period_matc0 h_ch1_ack | 0 | 1 – pwm time period match interrupt for 1st channel will be cleared. 0 – No effect. |
| 3 | W | flt_B_intr_ack | 0 | 1 – pwm fault B interrupt will be cleared. 0 – No effect. |
| 2 | W | flt_A_intr_ack | 0 | 1 – pwm fault A interrupt will be cleared. 0 – No effect. |
| 1 | W | pwm_time_prd_match_intr_0 ch0_ack | 0 | 1 – pwm time period match interrupt for 0th channel will be cleared. 0 – No effect. |
| 0 | W | rise_pwm_time_period_matc0 h_ch0_ack | 0 | 1 – pwm time period match interrupt for 0th channel will be cleared. 0 – No effect. |

850 . PWM_INTR_ACK Description

PWM_TIME_PRD_WR_REG_ch0

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_reg_w0 r_value_ch0 | 0 | Value to update the base timer period register of channel 0 |

851 . PWM_TIME_PRD_WR_REG_ch0 Description

PWM_TIME_PRD_CNTR_WR_REG_ch0

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_cntr_0 wr_reg_ch0 | 0 | To update the base time counter initial value for channel 0 |

852 . PWM_TIME_PRD_CNTR_WR_REG_ch0 Description

PWM_TIME_PRD_PARAM_REG_ch0

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------------------------|-----------|---|
| 15:12 | R/W | Reserved | 0 | Reserved |
| 11:8 | R/W | pwm_time_prd_post_scalar_value_ch0 | 0 | Time base output postscale bits for channel0 0000 – 1:1 postscale 0001 – 1:2 1111 – 1:16 |
| 7 | R/W | Reserved | 0 | reserved |
| 6:4 | R/W | Pwm_time_prd_pre_scalar_value_ch0 | 0 | Base timer input clock prescale select value for channel0. 000 - 1x input clock period 001 - 2x input clock period 010 - 4x input clock period 011 - 8x input clock period 100 - 16x input clock period 101 - 32x input clock period 110 - 64x input clock period 111 is reserved |
| 3 | R/W | Reserved | 0 | reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 2:0 | R/W | tmr_operating_mode_ch0 | 0 | Base timer operating mode for channel0. 000 - free running mode 001 - single event mode 010 - down count mode 100 – up/down mode 101 – up/down mode with interrupts for double PWM updates 011,110 & 111 are reserved |

853 . PWM_TIME_PRD_PARAM_REG_ch0 Description

PWM_TIME_PRD_CTRL_REG_ch0

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------------|-----------|--|
| 15:3 | R/W | Reserved | 0 | Reserved |
| 2 | R/W | pwm_sft_RST | 0 | MC PWM soft reset. It is level signal. 1 means soft reset is enabled. 0 means it is out of soft reset. |
| 1 | R/W | pwm_time_base_en_f0 rm_reg_ch0 | 0 | Base timer enable for channel0 1 – timer is enabled 0 – timer is disabled |
| 0 | R/W | pwm_time_prd_cntr_r0 st_frm_reg | 0 | Time period counter soft reset |

854 . PWM_TIME_PRD_CTRL_REG_ch0 Description

PWM_TIME_PRD_STS_REG_ch0

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------------|-----------|--|
| 15:1 | R | Reserved | 0 | reserved |
| 0 | R | pwm_time_prd_dir_st0 s_ch0 | 0 | Time period counter direction status for channel0. 1 – upward 0 - downward |

855 . PWM_TIME_PRD_STS_REG_ch0 Description

PWM_TIME_PRD_CNTR_VALUE_ch0

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------------|-----------|--|
| 15:0 | R | pwm_time_prd_cntr_v 0 alue_ch0 | | Time period counter current value for channel0 |

856 . PWM_TIME_PRD_CNTR_VALUE_ch0 Description

PWM_DUTYCYCLE_CTRL_SET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 15:8 | R/W | Reserved | 0 | Reserved |
| 7:4 | R/W | dutycycle_update_disable | 0 | Duty cycle register updation disable. There is a separate bit for each channel. It is set register only. |
| 3:0 | R/W | imdt_dutycycle_update_en | 15 | Enable to update the duty cycle immediately (without syncing with PWM time base). There is a separate bit for each channel It is set register only. |

857 . PWM_DUTYCYCLE_CTRL_SET_REG Description

PWM_DUTYCYCLE_CTRL_RESET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|---|
| 15:8 | R/W | reserved | 0 | Reserved |
| 7:4 | R/W | dutycycle_update_disable | 0 | Duty cycle register updation disable. There is a separate bit for each channel. |
| 3:0 | R/W | imdt_dutycycle_update_en | 15 | Enable to update the duty cycle immediately (without syncing with PWM time base). There is a separate bit for each channel. |

858 . PWM_DUTYCYCLE_CTRL_RESET_REG Description

PWM_DUTYCYCLE_REG_WR_VALUE_0

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|-------------------------------|
| 15:0 | R/W | pwm_dutycycle_reg_wr_value_ch00 | | Duty cycle value for channel0 |

859 . PWM_DUTYCYCLE_REG_WR_VALUE_0 Description

PWM_DUTYCYCLE_REG_WR_VALUE_1

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|-------------------------------|
| 15:0 | R/W | pwm_dutycycle_reg_wr_value_ch10 | | Duty cycle value for channel1 |

860 . PWM_DUTYCYCLE_REG_WR_VALUE_1 Description

PWM_DUTYCYCLE_REG_WR_VALUE_2

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|-------------------------------|
| 15:0 | R/W | pwm_dutycycle_reg_wr_value_ch20 | | Duty cycle value for channel2 |

861 . PWM_DUTYCYCLE_REG_WR_VALUE_2 Description

PWM_DUTYCYCLE_REG_WR_VALUE_3

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|-------------------------------|
| 15:0 | R/W | pwm_dutycycle_reg_wr_value_ch30 | | Duty cycle value for channel3 |

862 . PWM_DUTYCYCLE_REG_WR_VALUE_3 Description

PWM_DEADTIME_CTRL_SET_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 15:12 | R/W | Reserved | 0 | Reserved |
| 11:8 | R/W | deadtime_disable_frm_reg0 | | Dead time counter soft reset for each channel. It is set register only. |
| 7:4 | R/W | deadtime_select_inactive | 0 | Deadtime select bits for PWM going inactive. 0 means use counter A 1 means use counter B There is a separate bit for each channel |
| 3:0 | R/W | deadtime_select_active | 0 | Deadtime select bits for PWM going active. 0 means use counter A 1 means use counter B There is a separate bit for each channel |

863 . PWM_DEADTIME_CTRL_SET_REG Description

PWM_DEADTIME_CTRL_RESET_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------------------|-----------|--|
| 15:12 | R/W | Reserved | 0 | Reserved |
| 11:8 | R/W | deadtime_disable_frm_reg0 | | Dead time counter soft reset for each channel. |
| 7:4 | R/W | deadtime_select_inactive | 0 | Deadtime select bits for PWM going inactive. 0 means use counter A 1 means use counter B There is a separate bit for each channel |
| 3:0 | R/W | deadtime_select_active | 0 | Deadtime select bits for PWM going active. 0 means use counter A 1 means use counter B There is a separate bit for each channel |

864 . PWM_DEADTIME_CTRL_RESET_REG Description

PWM_DEADTIME_PRESCALE_SELECT_A

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------------|-----------|--|
| 15:8 | R/W | Reserved | 0 | Reserved |
| 7:0 | R/W | deadtime_prescale_select_A | 0 | Dead time prescale selection bits for unit A. Used 2 bits for each channel. 00 means clock period for dead time unit A is 1x input clock period 01 means clock period for dead time unit A is 2x input clock period 10 means clock period for dead time unit A is 4x input clock period 11 means clock period for dead time unit A is 8x input clock period |

865 . PWM_DEADTIME_PRESCALE_SELECT_A Description

PWM_DEADTIME_PRESCALE_SELECT_B

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 15:8 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------|-----------|---|
| 7:0 | R/W | deadtime_prescale_select_B | 0 | <p>Dead time prescale selection bits for unit B.</p> <p>Used 2 bits for each channel.</p> <p>00 means clock period for dead time unit B is 1x input clock period</p> <p>01 means clock period for dead time unit B is 2x input clock period</p> <p>10 means clock period for dead time unit B is 4x input clock period</p> <p>11 means clock period for dead time unit B is 8x input clock period</p> |

866 .PWM_DEADTIME_PRESCALE_SELECT_B Description

PWM_DEADTIME_A_0

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_A_ch0 | 0 | Dead time A value to load into deadtime counter A of channel0 |

867 .PWM_DEADTIME_A_0 Description

PWM_DEADTIME_B_0

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_B_ch0 | 0 | Dead time B value to load into deadtime counter B of channel0 |

868 .PWM_DEADTIME_B_0 Description

PWM_DEADTIME_A_1

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_A_ch1 | 0 | Dead time A value to load into deadtime counter A of channel1 |

869 .PWM_DEADTIME_A_1 Description

PWM_DEADTIME_B_1

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_B_ch1 | 0 | Dead time B value to load into deadtime counter B of channel1 |

870 . PWM_DEADTIME_B_1 Description

PWM_DEADTIME_A_2

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_A_ch2 | 0 | Dead time A value to load into deadtime counter A of channel2 |

871 . PWM_DEADTIME_A_2 Description

PWM_DEADTIME_B_2

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_B_ch2 | 0 | Dead time B value to load into deadtime counter B of channel2 |

872 . PWM_DEADTIME_B_2 Description

PWM_DEADTIME_A_3

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_A_ch3 | 0 | Dead time A value to load into deadtime counter A of channel3 |

873 . PWM_DEADTIME_A_3 Description

PWM_DEADTIME_B_3

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------|-----------|---|
| 15:6 | R/W | Reserved | 0 | Reserved |
| 5:0 | R/W | deadtime_B_ch3 | 0 | Dead time B value to load into deadtime counter B of channel3 |

874 . PWM_DEADTIME_B_3 Description

PWM_OP_OVERRIDE_CTRL_SET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------|-----------|---|
| 15:1 | R/W | Reserved | 0 | Reserved |
| 0 | R/W | op_override_sync | 0 | Output override is synced with pwm time period depending on operating mode. It is set register only. |

875 . PWM_OP_OVERRIDE_CTRL_SET_REG Description

PWM_OP_OVERRIDE_CTRL_RESET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------|-----------|---|
| 15:1 | R/W | Reserved | 0 | Reserved |
| 0 | R/W | op_override_sync | 0 | Output override is synced with pwm time period depending on operating mode. It is reset register only. |

876 . PWM_OP_OVERRIDE_CTRL_RESET_REG Description

PWM_OP_OVERRIDE_ENABLE_SET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------|-----------|--|
| 15:8 | R/W | Reserved | 0 | Reserved |
| 7:0 | R/W | pwm_op_override_enable_reg0 | | pwm output over ride enable. 0 bit for L0 1 bit for L1 2 bit for L2 3 bit for L3 4 bit for H0 5 bit for H1 6 bit for H2 7 bit for H3 It is set register only. |

877 . PWM_OP_OVERRIDE_ENABLE_SET_REG Description

PWM_OP_OVERRIDE_ENABLE_RESET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------|-----------|--|
| 15:8 | R/W | Reserved | 0 | reserved |
| 7:0 | R/W | pwm_op_override_enable_reg0 | | pwm output over ride enable. 0 bit for L0 1 bit for L1 2 bit for L2 3 bit for L3 4 bit for H0 5 bit for H1 6 bit for H2 7 bit for H3 It is reset register only. |

878 . PWM_OP_OVERRIDE_ENABLE_RESET_REG Description

PWM_OP_OVERRIDE_VALUE_SET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------|-----------|---|
| 15:8 | R/W | Reserved | 0 | Reserved |
| 7:0 | R/W | op_override_value0 | | pwm output over ride value. 0 bit for L0 1 bit for L1 2 bit for L2 3 bit for L3 4 bit for H0 5 bit for H1 6 bit for H2 7 bit for H3 It is set register only. |

879 . PWM_OP_OVERRIDE_VALUE_SET_REG Description

PWM_OP_OVERRIDE_VALUE_RESET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 15:8 | R/W | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------|-----------|--|
| 7:0 | R/W | op_override_value | 0 | <p>Pwm output over ride value.</p> <p>0 bit for L0</p> <p>1 bit for L1</p> <p>2 bit for L2</p> <p>3 bit for L3</p> <p>4 bit for H0</p> <p>5 bit for H1</p> <p>6 bit for H2</p> <p>7 bit for H3</p> <p>It is reset register only.</p> |

880 . PWM_OP_OVERRIDE_VALUE_RESET_REG Description

PWM_FLT_OVERRIDE_CTRL_SET_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------|-----------|---|
| 15:12 | R/W | complementary_mode | 0 | <p>PWM I/O pair mode.</p> <p>1 – PWM I/O pin pair is in the complementary output mode</p> <p>0 - PWM I/O pin pair is in the independent output mode</p> <p>Separate enable bit is present for channel</p> <p>It is set register only.</p> |
| 11:8 | R/W | flt_B_enable | 0 | <p>Fault B enable. Separate enable bit is present for channel</p> <p>It is set register only.</p> |
| 7:4 | R/W | Flt_A_enable | 0 | <p>Fault A enable. Separate enable bit is present for channel</p> <p>It is set register only.</p> |
| 3 | R/W | op_polarity_L | 0 | <p>Ouput polarity for low (L3, L2, L1, L0) side signals.</p> <p>0 means active low mode</p> <p>1 means active high mode</p> <p>It is set register only.</p> |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 2 | R/W | op_polarity_H | 0 | Ouput polarity for high (H3, H2, H1, H0) side signals. 0 means active low mode 1 means active high mode It is set register only. |
| 1 | R/W | flt_B_mode | 0 | Fault B mode 1 – cycle by cycle by mode 0 – latched mode It is set register only. |
| 0 | R/W | flt_A_mode | 0 | Fault A mode 1 – cycle by cycle by mode 0 – latched mode It is set register only. |

881 . PWM_FLT_OVERRIDE_CTRL_SET_REG Description

PWM_FLT_OVERRIDE_CTRL_RESET_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|--------------------|------------------|--|
| 15:12 | R/W | complementary_mode | 0 | PWM I/O pair mode. 1 – PWM I/O pin pair is in the complementary output mode 0 - PWM I/O pin pair is in the independent output mode Separate enable bit is present for channel It is reset register only. |
| 11:8 | R/W | flt_B_enable | 0 | Fault B enable. Separate enable bit is present for channel It is reset register only. |
| 7:4 | R/W | Flt_A_enable | 0 | Fault A enable. Separate enable bit is present for channel It is reset register only. |
| 3 | R/W | op_polarity_L | 0 | Output polarity for low (L3, L2, L1, L0) side signals. 0 means active low mode 1 means active high mode It is reset register only. |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 2 | R/W | op_polarity_H | 0 | <p>Output polarity for high (H3, H2, H1, H0) side signals.</p> <p>0 means active low mode</p> <p>1 means active high mode</p> <p>It is reset register only.</p> |
| 1 | R/W | flt_B_mode | 0 | <p>Fault B mode</p> <p>1 – cycle by cycle by mode</p> <p>0 – latched mode</p> <p>It is reset register only.</p> |
| 0 | R/W | flt_A_mode | 0 | <p>Fault A mode</p> <p>1 – cycle by cycle by mode</p> <p>0 – latched mode</p> <p>It is reset register only.</p> |

882 . PWM_FLT_OVERRIDE_CTRL_RESET_REG Description

PWM_FLT_A_OVERRIDE_VALUE_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|------------------------------|------------------|---|
| 15:0 | R/W | Reserved | 0 | Reserved |
| 7:0 | R/W | pwm_flt_A_override_value_reg | | <p>Fault input A PWM override value.</p> <p>1 means PWM output pin is driven active on an external fault input A event.</p> <p>0 means PWM output pin is driven inactive on an external fault input A event.</p> <p>0 bit for L0</p> <p>1 bit for L1</p> <p>2 bit for L2</p> <p>3 bit for L3</p> <p>4 bit for H0</p> <p>5 bit for H1</p> <p>6 bit for H2</p> <p>7 bit for H3</p> <p>Fault A has higher priority than fault B when both are enabled.</p> |

883 . PWM_FLT_A_OVERRIDE_VALUE_REG Description

PWM_FLT_B_OVERRIDE_VALUE_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------|-----------|---|
| 15:8 | R/W | Reserved | 0 | reserved |
| 7:0 | R/W | pwm_flt_B_override_value_reg | | <p>Fault input B PWM override value.</p> <p>1 means PWM output pin is driven active on an external fault input B event.</p> <p>0 means PWM output pin is driven inactive on an external fault input B event.</p> <p>0 bit for L0</p> <p>1 bit for L1</p> <p>2 bit for L2</p> <p>3 bit for L3</p> <p>4 bit for H0</p> <p>5 bit for H1</p> <p>6 bit for H2</p> <p>7 bit for H3</p> <p>Fault A has higher priority than fault B when both are enabled.</p> |

884 . PWM_FLT_B_OVERRIDE_VALUE_REG Description

PWM_SVT_CTRL_SET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------|-----------|--|
| 15:2 | R/W | Reserved | 0 | Reserved |
| 1 | R/W | svt_direction_frm_reg | 0 | <p>Special event trigger for time base direction</p> <p>1 – A special event trigger will occur when PWM time base is counting down</p> <p>0 – A special event trigger will occur when PWM time base is counting up</p> <p>It is set register only.</p> |
| 0 | R/W | svt_enable_frm_reg | 0 | <p>Special event trigger enable. This is used to enable generation special event trigger</p> <p>It is set register only.</p> |

885 . PWM_SVT_CTRL_SET_REG Description

PWM_SVT_CTRL_RESET_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------|-----------|--|
| 15:2 | R/W | Reserved | 0 | Reserved |
| 1 | R/W | svt_direction_frm_reg | 0 | <p>Special event trigger for time base direction</p> <p>1 – A special event trigger will occur when PWM time base is counting down</p> <p>0 – A special event trigger will occur when PWM time base is counting up</p> <p>It is reset register only.</p> |
| 0 | R/W | svt_enable_frm_reg | 0 | <p>Special event trigger enable. This is used to enable generation special event trigger</p> <p>It is reset register only.</p> |

886 . PWM_SVT_CTRL_RESET_REG Description

PWM_SVT_PARAM_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------|-----------|---|
| 15:4 | R/W | Reserved | 0 | Reserved |
| 3:0 | R/W | svt_postsclaler_select | 0 | <p>PWM special event trigger output postscale select bits</p> <p>0000 means 1:1 post scale</p> <p>.....</p> <p>1111 means 1:16 post scale</p> |

887 . PWM_SVT_PARAM_REG Description

PWM_SVT_COMPARE_VALUE_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|--|
| 15:0 | R/W | pwm_svt_compare_val | 0 | Special event compare value. This is used to compare with pwm time period counter to generate special event trigger. |

888 . PWM_SVT_COMPARE_VALUE_REG Description

PWM_TIME_PRD_WR_REG_ch1

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_reg_w0 r_value_ch1 | | Value to update the base timer period register of channel 1 |

889 . PWM_TIME_PRD_WR_REG_ch1 Description

PWM_TIME_PRD_CNTR_WR_REG_ch1

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_cntr_0 wr_reg_ch1 | | To update the base time counter initial value for channel 1 |

890 . PWM_TIME_PRD_CNTR_WR_REG_ch1 Description

PWM_TIME_PRD_PARAM_REG_ch1

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------------------------|-----------|---|
| 15:12 | R/W | Reserved | 0 | Reserved |
| 11:8 | R/W | pwm_time_prd_post_scalar_value_ch1 | 0 | Time base output postscale bits for channel1 0000 – 1:1 postscale 0001 – 1:2 1111 – 1:16 |
| 7 | R/W | Reserved | 0 | reserved |
| 6:4 | R/W | Pwm_time_prd_pre_scalar_value_ch1 | 0 | Base timer input clock prescale select value for channel1. 000 - 1x input clock period 001 - 2x input clock period 010 - 4x input clock period 011 - 8x input clock period 100 - 16x input clock period 101 - 32x input clock period 110 - 64x input clock period 111 is reserved |
| 3 | R/W | Reserved | 0 | reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 2:0 | R/W | tmr_operating_mode_ch1 | 0 | Base timer operating mode for channel1. 000 - free running mode 001 - single event mode 010 - down count mode 100 – up/down mode 101 – up/down mode with interrupts for double PWM updates 011,110 & 111 are reserved |

891 . PWM_TIME_PRD_RARAM_REG_ch1 Description

PWM_TIME_PRD_CTRL_REG_ch1

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|---|
| 15:3 | R/W | Reserved | 0 | Reserved |
| 2 | R/W | pwm_sft_RST | 0 | MC PWM soft reset |
| 1 | R/W | pwm_time_base_en_frm_reg_ch10 | | Base timer enable for channel1 1 – timer is enabled 0 – timer is disabled |
| 0 | R/W | pwm_time_prd_cntr_RST_frm_reg_0 | | Time period counter soft reset |

892 . PWM_TIME_PRD_CTRL_REG_ch1 Description

PWM_TIME_PRD_STS_REG_ch1

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 15:1 | R | Reserved | 0 | reserved |
| 0 | R | pwm_time_prd_dir_sts_ch1 | 0 | Time period counter direction status for channel1. 1 – upward 0 - downward |

893 . PWM_TIME_PRD_STS_REG_ch1 Description

PWM_TIME_PRD_CNTR_VALUE_ch1

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------|-----------|--|
| 15:0 | R | pwm_time_prd_cntr_value_ch1 | 0 | Time period counter current value for channel1 |

894 . PWM_TIME_PRD_CNTR_VALUE_ch1 Description

PWM_TIME_PRD_WR_REG_ch2

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_reg_w0 r_value_ch2 | | Value to update the base timer period register of channel 2 |

895 .PWM_TIME_PRD_WR_REG_ch2 Description

PWM_TIME_PRD_CNTR_WR_REG_ch2

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_cntr_0 wr_reg_ch2 | | To update the base time counter initial value for channel 2 |

896 .PWM_TIME_PRD_CNTR_WR_REG_ch2 Description

PWM_TIME_PRD_PARAM_REG_ch2

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------------------------|-----------|---|
| 15:12 | R/W | Reserved | 0 | Reserved |
| 11:8 | R/W | pwm_time_prd_post_scalar_value_ch2 | 0 | Time base output postscale bits for channel2 0000 – 1:1 postscale 0001 – 1:2 1111 – 1:16 |
| 7 | R/W | Reserved | 0 | reserved |
| 6:4 | R/W | Pwm_time_prd_pre_scalar_value_ch2 | 0 | Base timer input clock prescale select value for channel2. 000 - 1x input clock period 001 - 2x input clock period 010 - 4x input clock period 011 - 8x input clock period 100 - 16x input clock period 101 - 32x input clock period 110 - 64x input clock period 111 is reserved |
| 3 | R/W | Reserved | 0 | reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 2:0 | R/W | tmr_operating_mode_ch2 | 0 | Base timer operating mode for channel2. 000 - free running mode 001 - single event mode 010 - down count mode 100 – up/down mode 101 – up/down mode with interrupts for double PWM updates 011,110 & 111 are reserved |

897 . PWM_TIME_PRD_RARAM_REG_ch2 Description

PWM_TIME_PRD_CTRL_REG_ch2

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|---|
| 15:3 | R/W | Reserved | 0 | Reserved |
| 2 | R/W | pwm_sft_RST | 0 | MC PWM soft reset |
| 1 | R/W | pwm_time_base_en_frm_reg_ch20 | | Base timer enable for channel2 1 – timer is enabled 0 – timer is disabled |
| 0 | R/W | pwm_time_prd_cntr_RST_frm_reg_0 | | Time period counter soft reset |

898 . PWM_TIME_PRD_CTRL_REG_ch2 Description

PWM_TIME_PRD_STS_REG_ch2

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 15:1 | R | Reserved | 0 | Reserved |
| 0 | R | pwm_time_prd_dir_sts_ch2 | 0 | Time period counter direction status for channel2. 1 – upward 0 - downward |

899 . PWM_TIME_PRD_STS_REG_ch2 Description

PWM_TIME_PRD_CNTR_VALUE_ch2

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------|-----------|--|
| 15:0 | R | pwm_time_prd_cntr_value_ch2 | 0 | Time period counter current value for channel2 |

900 . PWM_TIME_PRD_CNTR_VALUE_ch2 Description

PWM_TIME_PRD_WR_REG_ch3

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_reg_w0 r_value_ch3 | | Value to update the base timer period register of channel 3 |

901 . PWM_TIME_PRD_WR_REG_ch3 Description

PWM_TIME_PRD_CNTR_WR_REG_ch3

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------------|-----------|---|
| 15:0 | R/W | pwm_time_prd_cntr_0 wr_reg_ch3 | | To update the base time counter initial value for channel 3 |

902 . PWM_TIME_PRD_CNTR_WR_REG_ch3 Description

PWM_TIME_PRD_PARAM_REG_ch3

| Bit | Access | Function | POR Value | Description |
|-------|--------|------------------------------------|-----------|---|
| 15:12 | R/W | Reserved | 0 | Reserved |
| 11:8 | R/W | pwm_time_prd_post_scalar_value_ch3 | 0 | Time base output postscale bits for channel3 0000 – 1:1 postscale 0001 – 1:2 1111 – 1:16 |
| 7 | R/W | Reserved | 0 | reserved |
| 6:4 | R/W | Pwm_time_prd_pre_scalar_value_ch3 | 0 | Base timer input clock prescale select value for channel3. 000 - 1x input clock period 001 - 2x input clock period 010 - 4x input clock period 011 - 8x input clock period 100 - 16x input clock period 101 - 32x input clock period 110 - 64x input clock period 111 is reserved |
| 3 | R/W | Reserved | 0 | reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 2:0 | R/W | tmr_operating_mode_ch3 | 0 | Base timer operating mode for channel3. 000 - free running mode 001 - single event mode 010 - down count mode 100 – up/down mode 101 – up/down mode with interrupts for double PWM updates 011,110 & 111 are reserved |

903 . PWM_TIME_PRD_RARAM_REG_ch3 Description

PWM_TIME_PRD_CTRL_REG_ch3

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------------|-----------|---|
| 15:3 | R/W | Reserved | 0 | Reserved |
| 2 | R/W | pwm_sft_RST | 0 | MC PWM soft reset |
| 1 | R/W | pwm_time_base_en_frm_reg_ch30 | | Base timer enable for channel3 1 – timer is enabled 0 – timer is disabled |
| 0 | R/W | pwm_time_prd_cntr_RST_frm_reg_0 | | Time period counter soft reset |

904 . PWM_TIME_PRD_CTRL_REG_ch3 Description

PWM_TIME_PRD_STS_REG_ch3

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 15:1 | R | Reserved | 0 | Reserved |
| 0 | R | pwm_time_prd_dir_sts_ch3 | 0 | Time period counter direction status for channel3. 1 – upward 0 - downward |

905 . PWM_TIME_PRD_STS_REG_ch3 Description

PWM_TIME_PRD_CNTR_VALUE_ch3

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------|-----------|--|
| 15:0 | R | pwm_time_prd_cntr_value_ch3 | 0 | Time period counter current value for channel3 |

906 . PWM_TIME_PRD_CNTR_VALUE_ch3 Description

PWM_TIME_PRD_COMMON_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------------|-----------|---|
| 15:3 | R/W | Reserved | 0 | Reserved |
| 3 | R/W | use_ext_timer_trig_frm_0_reg | 0 | Enable to use external trigger for base time counter increment or decrement. |
| 2:1 | R.W | pwm_time_prd_comm_0_on_timer_value | 0 | Base timers select to generate special event trigger. Out of four channel, special event can be generated for one channel (if only 0 th timer is used, bit is not set) |
| 0 | R/W | pwm_time_prd_use_0th_timer_only | 1 | Instead of use four base timers for four channels, use only one base timer for all channels. 0 – one base timer for each channel 1 – only one base timer for all channels |

907 . PWM_TIME_PRD_COMMON_REG Description

16.13 Quadrature Encoder

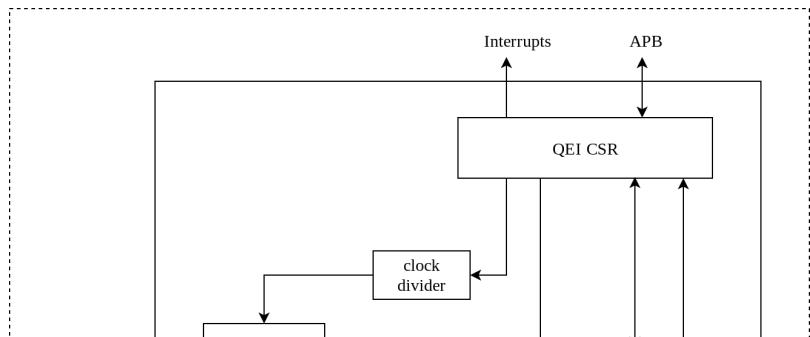
16.13.1 General Description

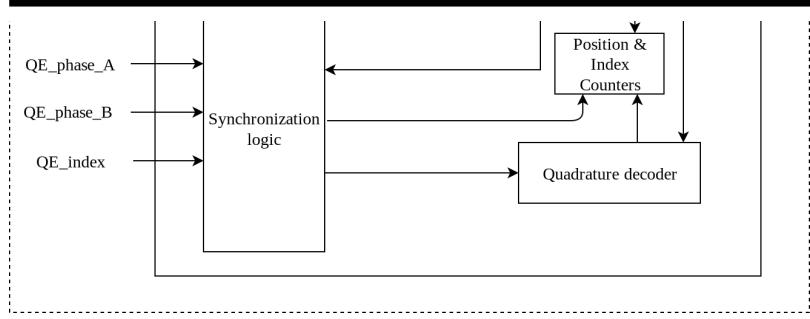
A quadrature encoder (QE), also known as a 2-channel incremental encoder, converts angular displacement into two pulse signals. These two pulses are positioned 90 degrees out of phase. By monitoring both the number of pulses and the relative phase of the two signals, the user code can track the position, direction of rotation, and velocity. In addition, index signal, can be used to reset the position counter. The quadrature encoder decodes the digital pulses from a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, the QE can capture the velocity of the encoder wheel. The QEI is present in MCU HP peripherals.

16.13.2 Features

- Tracks encoder wheel position
- Programmable for 1x, 2x or 4x position counting. Increments/decrements depending on direction
- Index counter for revolution counting
- Velocity capture using built-in timer.
- Supports position counter reset for rollover/underflow or Index pulse
- Position, Index and Velocity compare registers with interrupts
- Supports logically swapping the A and B inputs
- Accepts decoded signal inputs (clock and direction) in timer mode

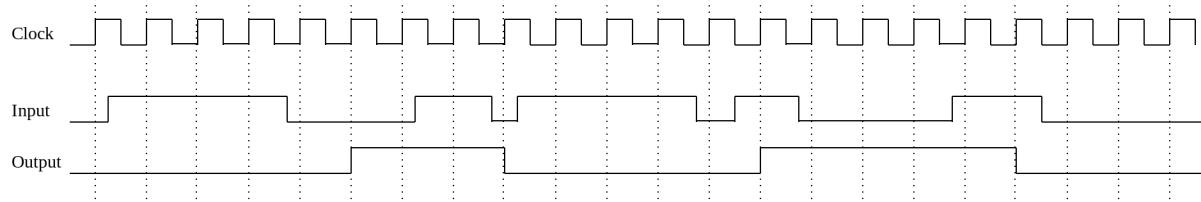
16.13.3 Functional Description





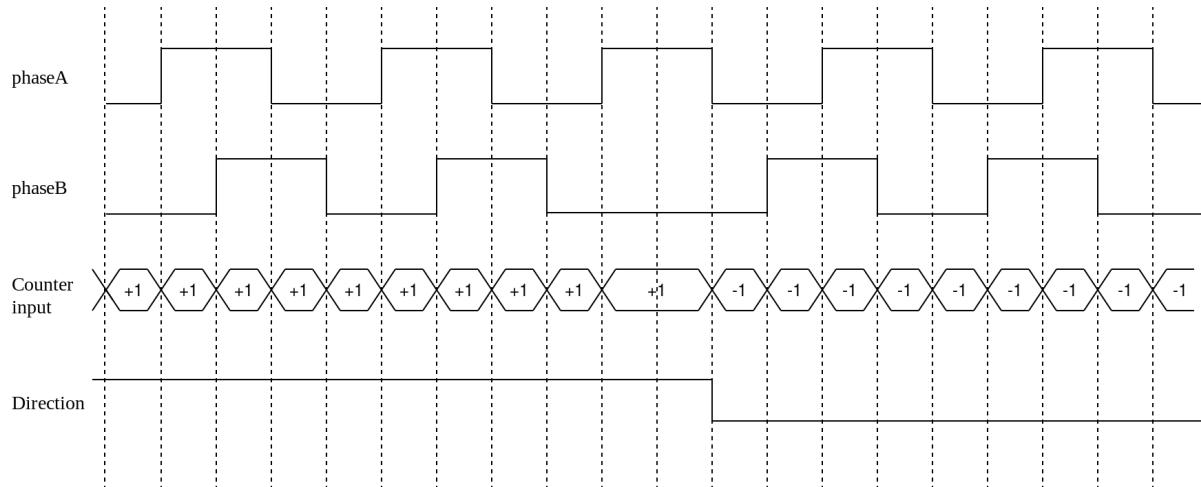
Quadrature Encoder block diagram

synchronization logic is responsible for rejecting noise on the incoming index pulse and quadrature phase signals. This reject low-level noise and short duration noise spikes that typically occur in motor systems. The divided clock, which is programmable, is used to sample the input signals. The synchronization logic output signals can change only after an input level has the same value for three consecutive rising clock edges. The result is that short duration noise spikes between rising clock edges are ignored, and pulses shorter than three clock periods are rejected. It is shown in below figure.2. QE uses inputs directly without filtering also, which is programmable.



Synchronization logic input and output waveforms

Quadrature decoder converts the incoming synchronized signals into count information. This multiplies the resolution of the input signals by a factor of two or four (2x or 4x decoding). When the 4x measurement mode is selected, the QE logic clocks the position counter on both edges of the Phase A and Phase B input signals (i.e increment/decrement the count by 4 for every cycle of phase A/B). Figure 3 illustrates the 4x measurement mode that provides finer resolution data (more position counts) to determine the encoder position.



Quadrature decoder signals in 4x mode

When 2x measurement mode is selected, the QE logic looks only at the rising and falling edge of the Phase A input for the position counter increment rate. The Phase B signal is still used to determine the counter direction. When 1x measurement mode is selected, the QE logic looks only at the rising or falling edge of the Phase A input for the position counter increment/decrement. The Phase B signal is still used to determine the counter direction. When phase A leads phase B, the increment occurs on the rising edge of phase A. When phase B lead phase A, the decrement occurs on the falling edge of phase A.

The 16-bit Up/Down Position Counter counts up or down on every count pulse generated by the quadrature decoder logic. The counter acts as an integrator and its count value is proportional to the position. The direction of

the count is determined by the quadrature decoder. The M4 firmware can examine the contents of the count by reading this register. Generate the interrupt value matches with position max count value. The M4 firmware can also write to the position count register to initialize a count. It also maintains Index counter. There are two modes for resetting position counter. They are reset on rollover/underflow and reset with Index pulse. If the encoder is traveling in the forward direction (position A leads position B), and the value in the position counter register matches the value in the position max count register, position counter resets to '0' on the next occurring quadrature pulse edge that increments position counter. An interrupt event is generated on this rollover event. If the encoder is traveling in the reverse direction (position B leads position A), and the value in the position counter register counts down to '0', the position counter register is loaded with the value in the position max count register on the next occurring quadrature pulse edge that decrements position counter. An interrupt event is generated on this underflow event. The position count can also reset, each time an index pulse is received on the Index pin. If the encoder is traveling in the forward direction (position A leads position B), position counter is reset to '0'. If the encoder is traveling in the reverse direction (position B leads position A), the value in the position max count register is loaded into position counter.

Compute the velocity with following procedure:

- Program the usec timer with operating frequency.
- Program the delta time counter (in usec)
- Set the start velocity counter register bit
- Wait for velocity computation over interrupt (5th bit in interrupt register) or velocity less than interrupt (3rd bit in interrupt register).
- Read the latched velocity counter value for given delta time.

16.13.4 Register Summary

Base Address: 0x4706_0000

| Register Name | Offset | Description |
|--------------------------|--------|--|
| QEI_STATUS_REG | 0x00 | Quadrature Encoder status register |
| QEI_CTRL_SET_REG | 0x04 | Quadrature Encoder control set register |
| QEI_CTRL_RESET_REG | 0x08 | Quadrature Encoder control reset register |
| QEI_CNTL_INIT_REG | 0x0C | Quadrature Encoder initialization register |
| QEI_INDEX_CNT_REG | 0x10 | Quadrature Encoder index counter register |
| QEI_INDEX_MAX_CNT_REG | 0x14 | Quadrature Encoder maximum index counter value register |
| QEI_POSITION_CNT_REG | 0x18 | Quadrature Encoder position counter register |
| QEI_POSITION_MAX_CNT_REG | 0x20 | Quadrature Encoder maximum position counter value register |
| QEI_INTR_STS_REG | 0x28 | Quadrature Encoder interrupt status register |
| QEI_INTR_ACK_REG | 0x2C | Quadrature Encoder interrupt acknowledge register |
| QEI_INTR_MASK_REG | 0x30 | Quadrature Encoder interrupt mask register |
| QEI_INTR_UNMASK_REG | 0x34 | Quadrature Encoder interrupt unmask register |
| QEI_CLK_FREQ_REG | 0x38 | Quadrature Encoder clock frequency register |
| QEI_DELTA_TIME_REG | 0x3C | Quadrature Delta time register |
| QEI_VELOCITY_REG | 0x44 | Quadrature velocity register |
| QEI_POSITION_MATCH_REG | 0x4C | Quadrature position match register |

908 . Register Summary Table

16.13.5 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, - = Reserved, *- dynamic value

QEI_STATUS_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------|-----------|---|
| 31:5 | R | Reserved | 0 | Reserved. |
| 4 | R | Position_cntr_direction | 0 | Position Counter Direction Status bit 1 = Position counter direction is positive (+) 0 = Position counter direction is negative (-) |
| 3 | R | Position_cntr_err | 0 | Count Error Status Flag bit 1 = Position count error has occurred 0 = Position count error has not occurred |
| 2 | R | Qei_position_A | 0* | This is a direct value from the position signal generator (Ex: motor wheel). If filter bypass is enable the direct value from the generator can be seen in this bit, if not filtered version of value from the generator can be seen. Value refers to the signal Position_A from the generator. * Indicates dynamic value. |
| 1 | R | Qei_position_B | 0* | This is a direct value from the position signal generator (Ex: motor wheel). If filter bypass is enable the direct value from the generator can be seen in this bit, if not filtered version of value from the generator can be seen. Value refers to the signal Position_B from the generator. * Indicates dynamic value. |
| 0 | R | Qei_index | 0* | This is a direct value from the position signal generator (Ex: motor wheel). If filter bypass is enable the direct value from the generator can be seen in this bit, if not filtered version of value from the generator can be seen. Value refers to the signal Index from the generator. * Indicates dynamic value. |

909 . QEI_STATUS_REG Register Description

QEI_CTRL_SET_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------|-----------|---|
| 31:16 | R | Reserved | 0 | - |
| 15 | R/W | Index_cnt_RST | 0 | 1 - index counter is going to reset. |
| 14 | R/W | Pos_cnt_RST | 0 | 1 - position counter is going to reset. |
| 13 | R/W | Qei_pos_cnt_16_bit_mode | 0 | Qei position counter 16 bit mode enable. While writing : 1 - QEI position status counter will be working as a 16 bit counter. 0 - No effect. QEI position status counter is working as 32 bit counter While reading : 1 - QEI position status counter is working as 16 bit counter. 0 - QEI position status counter is working as 32 bit counter. |
| 12 | R/W | Qei_stop_in_idle | 0 | 1 means stop qei ctrl in idle state. 0 means continue from idle state. |
| 11 | R/W | Start_velocity_cntr | 0* | Starting the velocity counter. It is self reset bit. |
| 10 | R/W | Timer_mode | 0 | 1 - timer mode. In this mode, decoded timer pulse and direction are taken from position A and position B pins respectively. 0 - Quadrature encoder mode. |
| 9 | R/W | Digital_filter_bypass | 0 | 1 - digital filter is bypassed for all input signals (position A, position B and Index) 0 - digital filter is in-path for all input signals |
| 8 | R/W | Index_cnt_RST_en | 0 | 1 - index counter is going to reset after reaching max count, which is mentioned in qei_index_max_cnt register. |
| 7 | R/W | reserved | 0 | reserved |
| 6 | R/W | reserved | 0 | reserved |
| 5 | R/W | Pos_cnt_dir_frm_reg | 0 | Position Counter Direction indication from user 1 - Position counter direction is positive (+) 0 - Position counter direction is negative (-) |
| 4 | R/W | Pos_cnt_direction_ctrl | 0 | 0 - position B pin defines the direction of position counter. 1 - pos_cnt_dir_frm_reg defines the position counter direction. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------------|-----------|---|
| 3 | R/W | reserved | 0 | reserved |
| 2 | R/W | Pos_cnt_RST_with_index_en | 0 | 1 - position counter is getting reset for every index pulse 0 - position counter is getting reset after reaching max count, which is mentioned in position_max_cnt |
| 1 | R/W | Qei_swap_phase_AB | 0 | Phase A and Phase B Input Swap Select bit 1 - Phase A and Phase B inputs are swapped 0 - Phase A and Phase B inputs are not swapped |
| 0 | R | Qei_sft_RST | 0* | Quadrature encoder soft reset. It is self reset signal. |

910 . QEI_CTRL_SET Register Description

QEI_CTRL_RESET_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------|-----------|--|
| 31:16 | R | Reserved | 0 | - |
| 15 | R/W | Index_CNT_RST | 0 | 1 - index counter is going to reset. |
| 14 | R/W | Pos_CNT_RST | 0 | 1 - position counter is going to reset. |
| 13 | R/W | Qei_pos_CNT_16_bit_mode | 0 | Qei position counter 16 bit mode enable. While writing : 1 – QEI position status counter will be working as a 32 bit counter. 0 – No effect. While reading : 1 – QEI position status counter is working as 16 bit counter. 0 – QEI position status counter is working as 32 bit counter. |
| 12 | R/W | Qei_stop_in_idle | 0 | 1 means stop qei ctrl in idle state. 0 means continue from idle state. |
| 11 | R/W | Start_velocity_CNT_r | 0 | Starting the velocity counter. It is self reset bit. |
| 10 | R/W | Timer_mode | 0 | 1 - timer mode. In this mode, decoded timer pulse and direction are taken from position A and position B pins respectively. 0 - Quadrature encoder mode. |
| 9 | R/W | Digital_filter_bypass | 0 | 1 - digital filter is bypassed for all input signals (position A, position B and Index) 0 - digital filter is in-path for all input signals |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------------|-----------|---|
| 8 | R/W | Index_cnt_RST_en | 0 | 1 - index counter is going to reset after reaching max count, which is mentioned in qei_index_max_cnt register. |
| 7 | R/W | Reserved | 0 | Reserved |
| 6 | R/W | Reserved | 0 | Reserved |
| 5 | R/W | Pos_cnt_dir_frm_rg | 0 | Position Counter Direction indication from user 1 - Position counter direction is positive (+) 0 - Position counter direction is negative (-) |
| 4 | R/W | Pos_cnt_direction_ctrl | 0 | 0 - position B pin defines the direction of position counter. 1 - pos_cnt_dir_frm_reg defines the position counter direction. |
| 3 | R/W | Reserved | 0 | Reserved |
| 2 | R/W | Pos_cnt_RST_with_Index_en | 0 | 1 - position counter is getting reset for every index pulse 0 - position counter is getting reset after reaching max count, which is mentioned in position_max_cnt |
| 1 | R/W | Qei_swap_phase_AB | 0 | Phase A and Phase B Input Swap Select bit 1 - Phase A and Phase B inputs are swapped 0 - Phase A and Phase B inputs are not swapped |
| 0 | R | Qei_sft_RST | 0* | Quadrature encoder soft reset. It is self reset signal. |

911 . QEI_CTRL_RESET Register Description

QEI_CNTL_INIT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------|-----------|--|
| 31:13 | R/W | Reserved | 0 | Reserved |
| 12 | R/W | Index_cnt_init | 0 | Index counter initial value in unidirectional index enable mode. |
| 11 | R/W | Unidirectional_Index | 0 | Uni directional index enable. 1 means direction change in position counter resets index counter |
| 10 | R/W | Unidirectional_Velocity | 0 | Uni directional velocity enable. 1 means direction change in position counter resets velocity counter |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------|-----------|--|
| 9:6 | R/W | Df_clk_divide_slt | 0 | Digital Filter Clock Divide Select bits 1010 = 1:1024 Clock divide for Index, position A & B 1001 = 1:512 Clock divide for Index, position A & B 1000 = 1:256 Clock divide for Index, position A & B 0111 = 1:128 Clock divide for Index, position A & B 0110 = 1:64 Clock divide for Index, position A & B 0101 = 1:32 Clock divide for Index, position A & B 0100 = 1:16 Clock divide for Index, position A & B 0011 = 1:8 Clock divide for Index, position A & B 0010 = 1:4 Clock divide for Index, position A & B 0001 = 1:2 Clock divide for Index, position A & B 0000 = 1:1 Clock divide for Index, position A & B |
| 5:4 | R/W | Index_match_value | 0 | These bits allow user to specify the state of position A & B during index pulse generation. |
| 3:2 | R/W | Reserved | 0 | Reserved |
| 1:0 | R/W | Qei_encoding_mode | 0 | 00 = 1x mode 01 = 2x mode 10 = 4x mode |

912 . QEI_CNTL INIT Register Description

QEI_INDEX_CNT_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------|-----------|---|
| 31:16 | R | Reserved | 0 | - |
| 15:0 | R/W | Qei_index_cnt | 0* | Index counter value. User can initialize/change the index counter using this register. When read, this provides the current index counter value. Note : This value should be less than or equal to the value in QEI_INDEX_MAX_CNT register |

913 . QEI_INDEX_CNT Register Description

QEI_INDEX_MAX_CNT_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------|-------------|---|
| 31:0 | R/W | Qei_index_max_cnt | 0x0000_FFFF | <p>Qei index maximum count.</p> <p>This is a maximum count value that is allowed to increment in the index counter. If index counter reaches this value, will get reset to zero.</p> <p>Index_cnt_RST_en, qei_ctrl_reg[8] should be set (1) for resetting the index counter when reaches max value.</p> <p>In 16-bit mode, only the lower 16 bits are used.</p> |

914 . QEI_INDEX_MAX_CNT Register Description

QEI_POSITION_CNT_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------|-----------|---|
| 31:0 | R/W | Qei_position_cnt_wr_value | 0 | <p>This is used to program/change the value of position counter status.</p> <p>In 16-bit mode, only the lower 16 bits are used.</p> |

915 . QEI_POSITION_CNT Register Description

QEI_POSITION_MAX_CNT_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-------------|---|
| 31:0 | R/W | Qei_position_max_cnt | 0x0000_FFFF | <p>This is a maximum count value that is allowed to increment in the position counter.</p> <p>If position counter reaches this value in positive direction, will get set to zero and if reaches zero in negative direction, will get set to this max count.</p> <p>Pos_cnt_RST_with_Index_en, qei_ctrl_reg[2] should be reset (0) for setting the position counter when reaches max/zero value.</p> <p>In 16-bit mode, only the lower 16 bits are used.</p> |

916 . QEI_POSITION_MAX_CNT Register Description

QEI_INTR_STS_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:6 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--|-----------|--|
| 5 | R | Qei_velocity_computation_over_intr_lev | 0 | When velocity count is computed for given delta time, than interrupt is raised. |
| 4 | R | Qei_position_cnt_match_i0_ntr_lev | 0 | This is raised when the position counter reaches position match value, which is programmable. |
| 3 | R | Velocity_less_than_intr_le0_v | 0 | When velocity count is less than the value given in velocity_value_to_compare register, interrupt is raised. |
| 2 | R | Position_cntr_err_intr_lev0 | 0 | Whenever number of possible positions are mismatched with actual positions are received between two index pulses this will raised. |
| 1 | R | Qei_index_cnt_match_int0_r_lev | 0 | This is raised when index counter reaches max value loaded in to index_max_cnt register. |
| 0 | R | Qei_position_cnt_reset_in0_tr_lev | 0 | This is raised when the position counter reaches it's extremes. In position direction position_max_cnt is the extreme value and in negative direction zero is the extreme value. |

917 . QEI_INTR_STS Register Description

QEI_INTR_ACK_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------------------|-----------|--|
| 31:6 | R/W | Reserved | 0 | Reserved |
| 5 | R/W | Velocity_computation_over_in0_tr_lev | 0 | Velocity_computation_over_intr_ack If you write 1 – Velocity computation is over intr will be cleared. 0 – No effect. Zero when read |
| 4 | R/W | Qei_position_cnt_match_intr_0_lev | 0 | Qei_position_cnt_match_intr_ack If you write 1 – Qei position cnt match intr will be cleared. 0 – No effect. Zero when read |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------------|-----------|---|
| 3 | R/W | Velocity_less_than_intr_lev | 0 | Velocity_less_than_intr_ack If you write 1 – Velocity less than intr will be cleared. 0 – No effect. Zero when read |
| 2 | R/W | Position_cntr_err_intr_lev | 0 | Position_cntr_err_intr_ack If you write 1 – Position cntr err intr will be cleared. 0 – No effect. Zero when read |
| 1 | R/W | Qei_index_cnt_match_intr_lev | 0 | Position_cntr_err_intr_ack If you write 1 – Qei index cnt match intr will be cleared. 0 – No effect. Zero when read |
| 0 | R/W | Qei_position_cnt_reset_intr_le0v | 0 | Qei_position_cnt_reset_intr_ack If you write 1 – Qei position cnt reset intr will be cleared. 0 – No effect. Zero when read |

918 . QEI_INTR_ACK Register Description

QEI_INTR_MASK_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:6 | R/W | Reserved | 0 | Reserved. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------------------------|-----------|--|
| 5 | R/W | Velocity_computation_over_intr_mask | | <p>Velocity_computation_over_intr_mask</p> <p>If you write 1 – Velocity computation over intr will not be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Velocity less than intr is given on qei_intr pin. 0 – Velocity less than intr is not given on qei_intr pin.</p> |
| 4 | R/W | Qei_position_cnt_match_intr_mask | 0 | <p>Qei_position_cnt_match_intr_mask</p> <p>If you write 1 – Qei position cnt match intr will not be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Qei position cnt match intr is given on qei_intr pin. 1 – Qei position cnt match intr is given on qei_intr pin.</p> |
| 3 | R/W | Velocity_less_than_intr_mask | 0 | <p>Velocity_less_than_intr_mask</p> <p>If you write 1 – Velocity less than intr will not be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Velocity less than intr is given on qei_intr pin. 0 – Velocity less than intr is not given on qei_intr pin.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------------|-----------|--|
| 2 | R/W | Position_cntr_err_intr_mask | 0 | <p>Position_cntr_err_intr_mask</p> <p>If you write</p> <p>1 – Position cntr err intr will not be given on qei_intr pin.</p> <p>0 – No effect.</p> <p>If you read</p> <p>1 – Position cntr err intr is given on qei_intr pin.</p> <p>0 – Position cntr err intr is not given on qei_intr pin.</p> |
| 1 | R/W | Qei_index_cnt_match_intr_mask | 0 | <p>Position_cntr_err_intr_mask</p> <p>If you write</p> <p>1 – Qei index cnt match intr will not be given on qei_intr pin.</p> <p>0 – No effect.</p> <p>If you read</p> <p>1 – Qei index cnt match intr is given on qei_intr pin.</p> <p>0 – Qei index cnt match intr is not given on qei_intr pin.</p> |
| 0 | R/W | Qei_position_cnt_reset_intr_0mask | 0 | <p>Qei_position_cnt_reset_intr_mask</p> <p>If you write</p> <p>1 – Qei position cnt reset intr will not be given on qei_intr pin.</p> <p>0 – No effect.</p> <p>If you read</p> <p>1 – Qei position cnt reset intr is given on qei_intr pin.</p> <p>0 – Qei position cnt reset intr is given on qei_intr pin.</p> |

919 . QEI_INTR_MASK Register Description

QEI_INTR_UNMASK_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:5 | R/W | Reserved | 0 | Reserved. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------------------|-----------|--|
| 4 | R/W | Qei_index_cnt_match_intr_unmask | 0 | <p>Qei_position_cnt_match_intr_unmask</p> <p>If you write 1 – Qei position cnt match intr will be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Qei position cnt match intr is given on qei_intr pin. 1 – Qei position cnt match intr is given on qei_intr pin.</p> |
| 3 | R/W | Velocity_less_than_intr_unmask | 0 | <p>Velocity_less_than_intr_unmask</p> <p>If you write 1 – Velocity less than intr will be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Velocity less than intr is given on qei_intr pin. 0 – Velocity less than intr is not given on qei_intr pin.</p> |
| 2 | R/W | Position_cntr_err_intr_unmask | 0 | <p>Position_cntr_err_intr_unmask</p> <p>If you write 1 – Position cntr err intr will be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Position cntr err intr is given on qei_intr pin. 0 – Position cntr err intr is not given on qei_intr pin.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------------------|-----------|--|
| 1 | R/W | Qei_index_cnt_match_intr_unmask | 0 | <p>Position_cntr_err_intr_unmask</p> <p>If you write 1 – Qei index cnt match intr will be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Qei index cnt match intr is given on qei_intr pin. 1 – Qei index cnt match intr is not given on qei_intr pin.</p> |
| 0 | R/W | Qei_position_cnt_reset_intr_unmask | 0 | <p>Qei_position_cnt_err_intr_unmask</p> <p>If you write 1 – Qei position cnt reset intr will be given on qei_intr pin. 0 – No effect.</p> <p>If you read 1 – Qei position cnt reset intr is given on qei_intr pin. 1 – Qei position cnt reset intr is given on qei_intr pin.</p> |

920 . QEI_INTR_UNMASK Register Description

QEI_CLK_FREQ_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|-----------|---|
| 31:9 | R/W | Reserved | 0 | Reserved. |
| 8:0 | R/W | Qei_clk_freq | 39 | <p>Indication of clock frequency on which QEI controller is running.</p> <p>This should be loaded with running clk freq – 1. For generating real time micro sec QEI uses this value.</p> <p>For ex : If QEI is running on 40 MHz this should be loaded with 39.</p> |

921 . QEI_CLK_FREQ Register Description

QEI_DELTA_TIME_REG

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:20 | R/W | Reserved | 0 | Reserved. |

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------|-----------|---------------------------------|
| 19:0 | R/W | Delta_time_for_velocity | 999 | Delta time to compute velocity. |

922 . QEI_DELTA_TIME Register Description

QEI_VELOCITY_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------|-----------|---|
| 31:0 | R/W | Velocity_value_to_compare | 0* | <p>For write operation : It is the velocity value to compare with velocity count. If velocity count is less than the value given in this register, interrupt is raised.</p> <p>For read operation : It is the velocity count to compare using TA firmware. This is number of position pulses for given delta time in delta time register.</p> |

923 . QEI_VELOCITY Register Description

QEI_POSITION_MATCH_REG

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-----------|--|
| 31:0 | R/W | position_match_value | 0 | Position match value to compare the position counter. When it is matched with position counter, interrupt is raised. |

924 . QEI_POSITION_MATCH Register Description

16.14 Serial IO

16.14.1 General Description

SIO is present in MCU HP peripherals. Serial IO supports regular GPIO and enhanced serial stream processing features for 8 GPIO pins.

16.14.2 Features

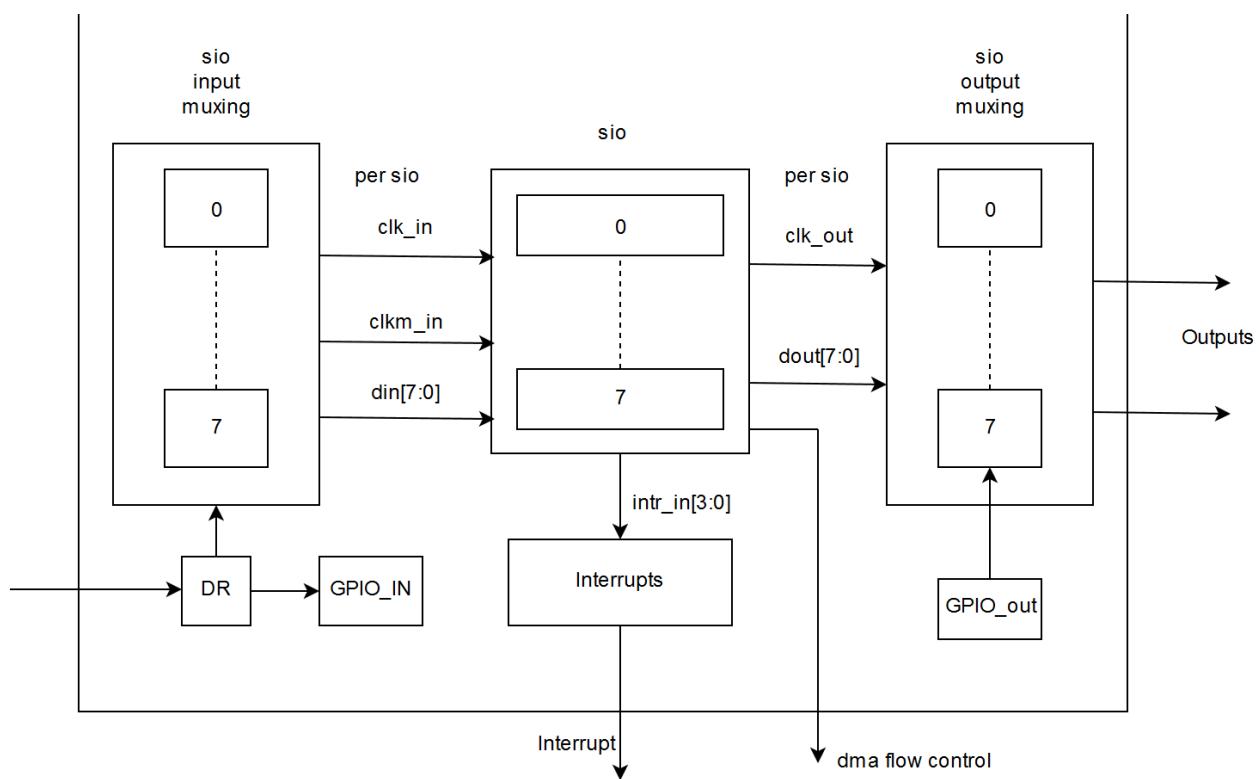
The key features of the SIO are listed below:

- Can be used to implement serial interfaces like I2C, UART and SPI protocols
- Eight GPIOs support the SIO functionality
- Supports pattern matching based interrupt generation - the GPIO is monitored to compare against a pre-programmed bit pattern. The bit pattern can be between 1 and 32 bits.
- Supports serial to parallel and parallel to serial data conversion
- Supports DMA flow control
- Supports generation of interrupt for different events.
- Programmable clock division factors for generating SIO clock out
- Performs edge and level detection on a single GPIO pinData_in[0] is monitored
 - Rise and fall edges detected
 - Level 0 and level 1 detected
- Generates a shift clock from 14-bit shift counter.

- Generates directed and inverted versions of clock
- Maintains a 32-bit shift register for shifting data
- Maintains an additional buffer for buffering additional data
- Shifts the data out and captures the input data at rising/falling edge of shift clock.
- Clock used for shift operations can be internal counter clock or external clock coming in.
- Clocks from 2 SIOs(4,5) can act as external clocks to other SIOs except SIOs 4 and 5.
- Clocks from 4 gpis(4,5,6,7) can act as external clocks to all SIOs.
- The outputs of few SIOs(4,5,12,13) can act as qualifiers for other SIOs.
- Few gpis(4,5,6,7) can act as qualifier for other SIOs.
- SIOs can be concatenated to increase buffer size.
- Few SIOs(6,7) provide OEN for other SIOs.
- Maintains 8-bit position counter for tracking shift operations.
 - The number of bits pending in the shift register is determined by the position counter.
 - Position counter decrements with every shift edge and reaches zero subsequently
 - When the position counter reaches zero, data in the shift register is exchanged with the data in buffer.
- Performs the operations based on qualifier-
 - Data shift operations are performed only when the qualifier is present
 - Qualifies clock with qualifier optionally
- Detects a 32-bit pattern on the input pins
- Generates interrupts on shift, swap (position counter reaching zero), pattern match (supported by 0,1st ,2nd slices only), GPIO edge/level detection and underun/overrun.
- Supports flow control mode in which operations and clock would be paused if data not available.
- Delays the shift operation by one shift clock period if first data has to stay on the bus for a longer duration.
- Supports loading and reading of shift data in reverse order. This feature is required for peripherals which transmit/receive MSB first.
- Data in shift registers will be repeated transferred to the stream(Self Loop Mechanism).
- Provides the following programmability:
 - Pin detection mode(EDGE/LEVEL detection)
 - Number of bits to shift
 - Posedge/Negedge clock selection for shifting
 - Internal/External clock for shifting
 - DMA flow control signal generation.
 - Flow Control
 - Delaying first shift by 1 clock edge
 - Option to send inverted clock
 - Option to qualify clock
 - Option to use direct and inverted versions of qualifier for shift operations
 - Pattern matching
 - Programmable clock division factors(for loading to 14-bit shift counter)
 - Programmable position counter (Initial value can be programmed different from the reload value. Useful for applications where different number of bits have to be shifted out during the initial phase and a fixed pattern later)
 - Default value of clock out pin before initiating the shift operations
 - Loading/reading the data in reverse order

16.14.3 Functional Description

There are 8 serial input and output processing modules internal to the SIO. Each module supports serial to parallel and parallel to serial conversion using a shift clock. Each module has 8 input data pins, 1 clock input and 1 qualifier input. It drives 8-output data pins and one output clock. Each of the 8 GPIO pins (OUT and OEN) can be driven from one or more of the SIOs. The inputs to each of these SIOs can come directly from GPIO pins or other internal SIOs. The SIO consists of a set of input multiplexers and a set of output multiplexers. With the aid of these, enhanced serial stream processing can be achieved.



SIO block diagram

16.14.4 SIO programming

Basic operation of an SIO

- Program per SIO IN_MUX_REG appropriately as per the clock, qualifier and data in inputs required to be passed to the SIO
- Program per SIO OUT_MUX_REG appropriately as per the GPIO output requirements required to be passed to the SIO
- Program the SIO config registers
- Program the BUFFER registers with data required to be shifted out
- Program the shift counter and position counter reload registers
- If pattern matching required, program the pattern match and mask registers
- Program the GPIO OUT and OEN register appropriately
- Program the interrupt enables and clear masks as required
- Reset the corresponding SIO bits in disable register
- Set the corresponding SIO bits in enable register to enable the operation

Buffer read/write operations

- Buffer lower 16-bits should be read/written first followed by buffer upper 16-bits

For disabling synchronously,

- Set the corresponding SIO bits in disable register
- Wait for disable to happen (No activity should be seen on output interface)
- Reset the corresponding SIO bits in disable register

UART Realization

- Choose one SIO slice for Rx and one for Tx (eg. SIO_0 for Tx and SIO_1 for rx)
- Default value of Tx line has to be made as 1 by setting 0th bit of GPIO_OUT_REG to 1 and also by loading FFFF to buffer reg0
- Configure OUT_MUX_REG_0 [5:3] as b100 for choosing o/p from buffer and [2:0] as b000 to chose OEN from GPIO_OEN_REG[0]
- Program DATA_POS_CNT_REG0[7:0] with number of bits to be shifted from the buffer
- Set 0th & 1st bit in SWAP_INTR_EN_SET reg for pin0 and pin1 to raise an interrupt when a word is transferred. Interrupt status can be read from SWAP_INTR_STS_REG
- Tx data can be written to BUFFER_REG_0 and Rx data can be read from BUFFER_REG_1. Extra 1 has to be padded to data to make default value of Tx line to 1
- set flow control bit in Config register to pause once data is sent
- Clocks to be used for shifting has configured in SHIFT_CNT_PRELOAD_REG_0 and SHIFT_CNT_PRELOAD_REG_1
- To detect the start bit, pattern matching has to be enabled for Rx line in CONFIG REG. To detect zero as start bit, load PATTERN_MASK_INTR_REG as 0x8000_0000 and PATTERN_MATCH_REG as 0x0000_0000. Also enable Pattern match intr enable reg for SIO1.
- In Rx SIO once pattern match is detected, pattern match enable should be made 0 in CONFIG_REG and proper value should be loaded in data position counter
- Once we get swap interrupt on Rx line, pattern match enable has to be set again and data read from buffer reg has to be shifted to receive proper data
- After doing all the configuration set SIO slices to be used in SIO_ENABLE_REG

I2C Realization

- Map one of the SIO pins to SDA line and drive 'zero' on this always by putting it in GPIO out mux mode OUT_MUX_REG [5:3] as b000
- OEN from shift register has to be used as the OEN for the SDA pad.
- For driving data, OEN should be low. Then, zero will be driven on SDA.OUT_MUX_REG [2:0] as b100
- When '1' has to be driven, OEN should be high(i.e. making SDA as input and not driving anything. As there will be pull up on I2C bus, SDA line will appear HIGH).
- Program the OEN slice's shift register in such a way that the data will be driven correctly on the bus.
- Configure one SIO for ACK and connect SDA as input line by configuring the INMUXREG
- If the ACK is proper, drive another byte by loading to shift register again
- After sending out multiple bytes, drive STOP condition from GPIO pin.

I2S Realization

- Map WS, data, SCLK to each one of SIO slices
- In SIO slice mapped to SCLK, For driving out clock, OEN should be low. Then load OUT_MUX_REG [5:3] as b000, and OUT_MUX_REG [5:3] as b001 to use SIO as clock output
- Clocks to be used for shifting has configured in SHIFT_CNT_PRELOAD_REGS use the same clock configuration for 0,1 and 2
- load the data in BUFFER_0 reg and corresponding WS as 0001_FFFE in BUFFER_1 reg
- Enable all three SIOs at once in the SIO_ENABLE_REG
- Enable swap interrupt for Slice 0 and slice 1 as next data has to be loaded once the data is shifted out

16.14.5 Register Summary

Base Address: 0x4700_0000

| Register Name | Offset |
|------------------------------|--------|
| SIO_ENABLE_REG | 0x0 |
| SIO_PAUSE_REG | 0x4 |
| SIO_GPIO_IN_REG | 0x8 |
| SIO_GPIO_OUT_REG | 0xC |
| SIO_GPIO_OEN_REG | 0x10 |
| SIO_GPIO_INTR_EN_SET_REG | 0x14 |
| SIO_GPIO_INTR_EN_CLEAR_REG | 0x18 |
| SIO_GPIO_INTR_MASK_SET_REG | 0x1C |
| SIO_GPIO_INTR_MASK_CLEAR_REG | 0x20 |
| SIO_GPIO_INTR_STATUS_REG | 0x24 |
| SIO_OUT_MUX_REG_0 | 0x230 |
| SIO_OUT_MUX_REG_1 | 0x234 |
| SIO_OUT_MUX_REG_2 | 0x238 |
| SIO_OUT_MUX_REG_3 | 0x23C |
| SIO_OUT_MUX_REG_4 | 0x240 |
| SIO_OUT_MUX_REG_5 | 0x244 |
| SIO_OUT_MUX_REG_6 | 0x248 |
| SIO_OUT_MUX_REG_7 | 0x24C |
| SIO_OUT_MUX_REG_8 | 0x250 |
| SIO_OUT_MUX_REG_9 | 0x254 |
| SIO_OUT_MUX_REG_10 | 0x258 |
| SIO_OUT_MUX_REG_11 | 0x25C |
| SIO_OUT_MUX_REG_12 | 0x260 |
| SIO_OUT_MUX_REG_13 | 0x264 |
| SIO_OUT_MUX_REG_14 | 0x268 |
| SIO_OUT_MUX_REG_15 | 0x26C |
| SIO_INPUT_MUX_REG_0 | 0x270 |
| SIO_INPUT_MUX_REG_1 | 0x274 |
| SIO_INPUT_MUX_REG_2 | 0x278 |
| SIO_INPUT_MUX_REG_3 | 0x27C |
| SIO_INPUT_MUX_REG_4 | 0x280 |
| SIO_INPUT_MUX_REG_5 | 0x284 |

| Register Name | Offset |
|--------------------------|--------|
| SIO_INPUT_MUX_REG_6 | 0x288 |
| SIO_INPUT_MUX_REG_7 | 0x28C |
| SIO_INPUT_MUX_REG_8 | 0x290 |
| SIO_INPUT_MUX_REG_9 | 0x294 |
| SIO_INPUT_MUX_REG_10 | 0x298 |
| SIO_INPUT_MUX_REG_11 | 0x29C |
| SIO_INPUT_MUX_REG_12 | 0x2A0 |
| SIO_INPUT_MUX_REG_13 | 0x2A4 |
| SIO_INPUT_MUX_REG_14 | 0x2A8 |
| SIO_INPUT_MUX_REG_15 | 0x2AC |
| SIO_SHIFT_COUNTER_REG_0 | 0x28 |
| SIO_SHIFT_COUNTER_REG_1 | 0x2C |
| SIO_SHIFT_COUNTER_REG_2 | 0x30 |
| SIO_SHIFT_COUNTER_REG_3 | 0x34 |
| SIO_SHIFT_COUNTER_REG_4 | 0x38 |
| SIO_SHIFT_COUNTER_REG_5 | 0x3C |
| SIO_SHIFT_COUNTER_REG_6 | 0x40 |
| SIO_SHIFT_COUNTER_REG_7 | 0x44 |
| SIO_SHIFT_COUNTER_REG_8 | 0x48 |
| SIO_SHIFT_COUNTER_REG_9 | 0x4C |
| SIO_SHIFT_COUNTER_REG_10 | 0x50 |
| SIO_SHIFT_COUNTER_REG_11 | 0x54 |
| SIO_SHIFT_COUNTER_REG_12 | 0x58 |
| SIO_SHIFT_COUNTER_REG_13 | 0x5C |
| SIO_SHIFT_COUNTER_REG_14 | 0x60 |
| SIO_SHIFT_COUNTER_REG_15 | 0x64 |
| SIO_BUFFER_REG_0 | 0x68 |
| SIO_BUFFER_REG_1 | 0x6C |
| SIO_BUFFER_REG_2 | 0x70 |
| SIO_BUFFER_REG_3 | 0x74 |
| SIO_BUFFER_REG_4 | 0x78 |
| SIO_BUFFER_REG_5 | 0x7C |
| SIO_BUFFER_REG_6 | |

| Register Name | Offset |
|--------------------------------|--------|
| SIO_BUFFER_REG_7 | 0x80 |
| SIO_BUFFER_REG_8 | 0x84 |
| SIO_BUFFER_REG_9 | 0x88 |
| SIO_BUFFER_REG_10 | 0x8C |
| SIO_BUFFER_REG_11 | 0x90 |
| SIO_BUFFER_REG_12 | 0x94 |
| SIO_BUFFER_REG_13 | 0x98 |
| SIO_BUFFER_REG_14 | 0x9C |
| SIO_BUFFER_REG_15 | 0xA0 |
| SIO_SHIFT_COUNT_PRELOAD_REG_0 | 0xA4 |
| SIO_SHIFT_COUNT_PRELOAD_REG_1 | 0xA8 |
| SIO_SHIFT_COUNT_PRELOAD_REG_2 | 0xAC |
| SIO_SHIFT_COUNT_PRELOAD_REG_3 | 0xB0 |
| SIO_SHIFT_COUNT_PRELOAD_REG_4 | 0xB4 |
| SIO_SHIFT_COUNT_PRELOAD_REG_5 | 0xB8 |
| SIO_SHIFT_COUNT_PRELOAD_REG_6 | 0xBC |
| SIO_SHIFT_COUNT_PRELOAD_REG_7 | 0xC0 |
| SIO_SHIFT_COUNT_PRELOAD_REG_8 | 0xC4 |
| SIO_SHIFT_COUNT_PRELOAD_REG_9 | 0xC8 |
| SIO_SHIFT_COUNT_PRELOAD_REG_10 | 0xCC |
| SIO_SHIFT_COUNT_PRELOAD_REG_11 | 0xD0 |
| SIO_SHIFT_COUNT_PRELOAD_REG_12 | 0xD4 |
| SIO_SHIFT_COUNT_PRELOAD_REG_13 | 0xD8 |
| SIO_SHIFT_COUNT_PRELOAD_REG_14 | 0xDC |
| SIO_SHIFT_COUNT_PRELOAD_REG_15 | 0xE0 |
| SIO_DATA_POS_COUNT_REG_0 | 0xE4 |
| SIO_DATA_POS_COUNT_REG_1 | 0xE8 |
| SIO_DATA_POS_COUNT_REG_2 | 0xEC |
| SIO_DATA_POS_COUNT_REG_3 | 0xF0 |
| SIO_DATA_POS_COUNT_REG_4 | 0xF4 |
| SIO_DATA_POS_COUNT_REG_5 | 0xF8 |
| SIO_DATA_POS_COUNT_REG_6 | 0xFC |

| Register Name | Offset |
|-------------------------------------|--------|
| SIO_DATA_POS_COUNT_REG_7 | 0x100 |
| SIO_DATA_POS_COUNT_REG_8 | 0x104 |
| SIO_DATA_POS_COUNT_REG_9 | 0x108 |
| SIO_DATA_POS_COUNT_REG_10 | 0x10C |
| SIO_DATA_POS_COUNT_REG_11 | 0x110 |
| SIO_DATA_POS_COUNT_REG_12 | 0x114 |
| SIO_DATA_POS_COUNT_REG_13 | 0x118 |
| SIO_DATA_POS_COUNT_REG_14 | 0x11C |
| SIO_DATA_POS_COUNT_REG_15 | 0x120 |
| SIO_CONFIG_REG_0 | 0x124 |
| SIO_CONFIG_REG_1 | 0x128 |
| SIO_CONFIG_REG_2 | 0x12C |
| SIO_CONFIG_REG_3 | 0x130 |
| SIO_CONFIG_REG_4 | 0x134 |
| SIO_CONFIG_REG_5 | 0x138 |
| SIO_CONFIG_REG_6 | 0x13C |
| SIO_CONFIG_REG_7 | 0x140 |
| SIO_CONFIG_REG_8 | 0x144 |
| SIO_CONFIG_REG_9 | 0x148 |
| SIO_CONFIG_REG_10 | 0x14C |
| SIO_CONFIG_REG_11 | 0x150 |
| SIO_CONFIG_REG_12 | 0x154 |
| SIO_CONFIG_REG_13 | 0x158 |
| SIO_CONFIG_REG_14 | 0x15C |
| SIO_CONFIG_REG_15 | 0x160 |
| SIO_PATTERN_MATCH_MASK_REG_slice_0 | 0x164 |
| SIO_PATTERN_MATCH_MASK_REG_slice_1 | 0x168 |
| SIO_PATTERN_MATCH_MASK_REG_slice_2 | 0x16C |
| SIO_PATTERN_MATCH_MASK_REG_slice_8 | 0x170 |
| SIO_PATTERN_MATCH_MASK_REG_slice_9 | 0x188 |
| SIO_PATTERN_MATCH_MASK_REG_slice_10 | 0x18C |
| SIO_PATTERN_MATCH_REG_slice_0 | 0x190 |

| Register Name | Offset |
|---------------------------------------|--------|
| SIO_PATTERN_MATCH_REG_slice_1 | 0x1A8 |
| SIO_PATTERN_MATCH_REG_slice_2 | 0x1AC |
| SIO_PATTERN_MATCH_REG_slice_8 | 0x1B0 |
| SIO_PATTERN_MATCH_REG_slice_9 | 0x1C8 |
| SIO_PATTERN_MATCH_REG_slice_10 | 0x1CC |
| SIO_SHIFT_INTR_EN_SET_REG | 0x1D0 |
| SIO_SHIFT_INTR_EN_CLEAR_REG | 0x1F0 |
| SIO_SHIFT_INTR_MASK_SET_REG | 0x1F4 |
| SIO_SHIFT_INTR_MASK_CLEAR_REG | 0x1F8 |
| SIO_SHIFT_INTR_STATUS_REG | 0x1FC |
| SIO_SWAP_INTR_EN_SET_REG | 0x200 |
| SIO_SWAP_INTR_EN_CLEAR_REG | 0x204 |
| SIO_SWAP_INTR_MASK_SET_REG | 0x208 |
| SIO_SWAP_INTR_MASK_CLEAR_REG | 0x20C |
| SIO_SWAP_INTR_STATUS_REG | 0x210 |
| SIO_PATTERN_MATCH_INTR_EN_SET_REG | 0x214 |
| SIO_PATTERN_MATCH_INTR_EN_CLEAR_REG | 0x218 |
| SIO_PATTERN_MATCH_INTR_MASK_SET_REG | 0x21C |
| SIO_PATTERN_MATCH_INTR_MASK_CLEAR_REG | 0x220 |
| SIO_PATTERN_MATCH_INTR_STATUS_REG | 0x224 |
| SIO_BUFFER_INTR_STATUS_REG | 0x228 |
| SIO_FIFO_WR_RD_REG | 0x22C |
| SIO_FIFO_WR_OFFSET_START_REG | 0x2B0 |
| SIO_FIFO_WR_OFFSET_END_REG | 0x2B4 |
| SIO_FIFO_WR_OFFSET_CNT_REG | 0x2B8 |
| SIO_FIFO_RD_OFFSET_START_REG | 0x2BC |
| SIO_FIFO_RD_OFFSET_END_REG | 0x2C0 |
| SIO_FIFO_RD_OFFSET_CNT_REG | 0x2C4 |
| | 0x2C8 |

925 . Register Summary Table

16.14.6 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write

SIO_ENABLE_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------------------|-----------|--|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | sio operation enable | 0x00 | <p>Contains the enables for all SIO.</p> <p>When any bit is '1', corresponding SIO is enabled. Shift counter and position counters starts decrementing.</p> <p>Shift operations happen. Output clock is provided. Position counter pauses if position counter disable is set for the corresponding SIO.</p> <p>When any bit is '0', corresponding SIO is disabled. Shift operations are paused and counters set to reload values. Counters do not decrement.</p> |

926 . ENABLE_REG Description

SIO_PAUSE_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|------------------------------|-----------|--|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | SIO position counter disable | 0x00 | <p>Contains sio position counter disable for all SIOs. Each bit in the register corresponds to an SIO.</p> <p>When set to '1', indicates that the position counter in the corresponding SIO has to be disabled. Position counter will not be disabled immediately.</p> <p>It counts to zero and then gets disabled. This is for synchronous operation.</p> <p>When set to '0', position counter decrements if SIO is enabled</p> |

927 . PAUSE_REG Description

SIO_GPIO_IN_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------|-----------|---|
| [31:0] | R | In value | 0x0000 | GPIO input pin status. It is 0 on reset and reflect pin status when once clk is up. |

928 .GPIO_IN_REG Description

SIO_GPIO_OUT_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|-----------|-----------|-------------------------------------|
| [31:0] | R/W | Out value | 0 | Value to be loaded on GPIO out pins |

929 .GPIO_OUT_REG Description

SIO_GPIO_OEN_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|-----------|-----------|-----------------------|
| [31:0] | R/W | OEN value | 0xffff | OEN for the GPIO pins |

930 .GPIO_IN_REG Description

SIO_GPIO_INTR_EN_SET_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr enable set | 0 | Common gpio interrupt enable set register for all SIOs. Each bit corresponds to one SIO. When '1' is written, gpio interrupt enable gets set. Interrupt would be raised if not masked and interrupt status would be reflected in interrupt status register. When '0' is written, there is no effect. Upon read, a '1' indicates that the interrupt is enabled. '0' indicates, interrupt is disabled |

931 .GPIO_INTR_EN_SET_REG Description

SIO_GPIO_INTR_EN_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------|-----------|-------------|
| [31:16] | R/W | Reserved | 0x00 | Reserved |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| [15:0] | W | intr enable set | 0 | <p>Common gpio interrupt enable clear register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, gpio interrupt enable gets cleared. Interrupt would not be raised and interrupt status would not be reflected in interrupt status register.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt is enabled.</p> <p>'0' indicates, interrupt is disabled</p> |

932 .GPIO_INTR_EN_CLEAR_REG Description

SIO_GPIO_INTR_MASK_SET_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr mask set | 0xffff | <p>Common gpio interrupt mask set register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, gpio interrupt mask gets set. Interrupt would not be raised but interrupt status would be reflected in interrupt status register if interrupt enabled.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt mask is set.</p> <p>'0' indicates, interrupt mask is not present</p> |

933 .GPIO_INTR_MASK_SET_REG Description

SIO_GPIO_INTR_MASK_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | W | intr mask clear | 0xffff | <p>Common gpio interrupt mask clear register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, gpio interrupt mask gets cleared. Interrupt would be raised if enabled and occurs. Interrupt status would be reflected in interrupt status register if enabled.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt mask is set.</p> <p>'0' indicates, interrupt mask is not present</p> |

934 .GPIO_INTR_MASK_CLEAR_REG Description

SIO_GPIO_INTR_STATUS_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr enable set | 0 | Common gpio interrupt status register for all SIOs. Upon read, 0 - intr not present/not enabled, 1 - intr present. Writing a '1' clears the corresponding bit. Writing a zero does not have effect. |

935 .GPIO_INTR_STATUS_REG Description

SIO_OUT_MUX_REG_N

There are 16 out_mux_regN(N = 0 to 15) each for a slice. Controls the GPIO data out and oen muxing for all 16 GPIO pins. Please refer to the architecture diagram for muxing information.

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------|-----------|----------------------------------|
| [31:6] | R | Reserved | 0 | Reserved |
| [5:3] | R/W | dout sel | 0 | Output mux select for GPIO pin N |
| [2:0] | R/W | dout oen sel | 0 | OEN select for GPIO pin N |

936 .OUT_MUX_REG_N Description

SIO_INPUT_MUX_REG_N

There are 16 input_mux_reg_N(N = 0 to 15) each for a slice. Controls the SIO input data, clock and qualifier muxing for all 16 SIO modules. Please refer to the architecture diagram for muxing information.

| Bit | Access | Function | POR Value | Description |
|---------|--------|------------------|-----------|------------------------------|
| [31:10] | R | Reserved | 0 | Reserved |
| [9:7] | R/W | din sel | 0 | Data in mux select |
| [6:5] | R/W | qualifier mode | 0 | qualifier mode |
| [4:3] | R/W | qualifier select | 0 | qualifier select |
| [2:0] | R/W | clk sel | 0 | Input clock select for SIO N |

937 .INPUT_MUX_REG_N Description

SIO_SHIFT_COUNTER_REG_N

There are 16 Shift_counter_N(N = 0 to 15) registers each for a slice.

| Bit | Access | Function | POR Value | Description |
|---------|--------|---------------|-----------|-----------------------------|
| [31:14] | R | Reserved | 0 | Reserved |
| [13:0] | R | shift counter | 0 | shift counter current value |

938 .SHIFT_COUNTER_REG_N Description

SIO_BUFFER_REG_N

There are 16 Buffer_Reg_N(N = 0 to 15) registers each for a slice.

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------|-----------|--|
| [31:0] | R/W | Data | 0 | Data to loaded into the shift register when swap occurs/ Data to be latched from the shift register. Carries lower 16 bits |

939 .BUFFER_REG_N Description

SIO_SHIFT_COUNT_PRELOAD_REG_N

There are 16 Shift_count_preload_reg_N(N = 0 to 15) registers each for a slice.

| Bit | Access | Function | POR Value | Description |
|---------|--------|--------------|-----------|--|
| [31:16] | R | Reserved | 0 | Reserved |
| 15 | R/W | reverse_load | 0 | When set, the data on APB is loaded to buffer is reverse order. If the requirement is to shift out the MSB first, this has to be enabled. If data sample is less than 32 bits, data sample has to be shifted by software so that the MSB is aligned to the left most bit and then programmed to hardware. |
| 14 | R | Reserved | 0 | Reserved |
| [13:0] | R/W | reload value | 0 | division factor required to generate shift clk. Shift clk freq = sgpio input clk/2*(counter + 1) Max freq of shift clock = input clk /2 |

940 .SHIFT_COUNT_PRELOAD_REG_N Description

SIO_DATA_POS_COUNT_REG_N

There are 16 Data_pos_count_reg_N(N = 0 to 15) registers each for a slice.

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------|-----------|-------------|
| [31:16] | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------|-----------|---|
| [15:8] | R/W | position counter | 0 | The position counter can be loaded via AHB only when the load_pos_cntr_via_apb bit is set in the config register 2 and the sio enable is reset. Otherwise, position counter gets reloaded with reload value when it reaches zero. |
| [7:0] | R/W | reload value | 0 | No. of shifts to happen before reloading the shift register with data/ pausing the operation. i.e. value to be set = (total no. of valid bits in shift reg/ number of bits per shift) - 1 When SIOs are cascaded, this value should be equal to the (total number of bits to be shifted/ number of bits per shift) -1 |

941 . DATA_POS_COUNT_REG_N Description

SIO_CONFIG_REG_N

There are 16 Config_reg_N(N = 0 to 15) registers each for 1 slice.

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------------------------|-----------|--|
| [31:17] | R | Reserved | 0 | Reserved |
| 16 | R/W | Load_data_pos_cntr_via_apb | 0 | When set, data position counter can be loaded via APB. This allows us to have a different initial value instead of using the reload value as init value. |
| 15 | R/W | reset_clk_out | 0 | When high resets the sio clock_out port. This is used only when sio is not enabled |
| 14 | R/W | set_clk_out | 0 | When high sets the sio clock_out port. This is used only when sio is not enabled |
| [13:12] | R/W | pin detection mode | 0 | Pin mode to be considered for gpio interrupt 2'b00 – rise edge 2'b01- fall edge 2'b10 – level zero 2'b11 – level one |
| [11:10] | R/W | parallel mode | 0 | No. of bits to shift/capture at valid clk edge 2'b00 – 1 bit 2'b01- 2 bits 2'b10 – 4 bits 2'b11 – 8 bits |
| 9 | R/W | invert clock | 0 | 0 - direct version of shift clock is provided out 1 - inverted version of the clock is provided out |
| 8 | R/W | qualify clock | 0 | 0 - output clock is not qualified 1 - output clock is qualified with qualifier |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------------|-----------|---|
| 7 | R/W | qualifier mode | 0 | 0 - Use direct qualifier input. 1 - Use inverted qualifier |
| 6 | R/W | pattern match enable | 0 | 0 - pattern matching disabled. 1 - pattern matching enabled if supported by the SIO. When enabled, pattern match register should be programmed with data pattern to be detected. Pattern mask register to be programmed with valid bits to match. Note: Data buffering is not allowed in this case. Buffer reg is not valid. |
| 5 | R/W | flow control enabled | 0 | 0 - flow control not enabled. Data shifting continues even when data is shift register is not valid. Counters keep moving. 1 - flow control is enabled. Counters and shift operations pause when data is not available. |
| 4 | R/W | ignore first shift condition | 0 | 0 - data shift/capture happens at the first clock edge(pos/neg as programmed) that occurs immediately after SIO is enabled. Useful in cases where posedge driving/sampling is enabled and frequency is high. In that case, if data was shifted at first edge, data was put on the bus only for half clock and may not be sufficient for other side to sample. By ignoring first edge, data will be made available for 1 and $\frac{1}{2}$ cycle. |
| 3 | R/W | clk sel | 0 | 0 - internal counter clock is used for shift operations and sent out 1 - external clock is used for shift operations and is sent out When external clock is used, shift/capture happens one system clock after the external clock edge |
| 2 | R/W | edge sel | 0 | 0 - posedge 1 - negedge |
| 1 | R/W | empty enable | 0 | When set, fifo full indication would be asserted when internal buffer is empty |
| 0 | R/W | full enable | 0 | When set, fifo full indication would be asserted when internal buffer is full |

942 . CONFIG_REG_N Description

SIO_PATTERN_MATCH_MASK_REG_slice_k

There are 6 Pattern_match_mask_reg_slice_k (k = 0,1,2,8,9,10) registers each for a slice.

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------------------|-----------|---|
| [31:0] | R/W | Match mask lower 16 bits | 0 | Enable for lower 16 bits. When any bit is set, it is considered in pattern match When bit is zero, it is considered as don't care |

943 . PATTERN_MATCH_MASK_REG_slice_k Description

SIO_PATTERN_MATCH_REG_slice_k

There are 6 Pattern_match_reg_L_slice_k (k = 0,1,2,8,9,10) registers each for a slice.

| Bit | Access | Function | POR Value | Description |
|--------|--------|-----------------------------|-----------|---|
| [31:0] | R/W | Pattern match lower 16 bits | 0 | Lower 16-bits of pattern to be detected |

944 . PATTERN_MATCH_REG_slice_k Description

SIO_SHIFT_INTR_EN_SET_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr enable set | 0 | Common shift interrupt enable set register for all SIOs. Each bit corresponds to one SIO. When '1' is written, shift interrupt enable gets set. Interrupt would be raised if not masked and interrupt status would be reflected in interrupt status register. When '0' is written, there is no effect. Upon read, a '1' indicates that the interrupt is enabled. '0' indicates, interrupt is disabled |

945 . SHIFT_INTR_EN_SET_REG Description

SIO_SHIFT_INTR_EN_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------|-----------|-------------|
| [31:16] | R/W | Reserved | 0x00 | Reserved |

| Bit | Access | Function | POR Value | Description |
|--------|--------|-------------------|-----------|--|
| [15:0] | W | intr enable clear | 0 | <p>Common shift interrupt enable clear register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, shift interrupt enable gets cleared. Interrupt would not be raised and interrupt status would not be reflected in interrupt status register.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt is enabled.</p> <p>'0' indicates, interrupt is disabled</p> |

946 .SHIFT_INTR_EN_CLEAR_REG Description

SIO_SHIFT_INTR_MASK_SET_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|---------------|-----------|--|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr mask set | 0xffff | <p>Common shift interrupt mask set register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, shift interrupt mask gets set. Interrupt would not be raised but interrupt status would be reflected in interrupt status register if interrupt enabled.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt mask is set.</p> <p>'0' indicates, interrupt mask is not present</p> |

947 .SHIFT_INTR_MASK_SET_REG Description

SIO_SHIFT_INTR_MASK_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | W | intr mask clear | 0xffff | <p>Common shift interrupt mask clear register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, shift interrupt mask gets cleared. Interrupt would be raised if enabled and occurs. Interrupt status would be reflected in interrupt status register if enabled.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt mask is set.</p> <p>'0' indicates, interrupt mask is not present</p> |

948 .SHIFT_INTR_MASK_CLEAR_REG Description

SIO_SHIFT_INTR_STATUS_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr enable set | 0 | Common shift interrupt status register for all SIOs. Upon read, 0 - intr not present/not enabled 1 - intr present. Writing a '1' clears the corresponding bit. Writing a zero does not have effect. |

949 .SHIFT_INTR_STATUS_REG Description

SIO_SWAP_INTR_EN_SET_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr enable set | 0 | Common swap interrupt enable set register for all SIOs. Each bit corresponds to one SIO. When '1' is written, swap interrupt enable gets set. Interrupt would be raised if not masked and interrupt status would be reflected in interrupt status register. When '0' is written, there is no effect. Upon read, a '1' indicates that the interrupt is enabled. '0' indicates, interrupt is disabled |

950 .SWAP_INTR_EN_SET_REG Description

SIO_SWAP_INTR_EN_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | W | intr enable set | 0 | Common swap interrupt enable clear register for all SIOs. Each bit corresponds to one SIO. When '1' is written, swap interrupt enable gets cleared. Interrupt would not be raised and interrupt status would not be reflected in interrupt status register. When '0' is written, there is no effect. Upon read, a '1' indicates that the interrupt is enabled. '0' indicates, interrupt is disabled |

951 .SWAP_INTR_EN_CLEAR_REG Description

SIO_SWAP_INTR_MASK_SET_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|---------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr mask set | 0xffff | Common swap interrupt mask set register for all SIOs. Each bit corresponds to one SIO. When '1' is written, swap interrupt mask gets set. Interrupt would not be raised but interrupt status would be reflected in interrupt status register if interrupt enabled. When '0' is written, there is no effect. Upon read, a '1' indicates that the interrupt mask is set. '0' indicates, interrupt mask is not present |

952 .SWAP_INTR_MASK_SET_REG Description

SIO_SWAP_INTR_MASK_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|--|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | W | intr mask clear | 0xffff | Common swap interrupt mask clear register for all SIOs. Each bit corresponds to one SIO. When '1' is written, swap interrupt mask gets cleared. Interrupt would be raised if enabled and occurs. Interrupt status would be reflected in interrupt status register if enabled. When '0' is written, there is no effect. Upon read, a '1' indicates that the interrupt mask is set. '0' indicates, interrupt mask is not present |

953 .SWAP_INTR_MASK_CLEAR_REG Description

SIO_SWAP_INTR_STATUS_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr enable set | 0 | Common shift interrupt status register for all SIOs. Upon read, 0 - intr not present/not enabled 1 - intr present. Writing a '1' clears the corresponding bit. Writing a zero does not have effect. |

954 .SWAP_INTR_STATUS_REG Description

SIO_PATTERN_MATCH_INTR_EN_SET_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|--|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr enable set | 0 | <p>Common pattern/buffer underrun interrupt enable set register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, pattern interrupt enable gets set. Interrupt would be raised if not masked and interrupt status would be reflected in interrupt status register.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt is enabled.</p> <p>'0' indicates, interrupt is disabled</p> |

955 . PATTERN_MATCH_INTR_EN_SET_REG Description

SIO_PATTERN_MATCH_INTR_EN_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-------------------|-----------|--|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | W | intr enable clear | 0 | <p>Common pattern/buffer underrun interrupt enable clear register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, pattern interrupt enable gets cleared. Interrupt would not be raised and interrupt status would not be reflected in interrupt status register.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt is enabled.</p> <p>'0' indicates, interrupt is disabled</p> |

956 . PATTERN_MATCH_INTR_EN_CLEAR_REG Description

SIO_PATTERN_MATCH_INTR_MASK_SET_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|----------|-----------|-------------|
| [31:16] | R/W | Reserved | 0x00 | Reserved |

| Bit | Access | Function | POR Value | Description |
|--------|--------|---------------|-----------|--|
| [15:0] | R/W | intr mask set | 0xffff | <p>Common pattern/buffer underrun interrupt mask set register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, pattern interrupt mask gets set. Interrupt would not be raised but interrupt status would be reflected in interrupt status register if interrupt enabled.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt mask is set.</p> <p>'0' indicates, interrupt mask is not present</p> |

957 . PATTERN_MATCH_INTR_MASK_SET_REG Description

SIO_PATTERN_MATCH_INTR_MASK_CLEAR_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-----------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | W | intr mask clear | 0xffff | <p>Common pattern/buffer underrun interrupt mask clear register for all SIOs. Each bit corresponds to one SIO.</p> <p>When '1' is written, pattern interrupt mask gets cleared. Interrupt would be raised if enabled and occurs. Interrupt status would be reflected in interrupt status register if enabled.</p> <p>When '0' is written, there is no effect.</p> <p>Upon read, a '1' indicates that the interrupt mask is set.</p> <p>'0' indicates, interrupt mask is not present</p> |

958 . PATTERN_MATCH_INTR_MASK_CLEAR_REG Description

SIO_PATTERN_MATCH_INTR_STATUS_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr status | 0x0000 | <p>Common pattern interrupt status register for all SIOs.</p> <p>Upon read,</p> <p>0 - intr not present/not enabled</p> <p>1 - intr present.</p> <p>Writing a '1' clears the corresponding bit.</p> <p>Writing a zero does not have effect.</p> |

959 . PATTERN_MATCH_INTR_STATUS_REG Description

SIO_BUFFER_INTR_STATUS_REG

| Bit | Access | Function | POR Value | Description |
|---------|--------|-------------|-----------|---|
| [31:16] | R/W | Reserved | 0x00 | Reserved |
| [15:0] | R/W | intr status | 0 | Common underrun/overrun interrupt status register for all SIOs. Upon read, 0 - intr not present/not enabled 1 - intr present. Writing a '1' clears the corresponding bit. Writing a zero does not have effect. |

960 .BUFFER_INTR_STATUS_REG Description

SIO_FIFO_WR_RD_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------------|-----------|--|
| [31:0] | R/W | FIFO Data Register | 0x0 | Writes into this register will be written into SIO buffer register pointed by FIFO_WR_OFFSET_CNT_REG and this FIFO_WR_OFFSET_CNT_REG will be incremented by 1 Reads into this register will be directed to SIO buffer register pointed by FIFO_RD_OFFSET_CNT_REG and this FIFO_RD_OFFSET_CNT_REG will be incremented by 1 |

961 .FIFO_WR_RD_REG Description

SIO_FIFO_WR_OFFSET_START_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------------|-----------|---|
| [31:0] | R/W | SIO Start Slice Number | 0x0 | Points to start slice number forming the FIFO. On write, FIFO_WR_OFFSET_CNT_REG will also be reset to the value pointed written into this register |

962 .FIFO_WR_OFFSET_START_REG Description

SIO_FIFO_WR_OFFSET_END_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------------------|-----------|--|
| [31:0] | R/W | SIO End Slice Number | 0x0 | Points to the last slice number forming the FIFO |

963 .FIFO_WR_OFFSET_END_REG Description

SIO_FIFO_WR_OFFSET_CNT_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------------------|-----------|--|
| [31:0] | R/W | SIO Current Slice Number | 0x0 | <p>Next FIFO operation will happen to buffer in the slice pointed by this register.</p> <p>This register has to be set to zero before starting fresh DMA operation</p> |

964 . FIFO_WR_OFFSET_CNT_REG Description

SIO_FIFO_RD_OFFSET_START_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|------------------------|-----------|---|
| [31:0] | R/W | SIO Start Slice Number | 0x0 | <p>Points to start slice number forming the FIFO.</p> <p>On write, FIFO_WR_OFFSET_CNT_REG will also be reset to the value pointer has written into this register.</p> |

965 . FIFO_RD_OFFSET_START_REG Description

SIO_FIFO_RD_OFFSET_END_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|----------------------|-----------|--|
| [31:0] | R/W | SIO End Slice Number | 0x0 | Points to the last slice number forming the FIFO |

966 . FIFO_RD_OFFSET_END_REG Description

SIO_FIFO_RD_OFFSET_CNT_REG

| Bit | Access | Function | POR Value | Description |
|--------|--------|--------------------------|-----------|--|
| [31:0] | R/W | SIO Current Slice Number | 0x0 | <p>Next FIFO operation will happen to buffer in the slice pointed by this register. This register has to be set to zero before starting fresh DMA operation.</p> |

967 . FIFO_RD_OFFSET_CNT_REG Description

16.15 SSI Master

16.15.1 General Description

There are two Synchronous Serial Interface (SSI) masters - one in the MCU HP peripherals (SSI_MST) and one in the MCU ULP subsystem (ULP_SSI_MST). The SSI masters are programmable controllers that can be configured to support different full-duplex master synchronous serial interface protocols.

The SSI master can connect to any serial-slave peripheral device using one of the following interfaces:

- Motorola Serial Peripheral Interface (SPI)
- Texas Instruments Serial Protocol (SSP)

- National Semiconductor Microwire.

16.15.2 Features

Each of these SSI Masters support the following features:

- Support for Motorola SPI, TI SSP and National Semiconductors Microwire protocols
- The SSI_MST in MCU HP peripherals provides an option to connect upto four slaves and supports Single, Dual and Quad modes.
- The ULP_SSI_MST in the MCU ULP peripherals supports Single-bit mode and can be connected to only one slave
- Programmable receive sampling delay

In addition to the above features, the SSI Masters reduce the load on the processor by supporting the features below:

- Programmable FIFO thresholds with maximum FIFO depth of 16 and support for DMA
- Supports generation of interrupt for different events.
- Programmable division factor for generating SSI clock out.

The ULP_SSI_MST supports following additional power-save features:

- After the DMA is programmed in PS2 state for SSI transfers, the MCU can switch to PS1 state (processor is shutdown) while the SSI Master continues with the data transfer
- In PS1 state (ULP Peripheral mode) the SSI Master completes the data transfer and, triggered by the Peripheral Interrupt, shifts either to the sleep state (without processor intervention) or the active state

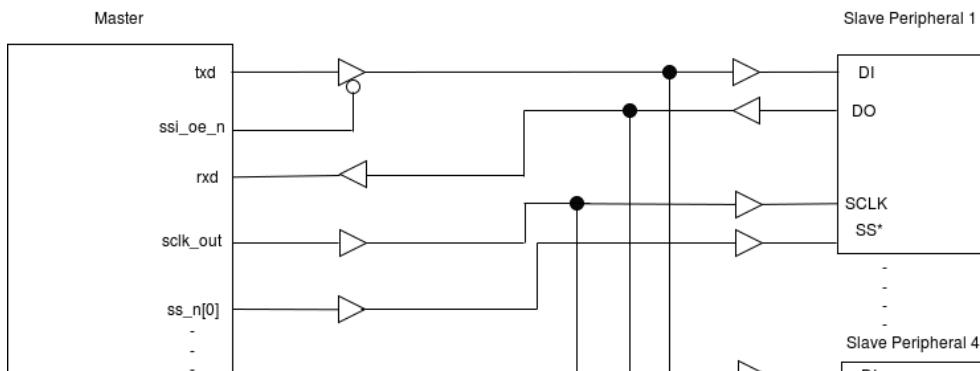
16.15.3 Functional Description

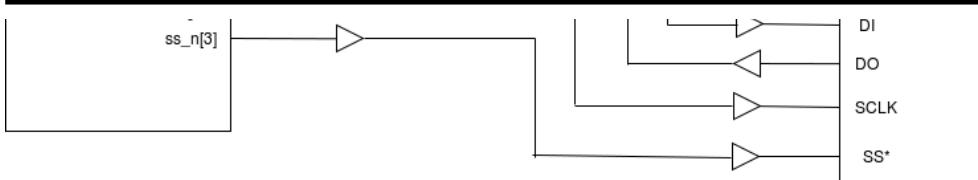
The MCU accesses data, control, and status information on the SSI master through the APB interface. This may also interface with a DMA Controller using an optional set of DMA signals. The SSI master can connect to any serial-slave peripheral device using one of the following interfaces:

- Motorola Serial Peripheral Interface (SPI): A four-wire, full-duplex serial protocol from Motorola. There are four possible combinations for the serial clock phase and polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock. The slave select line is held high when the SSI master is idle or disabled.
- Texas Instruments Serial Protocol (SSP): A four-wire, full-duplex serial protocol. The slave select line is used for SPI and Microwire protocols doubles as the frame indicator for the SSP protocol.
- National Semiconductor Microwire : A half-duplex serial protocol, which uses a control word transmitted from the serial master to the target serial slave.

FRF (frame format) bit field in the Control Register 0 (CTRLR0) is used to select which protocol is used. Four serial slave can be connected using this SSI master. It is shown in below Figure:1. The serial-master device asserts the select line of the target serial slave before data transfer begins.

If there are multiple serial masters in the system, the slave select output from all masters can be logically ANDed to generate a single slave select input for all serial slave devices.





SSI master connected to Multiple serial slaves

The frequency of the SSI master input clock (`ssi_clk`) must be less than or equal to the frequency of APB bus clock (`pclk`), which guarantees that control signals from the `ssi_clk` domain are synchronized to the `pclk` domain.

The maximum frequency of the SSI master bit-rate clock (`sclk_out`) is one-half the frequency of `ssi_clk`. This allows the shift control logic to capture data on one clock edge of `sclk_out` and propagate data on the opposite edge.

The `sclk_out` line toggles only when an active transfer is in progress. At all other times it is held in an inactive state, as defined by the serial protocol under which it operates.

The frequency of `sclk_out` can be derived from the following equation:

$$F_{sclk_out} = \frac{F_{ssi_clk}}{SCKDV}$$

`SCKDV` is a bit field in the programmable register `BAUDR`, holding any even value in the range 0 to 65,534. If `SCKDV` is 0, then `sclk_out` is disabled.

16.15.4 SSI Interrupts

The SSI master supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ORed result of all other interrupts after masking. All SSI master interrupts are level interrupts and have the same active polarity level. This polarity level can be configured as active-high or active-low. The SSI master 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 APB access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the APB is discarded. This interrupt remains set until reading of 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 reading of the receive FIFO overflow interrupt clear register (`RXOICR`).
- Receive FIFO Underflow Interrupt (`ssi_rxu_intr`) – Set when an APB 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 reading of the receive FIFO underflow interrupt clear register (`RXUICR`).
- Multi-Master Contention Interrupt (`ssi_mst_intr`) – The interrupt is set when another serial master on the serial bus selects the SSI master as a serial-slave device and is actively transferring data. This informs the processor of possible contention on the serial bus. This interrupt remains set until reading of the multi-master interrupt clear register (`MSTICR`).
- Combined Interrupt Request (`ssi_intr`) – OR'ed result of all the above interrupt requests after masking. To mask this interrupt signal, mask all other SSI interrupt requests.

16.15.5 Programming sequence

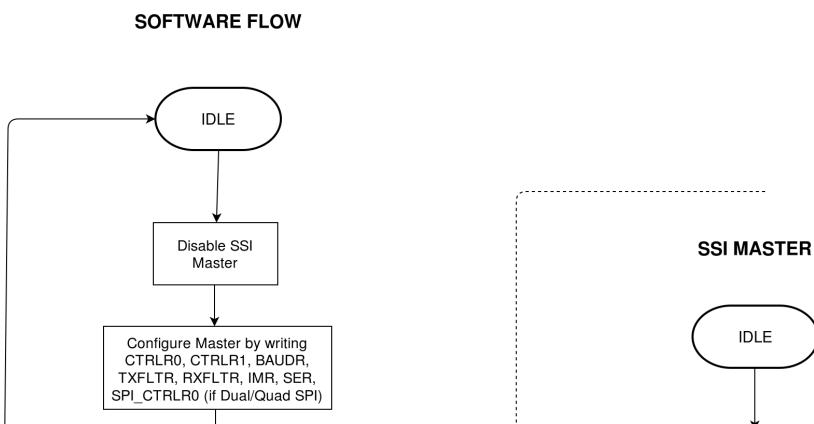
Data transfers are started by the serial-master device. When the SSI master is enabled (SSI_EN=1), at least one valid data entry is present in the transmit FIFO and a serial slave device is selected. When actively transferring data, the busy flag (BUSY) in the status register (SR) is set. Wait until the busy flag is cleared before attempting a new serial transfer.

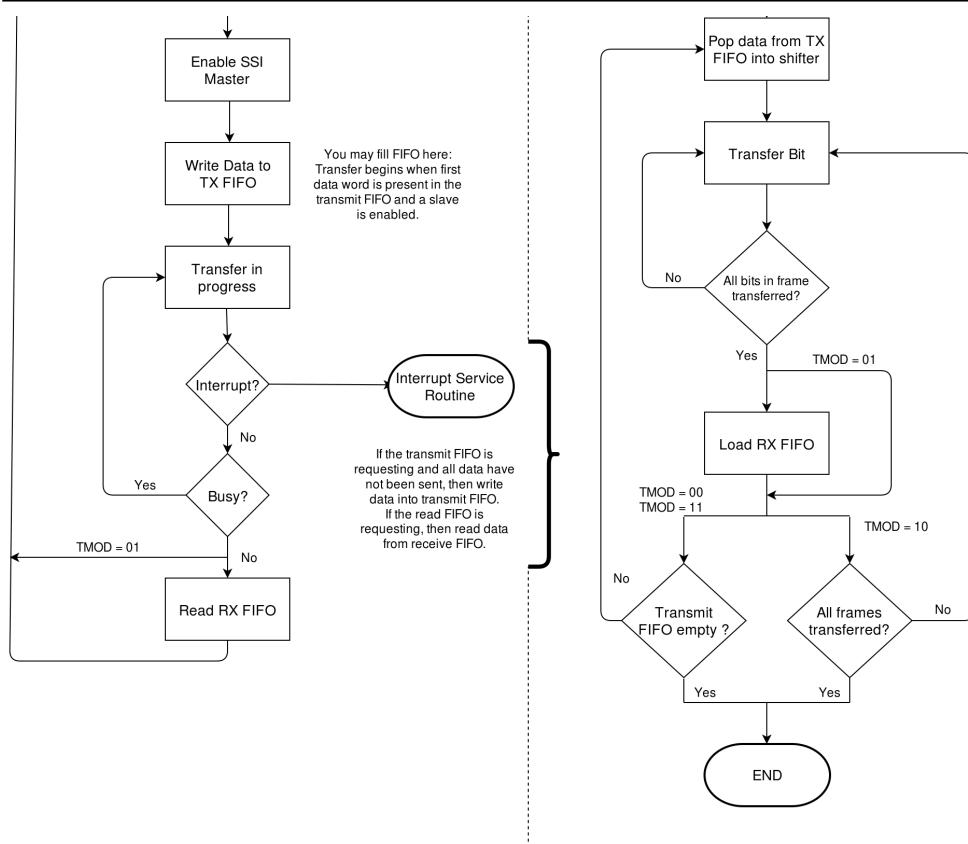
Master SPI and SSP Serial Transfers

A typical software flow for completing an SPI or SSP serial transfer from the SSI Master serial master is outlined as follows:

1. If the SSI master is enabled, disable it by writing 0 to the SSI Enable register (SSIENR).
2. Set up the SSI master control registers for the transfer; these registers can be set in any order.
 - Write Control Register 0 (CTRLR0). For SPI transfers, the serial clock polarity and serial clock phase parameters must be set identical to target slave device.
 - If the transfer mode is receive only, write CTRLR1 (Control Register 1) with the number of frames in the transfer minus 1;
 - Write the Baud Rate Select Register (BAUDR) to set the baud rate for the transfer.
 - Write the Transmit and Receive FIFO Threshold Level registers (TXFLTR and RXFLTR, respectively) to set FIFO threshold levels.
 - Write the IMR register to set up interrupt masks.
 - The Slave Enable Register (SER) register can be written here to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the Data Register (DR), the transfer does not begin until a slave is enabled.
3. Enable the SSI Master by writing 1 to the SSIENR register.
4. Write data for transmission to the target slave into the transmit FIFO (write DR).
If no slaves were enabled in the SER register at this point, enable it now to begin the transfer.
5. Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is receive only (TMOD = 2'b10), the transfer is stopped by the shift control logic when the specified number of frames have been received. When the transfer is done, the BUSY status is reset to 0.
7. If the transfer mode is not transmit only (TMOD != 01), read the receive FIFO until it is empty.
8. Disable the SSI master by writing 0 to SSIENR.

Figure:2 shows a typical software flow for starting SPI/SSP serial transfer.





SSI Master SPI/SSP Transfer Flow diagram

Master Microwire Serial Transfers

A typical software flow for completing a Microwire serial transfer from the SSI master is outlined as follows:

1. If the SSI master is enabled, disable it by writing 0 to SSIENR.
 2. Set up the SSI control registers for the transfer. These registers can be set in any order. Write CTRLR0 to set transfer parameters.
 - ? If the transfer is sequential and the SSI Master master receives data, write CTRLR1 with the number of frames in the transfer minus 1.
 - ? Write BAUDR to set the baud rate for the transfer.
 - ? Write TXFTLR and RXFTLR to set FIFO threshold levels.
 - ? Write the IMR register to set up interrupt masks.

Write the SER register to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO. If no slaves are enabled prior to writing to the DR register, the transfer does not begin until a slave is enabled.

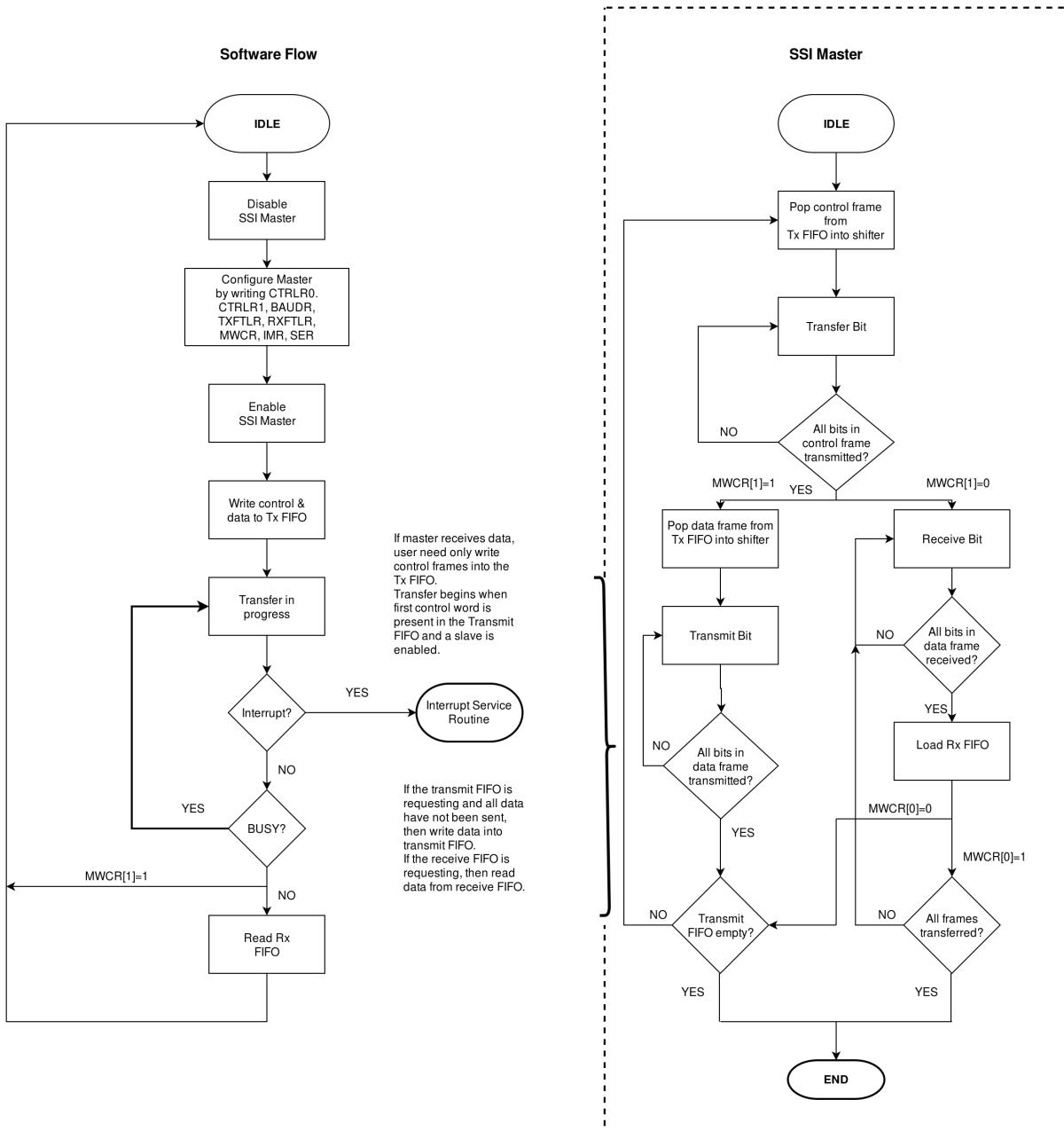
 3. Enable the SSI master by writing 1 to the SSIENR register.
 4. If the SSI master transmits data, write the control and data words into the transmit FIFO (write DR). If the SSI master receives data, write the control word(s) into the transmit FIFO.
If no slaves were enabled in the SER register at this point, enable now to begin the transfer.
 5. Poll the BUSY status to wait for completion of the transfer. The BUSY status cannot be polled immediately. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).

6. The transfer is stopped by the shift control logic when the transmit FIFO is empty. If the transfer mode is sequential and the SSI Master master receives data, the transfer is stopped by the shift control logic when the specified number of data frames is received. When the transfer is done, the BUSY status is reset to 0.

7. If the SSI master receives data, read the receive FIFO until it is empty.

8. Disable the SSI master by writing 0 to SSIENR.

Figure 3 shows a typical software flow for starting Microwire serial transfer.



Microwire serial Transfer Flow diagram using SSI Master

16.15.6 Register Summary

Base Address: 0x4402_0000

| Register Name | Offset | Description |
|-------------------|---------|---|
| SSIM_CTRLR0 | 0x00 | Control Register 0 |
| SSIM_CTRLR1 | 0x04 | Control Register 1 |
| SSIM_ENR | 0x08 | SSI Enable Register |
| SSIM_MWCR | 0x0c | Microwire Control Register |
| SSIM_SER | 0x10 | Slave Enable Register |
| SSIM_BAUDR | 0x14 | Baud Rate Select |
| SSIM_TXFTLR | 0x18 | Transmit FIFO Threshold Level Register |
| SSIM_RXFTLR | 0x1c | Receive FIFO Threshold Level Register |
| SSIM_TXFLR | 0x20 | Transmit FIFO Level Register |
| SSIM_RXFLR | 0x24 | Receive FIFO Level Register |
| SSIM_SR | 0x28 | Status Register |
| SSIM_IMR | 0x2c | Interrupt Mask Register |
| SSIM_ISR | 0x30 | Interrupt Status Register |
| SSIM_RISR | 0x34 | Raw Interrupt Status Register |
| SSIM_TXOICR | 0x38 | Transmit FIFO Overflow Interrupt Clear Register |
| SSIM_RXOICR | 0x3c | Receive FIFO Overflow Interrupt Clear Register |
| SSIM_RXUICR | 0x40 | Receive FIFO Underflow Interrupt Clear Register |
| SSIM_MSTICR | 0x44 | Multi-Master Interrupt Clear Register |
| SSIM_ICR | 0x48 | Interrupt Clear Register |
| SSIM_DMACR | 0x4c | DMA Control Register |
| SSIM_DMATDLR | 0x50 | DMA Transmit Data Level Register |
| SSIM_DMARDLR | 0x54 | DMA Receive Data Level Register |
| SSIM_IDR | 0x58 | Identification Register |
| SSIM_COMP_VERSION | 0x5c | SSI Component Version |
| SSIM_DR | 0x60-EC | Data Registers |
| SSIM_RX_SAMPLE_DL | 0xF0 | RXD Sample Delay Register |
| SSIM_SPI_CTRLR0 | 0xF4 | SPI control register |

968 . Register Summary Table

16.15.7 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, N/A = Reserved

SSIM CONTROL REGISTER 0

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:23 | N/A | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 22:21 | R/W | SPI_FRF | 0 | <p>SPI Frame Format:</p> <p>Selects data frame format for transmitting or receiving data.</p> <ul style="list-style-type: none"> • 00 – Standard SPI Format • 01 – Dual SPI Format • 10 – Quad SPI Format • 11 – Reserved |
| 20:16 | R/W | DFS_32 | 0x7 | <p>Data Frame Size. Selects the data frame length.</p> <p>Range :</p> <p>0011 -> 4 bit to 1111 -> 16 bit</p> |
| 15:12 | R/W | CFS | 0x0 | <p>Control Frame Size. Selects the length of the control word for the Microwire frame format.</p> <p>Ranges :</p> <p>0000 -> 1 bit word control to 1111 -> 16 bit word control</p> |
| 11 | R/W | SRL | 0 | <p>Shift Register Loop used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input.</p> <p>0 – Normal Mode Operation 1 – Test Mode Operation</p> |
| 10 | R/W | Reserved | 0 | Reserved |
| 9:8 | R/W | TMOD | 0 | <p>Transfer Mode. Selects the mode of transfer for serial communication.</p> <p>00 – Transmit & Receive 01 – Transmit Only 10 – Receive Only</p> |
| 7 | R/W | SCPOL | 0 | <p>Serial Clock Polarity. Valid when the frame format (FRF) is set to Motorola SPI.</p> <p>0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high</p> |
| 6 | R/W | SCPH | 0 | <p>Serial Clock Phase. Valid when the frame format (FRF) is set to Motorola SPI.</p> <p>When SCPH = 0, data are captured on the first edge of the serial clock.</p> <p>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.</p> <p>0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 5:4 | R/W | FRF | 0 | Frame Format. Selects which serial protocol transfers the data. 00 – Motorola SPI 01 – Texas Instruments SSP 10 – National Semiconductors Microwire 11 – Reserved |
| 3:0 | R/W | DFS | 0x7 | Data Frame Size. Selects the data frame length. Range : 0011 -> 4 bit to 1111 -> 16 bit |

969 . Control Register 0 Description

SSIM CONTROL REGISTER 1

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:16 | N/A | Reserved | | Reserved |
| 15:0 | R/W | NDF | 0x0 | Number of Data Frames. When TMOD = 10 or TMOD = 11, this register field sets the number of data frames to be continuously received by the SSI Master. The SSI Master continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer. |

970 . Control Register1 Description

SSIM ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 0 | R/W | SSI_EN | 0x0 | SSI Enable. Enables and disables all SSI Master operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. |

971 . SSI Enable Register Description

SSIM MICROWIRE CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:3 | N/A | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 2 | R/W | MHS | 0 | <p>Microwire Handshaking. Used to enable and disable the “busy/ready” handshaking interface for the Microwire protocol.</p> <p>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.</p> <p>0: handshaking interface is disabled 1: handshaking interface is enabled</p> |
| 1 | R/W | MDD | 0 | <p>Microwire Control. When this bit is set to 0, the data word is received by the SSI MacroCell from the external serial device.</p> <p>When this bit is set to 1, the data word is transmitted from the SSI MacroCell to the external serial device.</p> |
| 0 | R/W | MWMOD | 0 | <p>Microwire Transfer Mode. 0 – non-sequential transfer 1 – sequential transfer</p> <p>When sequential mode is used, only one control word is needed to transmit or receive a block of data words.</p> <p>When non-sequential mode is used, there must be a control word for each data word that is transmitted or received.</p> |

972 . Microwire Control Register Description

SSIM SLAVE ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:4 | N/A | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 3:0 | R/W | SER | 0 | <p>Slave Select Enable Flag. Each bit in this register corresponds to a slave select line (ss_x_n) from the SSI Master master.</p> <p>When a bit in this register is set (1), the corresponding slave select line from the master is activated when a serial transfer begins.</p> <p>It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started.</p> <p>Before beginning a transfer, the bit in this register that corresponds to the slave device with which the master wants to communicate should be enabled.</p> <p>When not operating in broadcast mode, only one bit in this field should be set.</p> <p>1: Selected 0: Not Selected</p> |

973 . Slave Enable Register Description

SSIM BAUD RATE SELECT REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 31:16 | N/A | Reserved | 0 | Reserved |
| 15:0 | R/W | SCKDV | 0 | <p>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.</p> <p>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:</p> $Fsclk_{out} = Fssi_clk/SCKDV \text{ where } SCKDV \text{ is any even value between 2 and 65534}$ |

974 . Baud Rate Select Register Description

SSIM TRANSMIT FIFO THRESHOLD LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:4 | N/A | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 3:0 | R/W | TFT | 0 | <p>Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.</p> <p>If this field is set to a value greater than or equal to the depth of the FIFO, this field is not written and retains its current value.</p> <p>When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.</p> |

975 . Transmit FIFO Threshold Level Register Description

SSIM RECEIVE FIFO THRESHOLD LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 3:0 | R/W | RFT | 0 | <p>Receive FIFO Threshold. Controls the level of entries (or below) at which the receive FIFO controller triggers an interrupt.</p> <p>If this field is set to a value greater than or equal to the depth of the FIFO, this field is not written and retains its current value.</p> <p>When the number of receive FIFO entries is less than or equal to this value + 1, the receive FIFO full interrupt is triggered.</p> |

976 . Receive FIFO Threshold Level Register

SSIM TRANSMIT FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 4:0 | R | TXTFL | 0 | Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO. |

977 . Transmit FIFO Level Register Description

SSIM RECEIVE FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--------------------|
| 31:4 | N/A | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 4:0 | R | RXTFL | 0 | Receive FIFO Level. Contains the number of valid data entries in the receive FIFO. |

978 . Receive FIFO Level Register Description

SSIM STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 6 | R | DCOL | 0 | Data Collision Error. This bit is set if the ss_in_n input is asserted by another master, while the ssi master is in the middle of the transfer. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmit data collision error |
| 5 | N/A | Reserved | 0 | Reserved |
| 4 | R | RFF | 0 | Receive FIFO Full. 0 – Receive FIFO is not full 1 – Receive FIFO is full |
| 3 | R | RFNE | 0 | Receive FIFO Not Empty. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty This bit can be polled by software to completely empty the receive FIFO. |
| 2 | R | TFE | 1 | Transmit FIFO Empty. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty This bit field does not request an interrupt. |
| 1 | R | TFNF | 1 | Transmit FIFO Not Full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full |
| 0 | R | BUSY | 0 | SSI Busy Flag. 0 – SSI is idle or disabled 1 – SSI is actively transferring data |

979 . Status Register Description

SSIM INTERRUPT MASK REGISTER

| Bit | Access | Function | POR Value | Description | |
|------|--------|----------|-----------|-------------|--|
| 31:6 | N/A | Reserved | 0 | Reserved | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 5 | R/W | MSTIM | 1 | Multi-Master Contention Interrupt Mask. 0 – ssi_mst_intr interrupt is masked 1 – ssi_mst_intr interrupt is not masked |
| 4 | R/W | RXFIM | 1 | Receive FIFO Full Interrupt Mask 0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked |
| 3 | R/W | RXOIM | 1 | Receive FIFO Overflow Interrupt Mask 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked |
| 2 | R/W | RXUIM | 1 | Receive FIFO Underflow Interrupt Mask 0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked |
| 1 | R/W | TXOIM | 1 | Transmit FIFO Overflow Interrupt Mask 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked |
| 0 | R/W | TXEIM | 1 | Transmit FIFO Empty Interrupt Mask 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked |

980 . Interrupt Mask Register Description

SSIM INTERRUPT STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:6 | N/A | Reserved | 0 | Reserved |
| 5 | R | MSTIS | 0 | Multi-Master Contention Interrupt Status. 0 – ssi_mst_intr interrupt not active after masking 1 – ssi_mst_intr interrupt is active after masking |
| 4 | R | RXFIS | 0 | Receive FIFO Full Interrupt Status 0 – ssi_rxf_intr interrupt is not active after masking 1 – ssi_rxf_intr interrupt is full after masking |
| 3 | R | RXOIS | 0 | Receive FIFO Overflow Interrupt Status 0 – ssi_rxo_intr interrupt is not active after masking 1 – ssi_rxo_intr interrupt is active after masking |
| 2 | R | RXUIS | 0 | Receive FIFO Underflow Interrupt Status 0 – ssi_rxu_intr interrupt is not active after masking 1 – ssi_rxu_intr interrupt is active after masking |
| 1 | R | TXOIS | 0 | Transmit FIFO Overflow Interrupt Status 0 – ssi_txo_intr interrupt is not active after masking 1 – ssi_txo_intr interrupt is active after masking |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 0 | R | TXEIS | 0 | Transmit FIFO Empty Interrupt Status 0 – ssi_txe_intr interrupt is not active after masking 1 – ssi_txe_intr interrupt is active after masking |

981 . Interrupt Status Register Description

SSIM RAW INTERRUPT STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:6 | N/A | Reserved | 0 | Reserved |
| 5 | R | MSTIR | 0 | Multi-Master Contention Raw Interrupt Status. 0 – ssi_mst_intr interrupt is not active prior to asking 1 – ssi_mst_intr interrupt is active prior masking |
| 4 | R | RXFIR | 0 | Receive FIFO Full Raw Interrupt Status 0 – ssi_rxf_intr interrupt is not active prior to masking 1 – ssi_rxf_intr interrupt is active prior to masking |
| 3 | R | RXOIR | 0 | Receive FIFO Overflow Raw Interrupt Status 0 – ssi_rxo_intr interrupt is not active prior to masking 1 – ssi_rxo_intr interrupt is active prior masking |
| 2 | R | RXUIR | 0 | Receive FIFO Underflow Raw Interrupt Status 0 – ssi_rxu_intr interrupt is not active prior to masking 1 – ssi_rxu_intr interrupt is active prior to masking |
| 1 | R | TXOIR | 0 | Transmit FIFO Overflow Raw Interrupt Status 0 – ssi_txo_intr interrupt is not active prior to masking 1 – ssi_txo_intr interrupt is active prior masking |
| 0 | R | TXEIR | 0 | Transmit FIFO Empty Raw Interrupt Status 0 – ssi_txe_intr interrupt is not active prior to masking 1 – ssi_txe_intr interrupt is active prior masking |

982 . RAW Interrupt Status Register Description

SSIM TRANSMIT FIFO OVERFLOW INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | TXOICR | 0 | 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. |

983 . Transmit FIFO Overflow Interrupt Clear Register Description

SSIM RECEIVE FIFO OVERFLOW INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | RXOICR | 0 | 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. |

984 . Receive FIFO Overflow Interrupt Clear Register Description

SSIM RECEIVE FIFO UNDERFLOW INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | RXUICR | 0 | 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. |

985 . Receive FIFO Underflow Interrupt Clear Register Description

SSIM MULTI_MASTER INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | MSTICR | 0 | Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect. |

986 . Multi-Master Interrupt Clear Register Description

SSIM INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | ICR | 0 | 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. |

987 . Interrupt Clear Register Description

SSIM DMA CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:2 | N/A | Reserved | 0 | Reserved |
| 1 | R/W | TDMAE | 0 | Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 – Transmit DMA disabled 1 – Transmit DMA enabled |
| 0 | R/W | RDMAE | 0 | Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 – Receive DMA disabled 1 – Receive DMA enabled |

988 . DMA Control Register Description

SSIM DMA TRANSMIT DATA LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 3:0 | R/W | DMATDL | 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 <code>dma_tx_req</code> 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. |

989 . DMA Transmit Data Level Register Description

SSIM DMA RECEIVE DATA LEVEL REGISTER

| Bi t | Acc ess | Fun ctio n | POR Valu e | Description |
|---------|------------|------------------|------------------|---|
| 31:4 | N/A | Reserv ed | 0 | Reserved |
| 3:0 | R/W | DMARDL | 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, <code>dma_rx_req</code> 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. |

990 . DMA Receive Data Level Register Description

SSIM IDENTIFICATION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | R | IDCODE | 0xFFFF_FFFF | Identification Code. This register contains the peripherals identification code. |

991 . Identification Register Description

SSIM COMPONENT VERSION

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------|------------|---|
| 31:0 | R | SSI_COMP_VERSIO N | 0x3430302a | Contains the hex representation of the component version. |

992 . SSI Component Version Register Description

SSIM DATA REGISTERS

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:0 | R/W | DR | 0 | Data Register. When writing to this register, the user must right-justify the data. Read data are automatically right-justified. Read – Receive FIFO buffer Write – Transmit FIFO buffer |

993 . Data Registers Description

SSIM RXD SAMPLE DELAY REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:8 | N/A | Reserved | | Reserved |
| 7:0 | R/W | RSD | 0 | Receive Data (rx) Sample Delay. This register is used to delay the sample of the rx input signal. Each value represents a single ssi_clk delay on the sample of the rx signal. NOTE: If this register is programmed with a value that exceeds the depth of the internal shift registers (63), a zero (0) delay will be applied to the rx sample. |

994 . RXD Sample Delay Register Description

SSIM SPI CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------|-----------|--|
| 31:15 | R | Reserved | 0 | Reserved |
| 14:11 | R/W | WAIT_CYCLES | 0 | <p>This bit defines the wait cycles in dual/quad mode between control frames transmit and data reception. Specified as number of SPI clock cycles.</p> <p>0000 – No Wait Cycles 0001 – 1 Wait Cycle up to 1111 – 15 Wait Cycles</p> |
| 10 | R | Reserved | 0 | Reserved |
| 9:8 | R/W | INST_L | 0x2 | <p>Dual/Quad mode instruction length in bits.</p> <ul style="list-style-type: none"> • 00 - 0 bit (No instruction) • 01 - 4 bits • 10 - 8 bits • 11 - 16 bits |
| 7:6 | R | Reserved | 0 | Reserved |
| 5:2 | R/W | ADDR_L | 0 | This bit defines length of address to be transmitted. The transfer begins only after these many bits are programmed into the FIFO. |
| 1:0 | R/W | TRANS_TYPE | 0 | <p>Address and instruction transfer format.</p> <p>This bit selects whether ssi will transmit instruction/address either in Standard SPI mode or the SPI mode when the mode is specified in the CTRLR0.SPI_FRF field.</p> <ul style="list-style-type: none"> • 00 - Instruction and Address will be sent in Standard SPI Mode. • 01 - Instruction will be sent in Standard SPI Mode and Address will be sent in the mode specified by CTRLR0.SPI_FRF. • 10 - Both Instruction and Address will be sent in the mode specified by CTRLR0.SPI_FRF. • 11 - Reserved. |

995 . SPI Control Register Description

16.16 SSI Slave

16.16.1 General Description

The SSI slave is a programmable synchronous serial(SSI) peripheral. The SSI slave can connect to any serial-master peripheral using one of the following interfaces:

- Motorola Serial Peripheral Interface (SPI)
- Texas Instruments Serial Protocol (SSP)
- National Semiconductor Microwire.

16.16.2 Features

- Acts as a Serial slave
- DMA handshake present
- Independent masking of interrupts - Master collision, transmit FIFO overflow, transmit FIFO empty, receive FIFO full, receive FIFO underflow, and receive FIFO overflow interrupts can all be masked independently.
- Works with Motorola SPI, Texas Instruments SSP and National Semiconductors Microwire
- Data Item size (4 to 16 bits) – Item size of each data transfer under the control of the programmer.
- Supported FIFO depth is 16(Independent TX and RX FIFOs are present)
- Combined interrupt line for all interrupts
- Generates active high interrupts
- APB clock(pclk) and SSI Slave serial clock are identical

16.16.3 Functional Description

The MCU accesses data, control, and status information on the SSI slave through the APB interface. This may also interface with a DMA Controller using an optional set of DMA signals. The SSI slave can connect to any serial-master peripheral using one of the following interfaces:

- Motorola Serial Peripheral Interface (SPI): A four-wire, full-duplex serial protocol from Motorola. There are four possible combinations for the serial clock phase and polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock. The slave select line is held high when the SSI master is idle or disabled.
- Texas Instruments Serial Protocol (SSP): A four-wire, full-duplex serial protocol. The slave select line is used for SPI and Microwire protocols doubles as the frame indicator for the SSP protocol.
- National Semiconductor Microwire : A half-duplex serial protocol, which uses a control word transmitted from the serial master to the target serial slave.

FRF (frame format) bit field in the Control Register 0 (CTRLR0) is used to select which protocol is used. Frequency ratio restrictions between the bit-rate clock (sclk_in) and the

SSI slave peripheral clock (ssi_clk) is as follows

$$F_{ssi_clk} \leq 4 \times (\text{maximum } F_{sclk_in})$$

16.16.4 SSI Interrupts

The SSI slave supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ORed result of all other interrupts after masking. All SSI slave interrupts are level interrupts and have the same active polarity level. This polarity level can be configured as active-high or active-low. The SSI slave 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 APB access attempts to write into the transmit FIFO after it has been completely filled. When set, data written from the APB is discarded. This interrupt remains set until reading of 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.

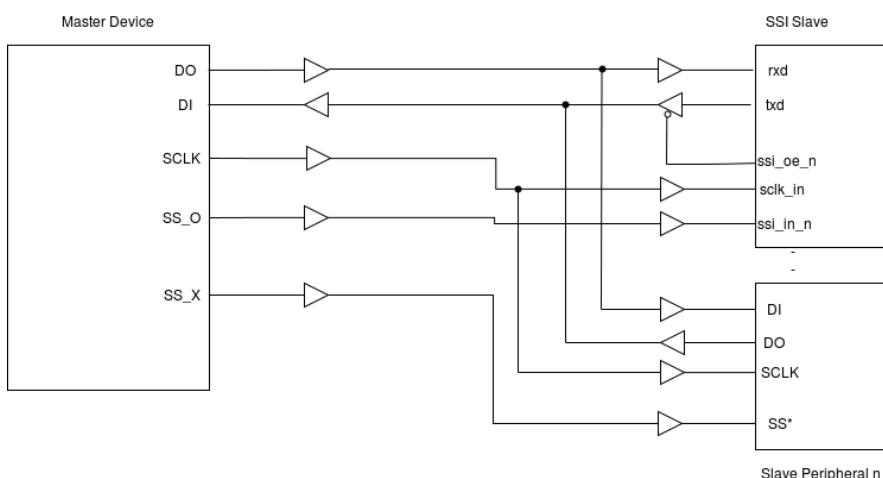
- Receive FIFO Overflow Interrupt (ssi_rxo_intr) – Set when the receive logic attempts to place data into the receive FIFO after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until reading of the receive FIFO overflow interrupt clear register (RXOICR).
- Receive FIFO Underflow Interrupt (ssi_rxu_intr) – Set when an APB 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 reading of 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, mask all other SSI interrupt requests.

16.16.5 Programming Sequence for data transfer

This mode enables serial communication with master peripheral devices.

All serial transfers are initiated and controlled by the serial bus master. Figure 1 shows an example of the SSI is configured as a serial slave in a single-master bus system.

All data transfers to and from the serial slave are regulated on the serial clock line (sclk_in), driven from the serial master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge. The slave remains in an idle state until selected by the bus master. When not actively transmitting data, the slave must hold its txd line in a high impedance state to avoid interference with serial transfers to other slave devices.



SSI slave in a single-master bus system

Slave SPI and SSP Serial Transfers

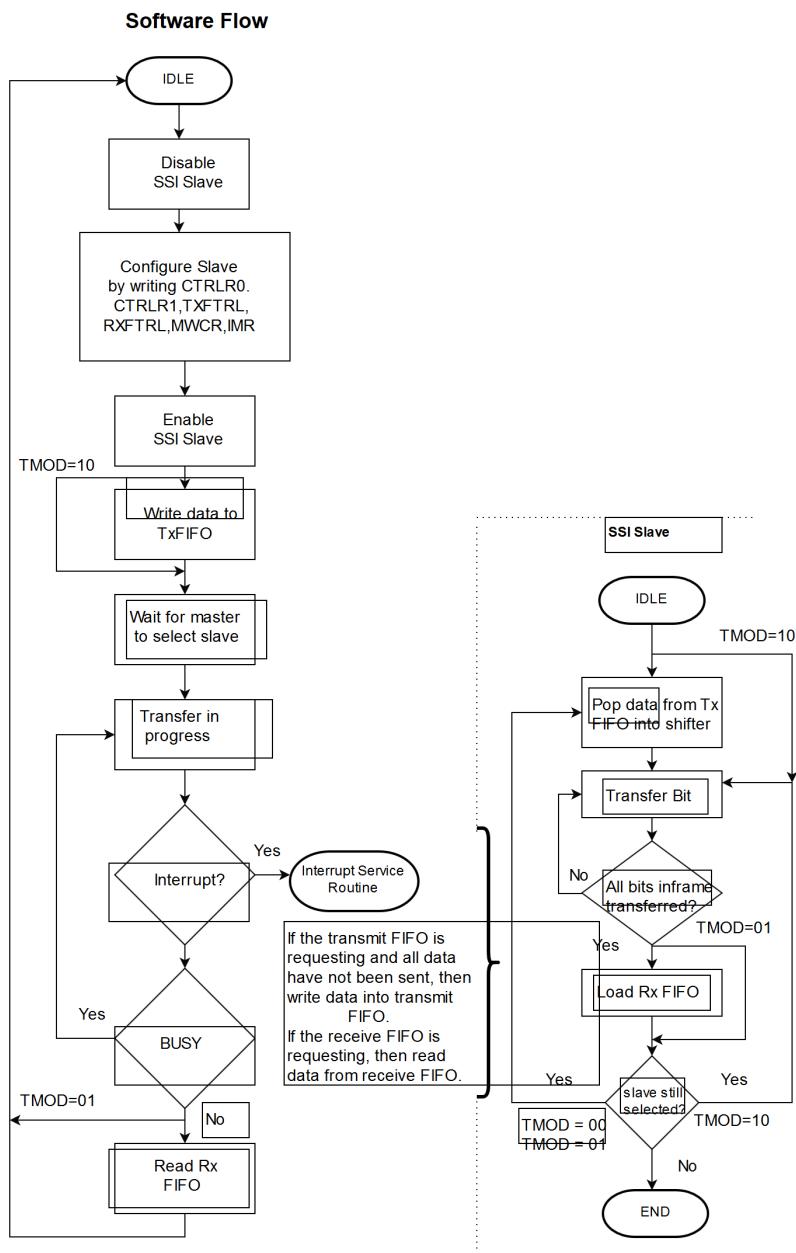
A typical software flow for completing a continuous serial transfer from a serial master to the SSI slave is described as follows:

1. If the SSI slave is enabled, disable it by writing 0 to SSIENR.
2. Set up the SSI Slave control registers for the transfer. These registers can be set in any order.
 - a. Write CTRLR0 (for SPI transfers SCPH and SCPOL must be set identical to the master device).
 - b. Write TXFTLR and RXFTLR to set FIFO threshold levels.
 - c. Write the IMR register to set up interrupt masks.
3. Enable the SSI by writing 1 to the SSIENR register.
4. If the transfer mode is transmit and receive (TMOD=2'b00) or transmit only (TMOD=2'b01), write data for transmission to the master into the transmit FIFO (Write DR).

If the transfer mode is receive only (TMOD=2'b10), there is no need to write data into the transmit FIFO; the current value in the transmit shift register is re-transmitted.

5. The SSI slave is now ready for the serial transfer. The transfer begins when the SSI slave is selected by a serial-master device.
6. When the transfer is underway, the BUSY status can be polled to return the transfer status. If a transmit FIFO empty interrupt request is made, write the transmit FIFO (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO (read DR).
7. The transfer ends when the serial master removes the select input to the SSI slave. When the transfer is completed, the BUSY status is reset to 0.
8. If the transfer mode is not transmit only (TMOD != 01), read the receive FIFO until empty.
9. Disable the SSI by writing 0 to SSIENR.

Figure 2 shows a typical software flow for a SSI slave SPI or SSP serial transfer.



SSI slave SPI/SSP Transfer Flow

Slave Microwire Serial Transfers

In SSI slave mode, the Microwire protocol operates in much the same way as the SPI protocol. There is no decode of the control frame by the SSI slave device.

16.16.6 Register Summary

Base Address: 0x4501_0000

| Register Name | Offset | Description |
|-------------------|-----------|---|
| SSIS_CTRLR0 | 0x00 | Control Register 0 |
| SSIS_ENR | 0x08 | SSI Enable Register |
| SSIS_MWCR | 0x0C | Microwire Control Register |
| SSIS_TXFTLR | 0x18 | Transmit FIFO Threshold Level Register |
| SSIS_RXFTLR | 0x1C | Receive FIFO Threshold Level Register |
| SSIS_TXFLR | 0x20 | Transmit FIFO Level Register |
| SSIS_RXFLR | 0x24 | Receive FIFO Level Register |
| SSIS_SR | 0x28 | Status Register |
| SSIS_IMR | 0x2C | Interrupt Mask Register |
| SSIS_ISR | 0x30 | Interrupt Status Register |
| SSIS_RISR | 0x34 | Raw Interrupt Status Register |
| SSIS_TXOICR | 0x38 | Transmit FIFO Overflow Interrupt Clear Register |
| SSIS_RXOICR | 0x3C | Receive FIFO Overflow Interrupt Clear Register |
| SSIS_RXUICR | 0x40 | Receive FIFO Underflow Interrupt Clear Register |
| SSIS_MSTICR | 0x44 | Multi-Master Interrupt Clear Register |
| SSIS_ICR | 0x48 | Interrupt Clear Register |
| SSIS_DMACR | 0x4C | DMA Control Register |
| SSIS_DMATDLR | 0x50 | DMA Transmit Data Level Register |
| SSIS_DMARDLR | 0x54 | DMA Receive Data Level Register |
| SSIS_IDR | 0x58 | Identification Register |
| SSIS_COMP_VERSION | 0x5C | SSI Component Version |
| SSIS_DR | 0x60-0xEC | Data Registers |

996 . Register Summary Table

16.16.7 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, N/A = Reserved

SSI SLAVE CONTROL REGISTER 0

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|--|
| 15:12 | R/W | CFS | 0x0 | Control Frame Size. Selects the length of the control word for the Microwire frame format. Ranges : 0000 -> 1 bit word control to 1111 -> 16 bit word control |
| 11 | R/W | SRL | 0 | Shift Register Loop used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. 0 – Normal Mode Operation 1 – Test Mode Operation Note : When SSI slave is configured in loop back mode, the ss_in_n and ssi_clk signals must be provided by an external source. |
| 10 | R/W | SLV_OE | 0 | Slave Output Enable. This bit enables or disables the setting of the ssi_oe_n output from the SSI Slave serial slave. 0 -- Slave txd is enabled 1 -- Slave txd is disabled |
| 9:8 | R/W | TMOD | 0 | Transfer Mode. Selects the mode of transfer for serial communication. 00 -- Transmit & Receive 01 -- Transmit Only 10 -- Receive Only |
| 7 | R/W | SCPOL | 0 | Serial Clock Polarity. Valid when the frame format (FRF) is set to Motorola SPI. 0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high |
| 6 | R/W | SCPH | 0 | Serial Clock Phase. Valid when the frame format (FRF) is set to Motorola SPI. 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. 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 5:4 | R/W | FRF | 0 | Frame Format. Selects which serial protocol transfers the data. 00 -- Motorola SPI 01 -- Texas Instruments SSP 10 -- National Semiconductors Microwire 11 -- Reserved |
| 3:0 | R/W | DFS | 0x7 | Data Frame Size. Selects the data frame length. Range : 0011 -> 4 bit to 1111 -> 16 bit |

997 . Control Register 0 Description

SSI SLAVE ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R/W | SSI_EN | 0x0 | SSI Enable. Enables and disables all SSI Slave operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. |

998 . SSI Enable Register Description

SSI SLAVE MICROWIRE CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 2 | N/A | Reserved | 0 | Reserved |
| 1 | R/W | MDD | 0 | Microwire Control. 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. |
| 0 | R/W | MWMOD | 0 | Microwire Transfer Mode. 0 – non-sequential transfer 1 – sequential transfer 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. |

999 . Microwire Control Register Description

SSI SLAVE TRANSMIT FIFO THRESHOLD LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 3:0 | R/W | TFT | 0 | <p>Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.</p> <p>If this field is set to a value greater than or equal to the depth of the FIFO, this field is not written and retains its current value.</p> <p>When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.</p> |

1000 . Transmit FIFO Threshold Level Register Description

SSI SLAVE RECEIVE FIFO THRESHOLD LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|---|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 3:0 | R/W | RFT | 0 | <p>Receive FIFO Threshold. Controls the level of entries (or below) at which the receive FIFO controller triggers an interrupt.</p> <p>If this field is set to a value greater than or equal to the depth of the FIFO, this field is not written and retains its current value.</p> <p>When the number of receive FIFO entries is less than or equal to this value + 1, the receive FIFO full interrupt is triggered.</p> |

1001 . Receive FIFO Threshold Level Register

SSI SLAVE TRANSMIT FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-----------------|------------------|--|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 4:0 | R | TXTFL | 0 | Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO. |

1002 . Transmit FIFO Level Register Description

SSI SLAVE RECEIVE FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 4:0 | R | RXTFL | 0 | Receive FIFO Level. Contains the number of valid data entries in the receive FIFO. |

1003 . Receive FIFO Level Register Description

SSI SLAVE STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 6 | N/A | Reserved | 0 | Reserved |
| 5 | R | TXE | 0 | Transmission Error. This bit is cleared when read. 0 – No error 1 – Transmission error |
| 4 | R | RFF | 0 | Receive FIFO Full. 0 – Receive FIFO is not full 1 – Receive FIFO is full |
| 3 | R | RFNE | 0 | Receive FIFO Not Empty. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty This bit can be polled by software to completely empty the receive FIFO. |
| 2 | R | TFE | 1 | Transmit FIFO Empty. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty This bit field does not request an interrupt. |
| 1 | R | TFNF | 1 | Transmit FIFO Not Full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full |
| 0 | R | BUSY | 0 | SSI Busy Flag. 0 – SSI is idle or disabled 1 – SSI is actively transferring data |

1004 . Status Register Description

SSI SLAVE INTERRUPT MASK REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:5 | N/A | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 4 | R/W | RXFIM | 1 | Receive FIFO Full Interrupt Mask 0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked |
| 3 | R/W | RXOIM | 1 | Receive FIFO Overflow Interrupt Mask 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked |
| 2 | R/W | RXUIM | 1 | Receive FIFO Underflow Interrupt Mask 0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked |
| 1 | R/W | TXOIM | 1 | Transmit FIFO Overflow Interrupt Mask 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked |
| 0 | R/W | TXEIM | 1 | Transmit FIFO Empty Interrupt Mask 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked |

1005 . Interrupt Mask Register Description

SSI SLAVE INTERRUPT STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:5 | N/A | Reserved | 0 | Reserved |
| 4 | R | RXFIS | 0 | Receive FIFO Full Interrupt Status 0 – ssi_rxf_intr interrupt is not active after masking 1 – ssi_rxf_intr interrupt is full after masking |
| 3 | R | RXOIS | 0 | Receive FIFO Overflow Interrupt Status 0 – ssi_rxo_intr interrupt is not active after masking 1 – ssi_rxo_intr interrupt is active after masking |
| 2 | R | RXUIS | 0 | Receive FIFO Underflow Interrupt Status 0 – ssi_rxu_intr interrupt is not active after masking 1 – ssi_rxu_intr interrupt is active after masking |
| 1 | R | TXOIS | 0 | Transmit FIFO Overflow Interrupt Status 0 – ssi_txo_intr interrupt is not active after masking 1 – ssi_txo_intr interrupt is active after masking |
| 0 | R | TXEIS | 0 | Transmit FIFO Empty Interrupt Status 0 – ssi_txe_intr interrupt is not active after masking 1 – ssi_txe_intr interrupt is active after masking |

1006 . Interrupt Status Register Description

SSI SLAVE RAW INTERRUPT STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:5 | N/A | Reserved | 0 | Reserved |
| 4 | R | RXFIR | 0 | Receive FIFO Full Raw Interrupt Status 0 – ssi_rxf_intr interrupt is not active prior to masking 1 – ssi_rxf_intr interrupt is active prior to masking |
| 3 | R | RXOIR | 0 | Receive FIFO Overflow Raw Interrupt Status 0 – ssi_rxo_intr interrupt is not active prior to masking 1 – ssi_rxo_intr interrupt is active prior to masking |
| 2 | R | RXUIR | 0 | Receive FIFO Underflow Raw Interrupt Status 0 – ssi_rxu_intr interrupt is not active prior to masking 1 – ssi_rxu_intr interrupt is active prior to masking |
| 1 | R | TXOIR | 0 | Transmit FIFO Overflow Raw Interrupt Status 0 – ssi_txo_intr interrupt is not active prior to masking 1 – ssi_txo_intr interrupt is active prior to masking |
| 0 | R | TXEIR | 0 | Transmit FIFO Empty Raw Interrupt Status 0 – ssi_txe_intr interrupt is not active prior to masking 1 – ssi_txe_intr interrupt is active prior to masking |

1007 . RAW Interrupt Status Register Description

SSI SLAVE TRANSMIT FIFO OVERFLOW INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | TXOICR | 0 | 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. |

1008 . Transmit FIFO Overflow Interrupt Clear Register Description

SSI SLAVE RECEIVE FIFO OVERFLOW INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | RXOICR | 0 | 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. |

1009 . Receive FIFO Overflow Interrupt Clear Register Description

SSI SLAVE RECEIVE FIFO UNDERFLOW INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | RXUICR | 0 | 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. |

1010 . Receive FIFO Underflow Interrupt Clear Register Description

SSI SLAVE MULTI_MASTER INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | MSTICR | 0 | Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect. |

1011 . Multi-Master Interrupt Clear Register Description

SSI SLAVE INTERRUPT CLEAR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | N/A | Reserved | 0 | Reserved |
| 0 | R | ICR | 0 | 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. |

1012 . Interrupt Clear Register Description

SSI SLAVE DMA CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:2 | N/A | Reserved | 0 | Reserved |
| 1 | R/W | TDMAE | 0 | Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 – Transmit DMA disabled 1 – Transmit DMA enabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 0 | R/W | RDMAE | 0 | Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 – Receive DMA disabled 1 – Receive DMA enabled |

1013 . DMA Control Register Description

SSI SLAVE DMA TRANSMIT DATA LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 3:0 | R/W | DMATDL | 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. |

1014 . DMA Transmit Data Level Register Description

SSI SLAVE DMA RECEIVE DATA LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:4 | N/A | Reserved | 0 | Reserved |
| 3:0 | R/W | DMARDL | 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. |

1015 . DMA Receive Data Level Register Description

SSI SLAVE IDENTIFICATION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | R | IDCODE | 0xFFFF_FFFF | Identification Code. This register contains the peripherals identification code. |

1016 . Identification Register Description

SSI SLAVE COMPONENT VERSION

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|------------|---|
| 31:0 | R | SSI_COMP_VERSIO N | 0x3430302a | Contains the hex representation of the component version. |

1017 . SSI Component Version Register Description

SSI SLAVE DATA REGISTERS

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 15:0 | R/W | DR | 0 | Data Register. When writing to this register, the user must right-justify the data. Read data are automatically right-justified. Read – Receive FIFO buffer Write – Transmit FIFO buffer |

1018 . Data Registers Description

16.17 UART

16.17.1 General Description

There are three UART controllers - two in the MCU HP peripherals (USART1, UART2) and one in the MCU ULP subsystem (ULP_UART). One of the controllers in the MCU HP peripherals (USART1) supports USART.

The UART is used for serial communication with:

- Peripherals
- Modems (data carrier equipment, DCE)
- Data sets

16.17.2 Features

Each of these UART controllers support the following features:

- Programmer interface compatible with the 16450
- 9-bit serial data support
- Multi-drop RS485 interface support
- 5, 6, 7 and 8-bit character encoding with Even, Odd and No Parity
- 1, 1.5 (only with 5 bit character encoding) and 2 stop bits
- Hardware Auto flow control (RTS/CTS)
- IrDA 1.0 SIR mode support (only for UART2) with up to 115.2 K baud data rate and a programmable pulse duration and low-power reception capabilities
- Programmable baud rate as calculated by the following: baud rate = (serial clock frequency)/(16× divisor). Maximum supported baud rate is 7,372,800 (around 7.3Mbps) with 118MHz UART input clock.
- Supported standard baud rates are 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1843200, 3686400, 7372800.

The UART controllers also support additional features listed below, which help in achieving better performance and reduce the burden on the processor:

- Programmable fractional baud rate support
- Programmable baud rate supporting upto 7.3Mbps

- Programmable FIFO thresholds with maximum FIFO depth of 16 and support for DMA
- Prioritized interrupt identification

The UART controller in the MCU ULP subsystem (ULP_UART) supports the following additional power-save features:

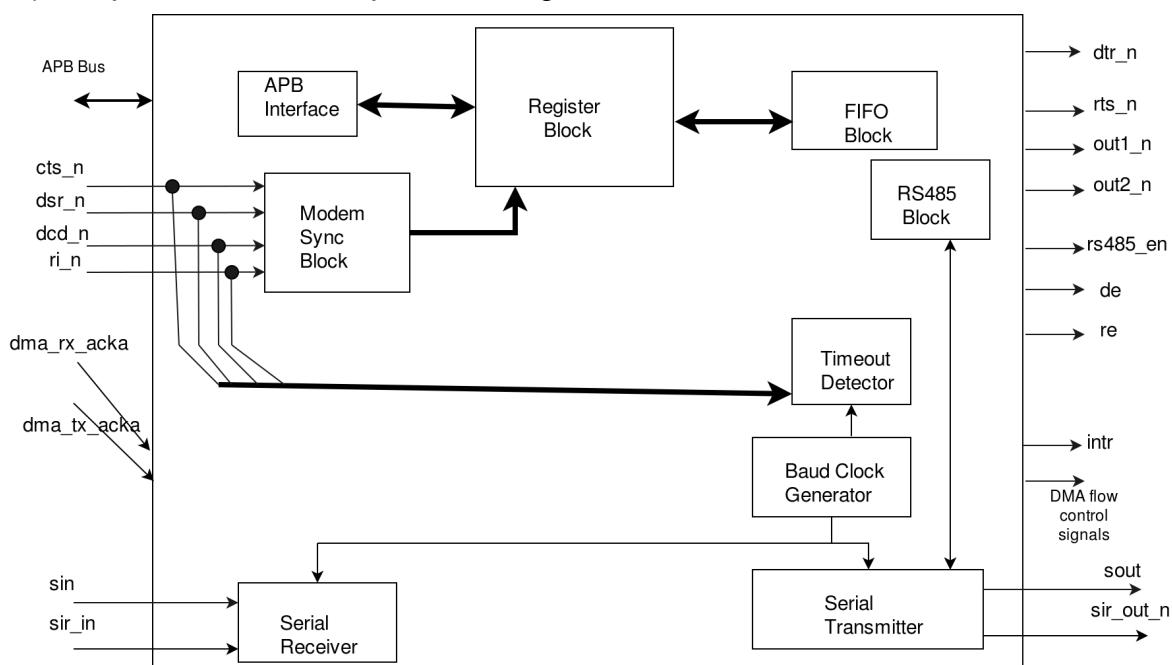
- After the DMA is programmed in PS2 state for UART transfers, the MCU can switch to PS1 state (processor is shutdown) while the UART controller continues with the data transfer
- In PS1 state (ULP Peripheral mode) the UART controller completes the data transfer and, triggered by the Peripheral Interrupt, shifts either to the sleep state (without processor intervention) or the active state

16.17.3 Functional Description

Figure 1 illustrates the functional block diagram. It contains APB interface, Register block, Modem sync block, Baud rate generator, Serial transmitter and serial receiver. The UART contains registers that control:

- Character length
- Baud rate
- Parity generation/checking
- Interrupt generation

Although there is only one interrupt output signal (intr) from the UART, there are several prioritized interrupt types that can be responsible for its assertion. Each of the interrupt types can be separately enabled or disabled by the control registers.



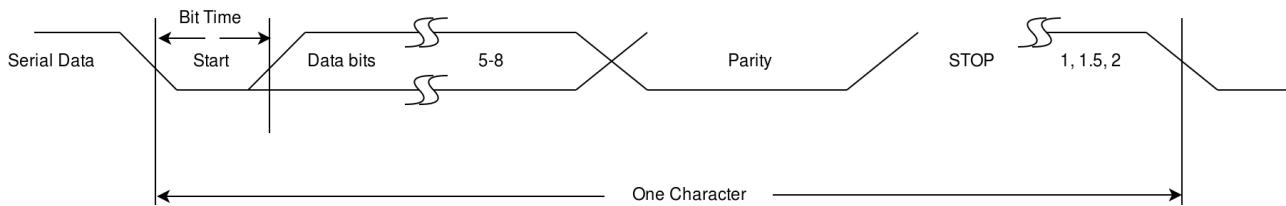
UART Functional Block Diagram

Because the serial communication between the UART and a selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Utilizing these bits allows two devices to be synchronized. This structure of serial data accompanied by start and stop bits is referred to as a character, as shown in Figure 2.

An additional parity bit can be added to the serial character. This bit appears after the last data bit and before the stop bits, which can be 1, 1.5, or 2 bits, in the character structure in order to provide the UART with the ability to perform simple error checking on the received data.

The UART Line Control Register (LCR) is used to control the serial character characteristics.

To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected. Because the exact number of baud clocks is known for which each bit was transmitted, calculating the midpoint for sampling is not difficult; that is, every sixteen baud clocks after the midpoint sample of the start bit.



UART Serial Data Format

16.17.4 UART Transmission

The UART can be configured to have 9-bit data transfer in both transmit and receive mode. The 9th bit in the character appears after the 8th bit and before the parity bit in the character. By enabling 9-bit data transfer mode, UART can be used in multi-drop systems where one master is connected to multiple slaves in a system. The master communicates with one of the slaves. When the master wants to transfer a block of data to a slave, it first sends an address byte to identify the target slave. The address byte is differentiated from the data byte by setting the 9th bit of the data byte to 1. If the 9th bit is set to 0, then the character represents an address byte. All the slave systems compare the address byte with their own address and only the target slave (in which the address has matched) is enabled to receive data from the master. The master then starts transmitting data bytes to the target slave. The non-addressed slave systems ignore the incoming data until a new address byte is received.

Configuration of the UART for 9-bit data transfer does the following:

- LCR_EXT[0] bit is used to enable or disable the 9-bit data transfer.
- LCR_EXT[1] bit is used to choose between hardware and software based address match in the case of receive.
- LCR_EXT[2] bit is used to enable to send the address in the case of transmit.
- LCR_EXT[3] bit is used to choose between hardware and software based address transmission.
- TAR and RAR registers are used to transmit address and to match the received address, respectively.
- THR, RBR, STHR and SRBR registers are of 9-bit which is used to do the data transfers in 9-bit mode.
- LSR[8] bit is used to indicate the address received interrupt.

Transmit Modes

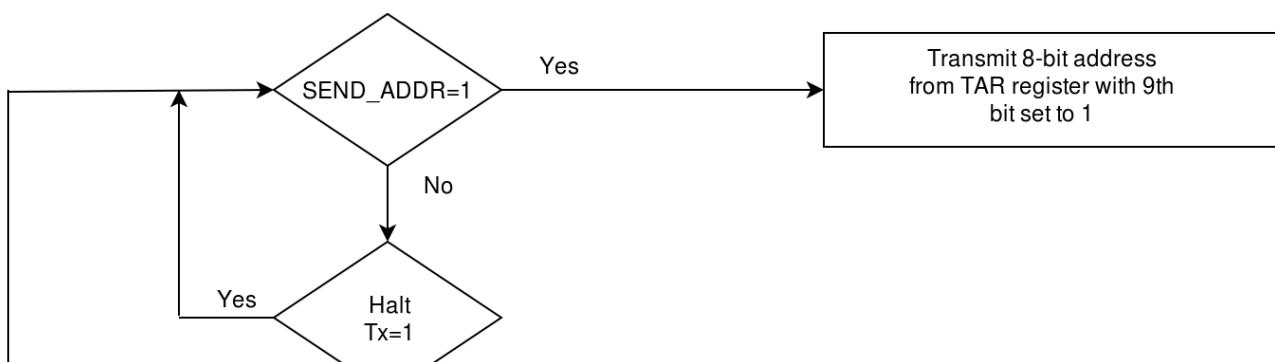
UART supports two types of transmit modes:

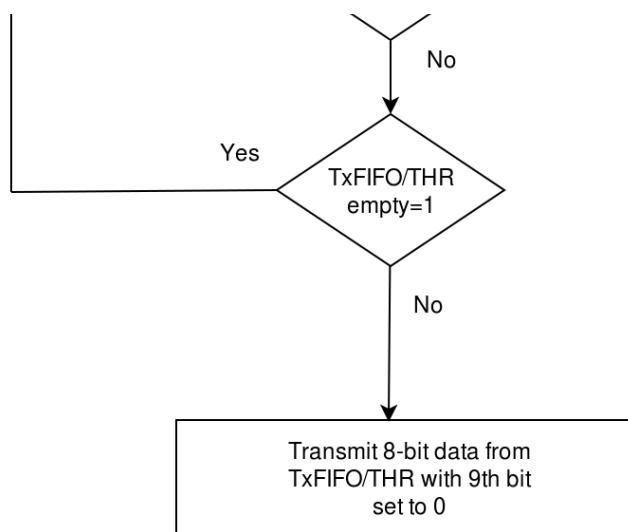
Transmit Mode 0 (when (LCR_EXT[3]) is set to 0)

Transmit Mode 1 (when (LCR_EXT[3]) is set to 1)

Transmit Mode 0

In transmit mode 0, the address is programmed in the Transmit Address Register (TAR) register and data is written into the Transmit Holding Register (THR) or the Shadow Transmit Holding Register (STHR). The 9th bit of the THR and STHR register is not applicable in this mode. Figure 3 illustrates the transmission of address and data based on SEND_ADDR (LCR_EXT[2]), Halt Tx, and TxFIFO/THR empty conditions.





UART Auto Address Transmit Flow Chart

The address of the target slave to which the data is to be transmitted is programmed in the TAR register. Must enable the SEND_ADDR (LCR_EXT[2]) bit to transmit the target slave address present in the TAR register on the serial UART line with 9th data bit set to 1 to indicate that the address is being sent to the slave. The UART clears the SEND_ADDR bit after the address character starts transmitting on the UART line. The data required to transmit to the target slave is programmed through Transmit Holding Register (THR). The data is transmitted on the UART line with 9th data bit set to 0 to indicate data is being sent to the slave. If the application is required to fill the data bytes in the Tx FIFO before sending the address on the UART line (before setting LCR_EXT[2]=1), then it is recommended to set the "Halt Tx" to 1 such that UART does not start sending out the data in the Tx FIFO as data byte. Once the Tx FIFO is filled, then program SEND_ADDR (LCR_EXT[2]) to 1 and then set "Halt Tx" to 0.

Transmit Mode 1

In transmit mode 1, THR and STHR registers are of 9-bit wide and both address and data are programmed through the THR and STHR registers. The UART does not differentiate between address and data, and both are taken from the Tx FIFO. The SEND_ADDR (LCR_EXT[2]) bit and Transmit address register (TAR) are not applicable in this mode. The software must pack the 9th bit with 1/0 depending on whether address/ data has to be sent.

16.17.5 UART Reception

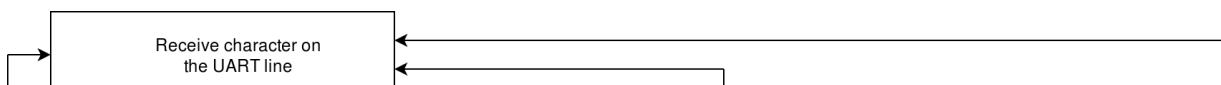
Receive Modes

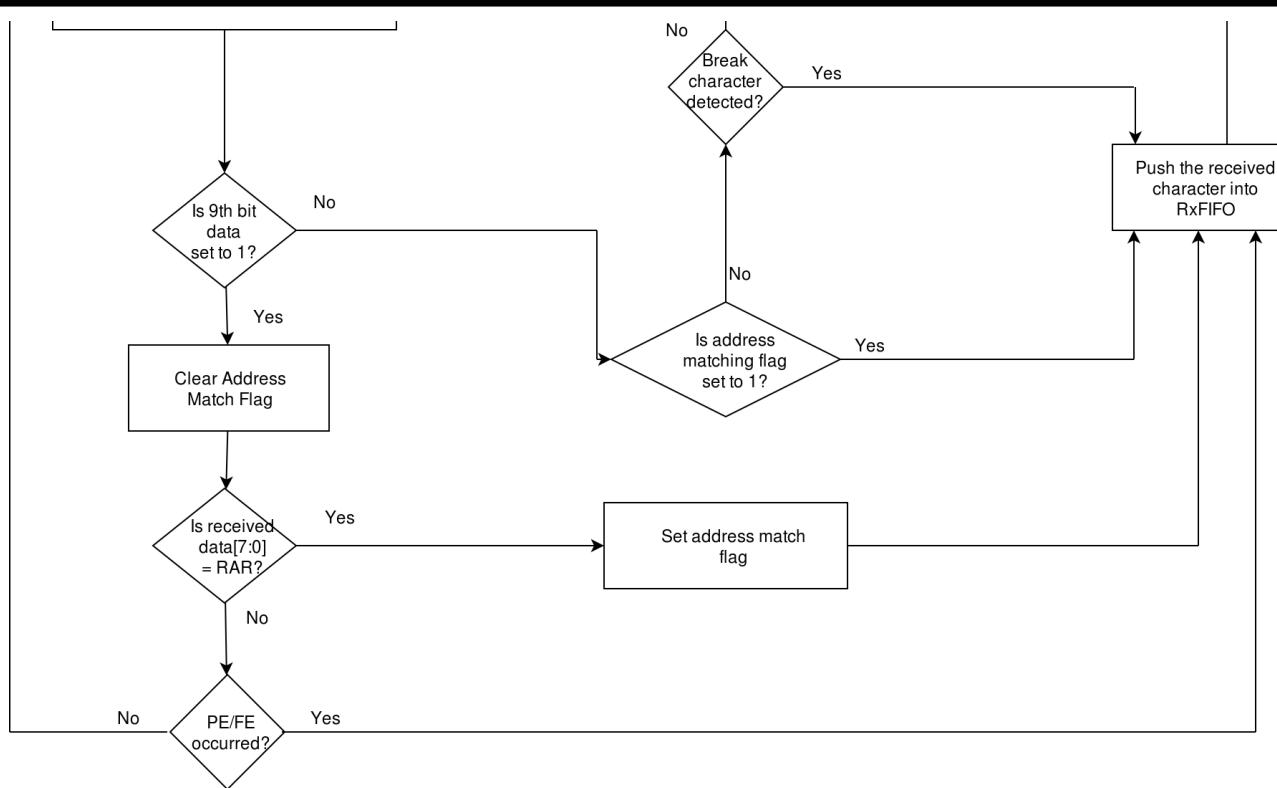
The UART supports two receive modes:

- Hardware Address Match Receive Mode (when ADDR_MATCH (LCR_EXT[1]) is set to 1)
- Software Address Match Receive Mode (when ADDR_MATCH (LCR_EXT[1]) is set to 0)

Hardware Address Match Receive Mode

In the hardware address match receive mode, the UART matches the received character with the address programmed in the Receive Address register (RAR), if the 9th bit of the received character is set to 1. If the received address is matched with the programmed address in RAR register, then subsequent data bytes (with 9th bit set to 0) are pushed into the Rx FIFO. If the address matching fails, then UART controller discards further data characters until a matching address is received. Figure 4 illustrates the flow chart for the reception of data bytes based on the address matching feature.





UART Hardware Address Match Receive Mode

UART receives the character irrespective of whether the 9th bit data is set to 1. If 9th bit of the received character is set to 1, then it clears internal address match flag and then compares the received 8-bit character information with the address programmed in the RAR register.

If the received address character matches with the address programmed in the RAR register, then the address match flag is set to 1 and the received character is pushed to the Rx FIFO in FIFO-mode or to RBR register in non-FIFO mode and the ADDR_RCVD bit in LSR register is set to indicate that the address has been received.

In case of parity or if a framing error is found in the received address character and if the address is not matched with the RAR register, then the received address character is still pushed to Rx FIFO or RBR register with ADDR_RCVD and PE/FE error bit set to 1.

The subsequent data bytes (9th bit of received character is set to 0) are pushed to the Rx_FIFO in FIFO mode or to the RBR register in non-FIFO mode until the new address character is received.

If any break character is received, UART treats it as a special character and pushes to the Rx FIFO or RBR register based on the FIFO_MODE irrespective of address match flag.

Software Address Match Receive Mode

In this mode of operation, the UART does not perform the address matching for the received address character (9th bit data set to 1) with the RAR register. The UART always receives the 9-bit data and pushes in to Rx FIFO in FIFO mode or to the RBR register in non-FIFO mode. The user must compare the address whenever address byte is received and indicated through ADDR_RCVD bit in the Line Status register. The user can flush/reset the Rx FIFO in case of address not matched through 'RCVR FIFO Reset' bit in FIFO control register (FCR).

16.17.6 Interrupts

Assertion of the UART interrupt output signal (intr) a positive-level interrupt occurs whenever one of the several prioritized interrupt types are enabled and active.

When an interrupt occurs, the master accesses the IIR register.

The following interrupt types can be enabled with the IER register:

- Receiver Error

- Receiver Data Available
- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THRE interrupt mode)
- Modem Status
- Busy Detect Indication

16.17.7 Auto Flow Control

The UART can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available; if FIFOs are not implemented, this mode cannot be selected. When Auto Flow Control is not selected, none of the corresponding logic is implemented and the mode cannot be enabled, reducing overall gate counts. When Auto Flow Control mode is selected, it can be enabled with the Modem Control Register (MCR[5]). Auto RTS and Auto CTS are described as follows:

Auto RTS

It becomes active when the following occurs:

- RTS (MCR[1] bit and MCR[5]bit are both set)
- FIFOs are enabled (FCR[0]) bit is set)
- SIR mode is disabled (MCR[6] bit is not set)

When Auto RTS is enabled, the rts_n output is forced inactive (high) when the receiver FIFO level reaches the threshold set by FCR[7:6], but only if the RTC flow-control trigger is disabled. Otherwise, the rts_n output is forced inactive (high) when the FIFO is almost full, where “almost full” refers to two available slots in the FIFO. When rts_n is connected to the cts_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space; that is, until it is completely empty.

Once the receiver FIFO becomes completely empty by reading the Receiver Buffer Register (RBR), rts_n again becomes active (low), signalling the other UART to continue sending data.

Auto CTS

It becomes active when the following occurs:

- AFCE (MCR[5] bit = 1)
- FIFOs are enabled through FIFO Control Register FCR[0] bit
- SIR mode is disabled (MCR[6] bit = 0)

When Auto CTS is enabled (active), the UART transmitter is disabled whenever the cts_n input becomes inactive (high); this prevents overflowing the FIFO of the receiving UART.

If the cts_n input is not inactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, the transmitter FIFO can still be written to, and even overflowed. Therefore, when using this mode, the following happens:

- UART status register can be read to check if transmit FIFO is full (USR[1] set to 0)
- Current FIFO level can be read using TFL register
- Programmable THRE Interrupt mode must be enabled to access “FIFO full” status using Line Status Register (LSR)

When using the “FIFO full” status, software can poll this before each write to the Transmitter FIFO. When the cts_n input becomes active (low) again, transmission resumes.

16.17.8 Register Summary

UART Base Address: 0x4502_0000

ULP_UART Base Address: 0x2404_1800

| Register Name | Offset | Description |
|---------------|-----------|---|
| UART_RBR | 0x00 | Receive Buffer Register |
| UART THR | 0x00 | Transmit Holding Register |
| UART_DLL | 0x00 | Divisor Latch (Low) Register |
| UART_DLH | 0x00 | Divisor Latch (High) Register |
| UART_IER | 0x04 | Interrupt Enable Register |
| UART_IIR | 0x08 | Interrupt Identification Register |
| UART_FCR | 0x08 | FIFO Control Register |
| UART_LCR | 0x0C | Line Control Register |
| UART_MCR | 0x10 | Modem Control Register |
| UART_LSR | 0x14 | Line Status Register |
| UART_MSR | 0x18 | Modem Status Register |
| UART_SCR | 0x1C | Scratchpad Register |
| UART_LPDL | 0x20 | Low Power Divisor Latch (Low) Register |
| UART_LPDLH | 0x24 | Low Power Divisor Latch (High) Register |
| Reserved | 0x28-0x2C | - |
| UART_SRBR | 0x30-0x6C | Shadow Receive Buffer Register |
| UART_STHR | 0x30-0x6C | Shadow Transmit Holding Register |
| UART_FAR | 0x70 | FIFO Access Register |
| UART_TFR | 0x74 | Transmit FIFO Read Register |
| UART_RFW | 0x78 | Receive FIFO Write Register |
| UART_USR | 0x7C | UART Status Register |
| UART_TFL | 0x80 | Transmit FIFO Level Register |
| UART_RFL | 0x84 | Receive FIFO Level Register |
| UART_SRR | 0x88 | Software Reset Register |
| UART_SRTS | 0x8C | Shadow Request to Send Register |
| UART_SBCR | 0x90 | Shadow Break Control Register |
| UART_SDMAM | 0x94 | Shadow DMA Mode Register |
| UART_SFE | 0x98 | Shadow FIFO Enable Register |
| UART_SRT | 0x9C | Shadow RCVR Trigger Register |
| UART_STET | 0xA0 | Shadow TX Empty Trigger Register |
| UART_HTX | 0xA4 | Halt TX Register |
| UART_DMASA | 0xA8 | DMA Software Acknowledge Register |

| Register Name | Offset | Description |
|---------------|-----------|---|
| UART_TCR | 0xAC | Transceiver Control Register |
| UART_DE_EN | 0xB0 | Driver Output Enable Register |
| UART_RE_EN | 0xB4 | Receiver Output Enable Register |
| UART_DET | 0xB8 | Driver Output Enable Timing Register |
| UART_TAT | 0xBC | Turn Around Timing Register |
| UART_DLF | 0xC0 | Divisor Latch Fractional Value Register |
| UART_RAR | 0xC4 | Receive Address Register |
| UART_TAR | 0xC8 | Transmit Address Register |
| UART_LCR_EXT | 0xCC | Line Extended Control Register |
| Reserved | 0xD0-0xF0 | – |
| UART_CPR | 0xF4 | Component Parameter Register |
| UART_CV | 0xF8 | UART Component Version Register |
| UART_CTR | 0xFC | Component Type Register |

1019 . Register Summary Table

16.17.9 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write

UART RECEIVE BUFFER REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--|-----------|---|
| 31:9 | R | Reserved | 0 | Reserved |
| 8 | R | Receive Buffer register (MSB 9th bit) | 0 | Data byte received on the serial input port (sin) in UART mode for the MSB 9th bit. |

| Bit | Access | Function | POR Value | Description |
|------------|---------------|---|------------------|--|
| 7:0 | R | Receive Buffer Register (LSB 8 bits) | 0 | <p>Data byte received on the serial input port (sin) in UART mode, or the serial infrared input (sir_in) in infrared mode.</p> <p>The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LSR) is set.</p> <p>If in non-FIFO mode (FIFO_MODE = NONE) or FIFOs are disabled (FCR[0] set to 0), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error.</p> <p>If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to 1), this register accesses the head of the receive FIFO.</p> <p>If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.</p> |

1020 . Receive Buffer Register Description

UART TRANSMIT HOLDING REGISTER

| Bit | Access | Function | POR Value | Description |
|------------|---------------|-------------------|------------------|---|
| 31:9 | R | Reserved | 0 | Reserved |
| 8 | W | THR (MSB 9th bit) | 0 | Data to be transmitted on the serial output port (sout) in UART mode for the MSB 9th bit. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 7:0 | W | THR (LSB 8 bits) | 0 | <p>Data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode.</p> <p>Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set.</p> <p>If in non-FIFO mode or FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the THR clears the THRE.</p> <p>Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten.</p> <p>If in FIFO mode and FIFOs are enabled (FCR[0] = 1) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full.</p> <p>The number x (default=16) is determined by the value of FIFO Depth that is set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.</p> |

1021 . Transmit Holding Register Description

UART DIVISOR LATCH LOW REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | Divisor Latch (Low) | 0 | <p>Lower 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.</p> <p>The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> $\text{baud rate} = (\text{serial clock freq}) / (16 * \text{divisor}).$ <p>Note that with the Divisor Latch Registers (DLL and DLH) set to 0, the baud clock is disabled and no serial communications occur.</p> <p>Also, once the DLL is set, at least 8 clock cycles of the slowest uart clock should be allowed to pass before transmitting or receiving data.</p> |

1022 . Divisor Latch Low Register Description

UART DIVISOR LATCH HIGH REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|--|
| 7:0 | R/W | Divisor Latch (High) | 0 | <p>Upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.</p> <p>The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> $\text{baud rate} = (\text{serial clock freq}) / (16 * \text{divisor}).$ <p>Note that with the Divisor Latch Registers (DLL and DLH) set to 0, the baud clock is disabled and no serial communications occur.</p> <p>Also, once the DLH is set, at least 8 clock cycles of the slowest uart clock should be allowed to pass before transmitting or receiving data.</p> |

1023 .Divisor Latch High Register Description

UART INTERRUPT ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7 | R/W | PTIME | 0 | <p>Programmable THRE Interrupt Mode Enable. This is used to enable/disable the generation of THRE Interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |
| 6:4 | R | Reserved | 0 | Reserved |
| 3 | R/W | EDSSI | 0 | <p>Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |
| 2 | R/W | ELSI | 0 | <p>Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |
| 1 | R/W | ETBEI | 0 | <p>Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt.</p> <p>This is the third highest priority interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 0 | R/W | ERBFI | 0 | <p>Enable Received Data Available Interrupt. These are the second highest priority interrupts.</p> <p>This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFOs enabled).</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |

1024 . Interrupt Enable Register Description

UART INTERRUPT IDENTIFICATION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:6 | R | FIFOs Enabled (or FIFOSE) | 0 | <p>This is used to indicate whether the FIFOs are enabled or disabled.</p> <ul style="list-style-type: none"> • 00 – disabled • 11 – enabled |
| 5:4 | R | Reserved | 0 | Reserved |
| 3:0 | R | Interrupt ID (IID) | 1 | <p>This indicates the highest priority pending interrupt which can be one of the following types:</p> <ul style="list-style-type: none"> • 0000 – modem status • 0001 – no interrupt pending • 0010 – THR empty • 0100 – received data available • 0110 – receiver line status • 0111 – busy detect • 1100 – character timeout <p>Bit 3 indicates an interrupt can only occur when the FIFOs are enabled and used to distinguish a Character Timeout condition interrupt</p> |

1025 . Interrupt Identity Register Description

UART FIFO CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------------|-----------|--|
| 7:6 | W | RCVR Trigger (or RT) | 0 | <p>RCVR Trigger. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated.</p> <p>It also determines when the <code>dma_rx_req_n</code> signal is asserted in certain modes of operation.</p> <ul style="list-style-type: none"> • 00 – 1 character in the FIFO • 01 – FIFO ¼ full • 10 – FIFO ½ full • 11 – FIFO 2 less than full |
| 5:4 | W | TX Empty Trigger (or TET) | 0 | <p>TX Empty Trigger. This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active.</p> <p>It also determines when the <code>dma_tx_req_n</code> signal is asserted when in certain modes of operation. The following trigger levels are supported:</p> <ul style="list-style-type: none"> • 00 – FIFO empty • 01 – 2 characters in the FIFO • 10 – FIFO ¼ full • 11 – FIFO ½ full |
| 3 | W | DMA Mode (or DMAM) | 0 | <p>DMA Mode. This determines the DMA signalling mode used for the <code>dma_tx_req_n</code> and <code>dma_rx_req_n</code> output signals.</p> <ul style="list-style-type: none"> • 0 – mode 0 • 1 – mode 1 |
| 2 | W | XMIT FIFO Reset (or XFIFOR) | 0 | <p>XMIT FIFO Reset. This resets the control portion of the transmit FIFO and treats the FIFO as empty.</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |
| 1 | W | RCVR FIFO Reset (or RFIFOR) | 0 | <p>RCVR FIFO Reset. This resets the control portion of the receive FIFO and treats the FIFO as empty.</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |
| 0 | W | FIFO Enable (or FIFOE) | 0 | <p>FIFO Enable. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs.</p> <p>Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.</p> |

1026 . FIFO Control Register Description

UART LINE CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------|-----------|---|
| 7 | R/W | DLAB | 0 | <p>Divisor Latch Access Bit. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH/LPDLL and LPDLH) to set the baud rate of the UART.</p> <p>This bit must be cleared after initial baud rate setup in order to access other registers</p> |
| 6 | R/W | Break (or BC) | 0 | <p>Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to 1, the serial output is forced to the spacing (logic 0) state.</p> <p>When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared.</p> <p>If active (MCR[6] set to 1) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.</p> |
| 5 | R/W | Stick Parity | 0 | <p>Stick Parity. This bit is used to force parity value. When PEN, EPS and Stick Parity are set to 1, the parity bit is transmitted and checked as logic 0.</p> <p>If PEN and Stick Parity are set to 1 and EPS is a logic 0, then parity bit is transmitted and checked as a logic 1.</p> <p>If this bit is set to 0, Stick Parity is disabled.</p> |
| 4 | R/W | EPS | 0 | <p>Even Parity Select. This is used to select between even and odd parity, when parity is enabled (PEN set to 1).</p> <p>If set to 1, an even number of logic 1s is transmitted or checked. If set to 0, an odd number of logic 1s is transmitted or checked.</p> |
| 3 | R/W | PEN | 0 | <p>Parity Enable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively.</p> <ul style="list-style-type: none"> • 0 – parity disabled • 1 – parity enabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------------------|-----------|--|
| 2 | R/W | STOP | 0 | <p>Number of stop bits. This is used to select the number of stop bits per character that the peripheral transmits and receives.</p> <p>If set to 0, one stop bit is transmitted in the serial data.</p> <p>If set to 1 and the data bits are set to 5 (LCR[1:0] set to 0) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted.</p> <p>Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit.</p> <ul style="list-style-type: none"> • 0 – 1 stop bit • 1 – 1.5 stop bits when DLS (LCR[1:0]) is 0, else 2 stop bit <p>NOTE: The STOP bit duration implemented by uart may appear longer due to idle time inserted between characters for some configurations and baud clock divisor values in the transmit direction.</p> |
| 1:0 | R/W | DLS (or CLS, as used in legacy) | 0 | <p>Data Length Select.</p> <p>When DLS_E in LCR_EXT is set to 0, this register is used to select the number of data bits per character that the peripheral transmits and receives.</p> <p>The number of bits that may be selected are as follows:</p> <ul style="list-style-type: none"> • 00 – 5 bits • 01 – 6 bits • 10 – 7 bits • 11 – 8 bits |

1027 . Line Control Register Description

UART MODEM CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:7 | R | Reserved | 0 | Reserved |
| 6 | R/W | SIRE | 0 | <p>SIR Mode Enable. This is used to enable/disable the IrDA SIR Mode features</p> <ul style="list-style-type: none"> • 0 – IrDA SIR Mode disabled • 1 – IrDA SIR Mode enabled <p>Note: To enable SIR mode, write the appropriate value to the MCR register before writing to the LCR register</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 5 | R/W | AFCE | 0 | <p>Auto Flow Control Enable. When FIFOs are enabled and the Auto Flow Control Enable (AFCE) bit is set, Auto Flow Control features are enabled</p> <ul style="list-style-type: none"> • 0 – Auto Flow Control Mode disabled • 1 – Auto Flow Control Mode enabled |
| 4 | R/W | LoopBack (or LB) | 0 | <p>LoopBack Bit. This is used to put the UART into a diagnostic mode for test purposes.</p> <p>If operating in UART mode, data on the sout line is held high, while serial data output is looped back to the sin line, internally.</p> <p>In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected</p> <p>and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally.</p> <p>If operating in infrared mode (MCR[6] set to 1), data on the sir_out_n line is held low, while serial data output is inverted and looped back to the sir_in line.</p> |
| 3 | R/W | OUT2 | 0 | <p>OUT2. This is used to directly control the user-designated Output2 (out2_n) output.</p> <p>The value written to this location is inverted and driven out on out2_n, that is:</p> <ul style="list-style-type: none"> • 0 – out2_n de-asserted (logic 1) • 1 – out2_n asserted (logic 0) <p>Note that in Loopback mode (MCR[4] set to 1), the out2_n output is held inactive high while the value of this location is internally looped back to an input.</p> |
| 2 | R/W | OUT1 | 0 | <p>OUT1. This is used to directly control the user-designated Output1 (out1_n) output.</p> <p>The value written to this location is inverted and driven out on out1_n, that is:</p> <ul style="list-style-type: none"> • 0 – out1_n de-asserted (logic 1) • 1 – out1_n asserted (logic 0) <p>Note that in Loopback mode (MCR[4] set to 1), the out1_n output is held inactive high while the value of this location is internally looped back to an input.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 1 | R/W | RTS | 0 | <p>Request to Send. This is used to directly control the Request to Send (rts_n) output.</p> <p>The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data.</p> <p>When Auto RTS Flow Control is not enabled (MCR[5] set to 0), the rts_n signal is set low by programming MCR[1] (RTS) to a high.</p> <p>In Auto Flow Control, (MCR[5] set to 1) and FIFOs enable (FCR[0] set to 1), the rts_n output is controlled in the same way but is also gated with the receiver FIFO threshold trigger</p> <p>(rts_n is inactive high when above the threshold) only when the RTC Flow Trigger is disabled; otherwise it is gated by the receiver FIFO almost-full trigger, where “almost full” refers</p> <p>to two available slots in the FIFO (rts_n is inactive high when above the threshold). The rts_n signal is de-asserted when MCR[1] is set low.</p> <p>Note that in Loopback mode (MCR[4] set to 1), the rts_n output is held inactive high while the value of this location is internally looped back to an input.</p> |
| 0 | R/W | DTR | 0 | <p>Data Terminal Ready. This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is:</p> <ul style="list-style-type: none"> • 0 – dtr_n de-asserted (logic 1) • 1 – dtr_n asserted (logic 0) <p>The Data Terminal Ready output is used to inform the modem or data set that the UART is ready to establish communications. Note that in Loopback mode (MCR[4] set to 1),</p> <p>the dtr_n output is held inactive high while the value of this location is internally looped back to an input.</p> |

1028 . ModemControl Register Description

UART LINE STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:9 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|---|
| 8 | R/W | ADDR_RCVD | 0 | <p>Address Received bit</p> <p>If 9-bit data mode (LCR_EXT[0]=1) is enabled, this bit is used to indicate that the 9th bit of the receive data is set to 1.</p> <p>This bit can also be used to indicate whether the incoming character is an address or data.</p> <ul style="list-style-type: none"> • 1 - Indicates that the character is an address. • 0 - Indicates that the character is data. <p>In the FIFO mode, since the 9th bit is associated with the received character, it is revealed when the character with the 9th bit set to 1 is at the top of the FIFO list.</p> <p>Reading the LSR clears the 9th bit.</p> <p>NOTE: You must ensure that an interrupt gets cleared (reading LSR register) before the next address byte arrives.</p> <p>If there is a delay in clearing the interrupt, then software will not be able to distinguish between multiple address related interrupt.</p> |
| 7 | R | RFE | 0 | <p>Receiver FIFO Error bit. This bit is only relevant when FIFO_MODE != NONE AND FIFOs are enabled (FCR[0] set to 1).</p> <p>This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO.</p> <ul style="list-style-type: none"> • 0 – no error in RX FIFO • 1 – error in RX FIFO <p>This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO.</p> |
| 6 | R | TEMT | 1 | <p>Transmitter Empty bit. If in FIFO mode (FIFO_MODE != NONE) and FIFOs enabled (FCR[0] set to 1), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty.</p> <p>If in non-FIFO mode or FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 5 | R | THRE | 1 | <p>Transmit Holding Register Empty bit. If THRE mode is disabled (IER[7] set to 0) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty.</p> <p>This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO.</p> <p>This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If both modes are active (IER[7] set to 1 and FCR[0] set to 1 respectively), the functionality is switched to indicate</p> <p>the transmitter FIFO is full and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.</p> |
| 4 | R | BI | 0 | <p>Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data.</p> <p>If in UART mode, it is set whenever the serial input, sin, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop bits.</p> <p>If in infrared mode, it is set whenever the serial input, sir_in, is continuously pulsed to logic '0' for longer than the sum of start time + data bits + parity + stop bits.</p> <p>A break condition on serial input causes one and only one character, consisting of all 0s, to be received by the UART.</p> <p>In FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO.</p> <p>Reading the LSR clears the BI bit. In non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read.</p> <p>NOTE: If a FIFO is full when a break condition is received, a FIFO overrun occurs.</p> <p>The break condition and all the information associated with it—parity and framing errors—is discarded; any information that a break character was received is lost.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 3 | R | FE | 0 | <p>A framing error occurs when the receiver does not detect a valid STOP bit in the received data.</p> <p>In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. When a framing error occurs, the uart tries to re-synchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit; that is, data, and/or parity and stop. It should be noted that the Framing Error (FE) bit (LSR[3]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). This happens because the break character implicitly generates a framing error by holding the sin input to logic 0 for longer than the duration of a character.</p> <ul style="list-style-type: none"> • 0 – no framing error • 1 – framing error <p>Reading the LSR clears the FE bit.</p> |
| 2 | R | PE | 0 | <p>Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set.</p> <p>In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO.</p> <p>It should be noted that the Parity Error (PE) bit (LSR[2]) can be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). In this situation, the Parity Error bit is set if parity generation and detection is enabled (LCR[3]=1) and the parity is set to odd (LCR[4]=0).</p> <ul style="list-style-type: none"> • 0 – no parity error • 1 – parity error <p>Reading the LSR clears the PE bit.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 1 | R | OE | 0 | <p>Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read.</p> <p>In the non-FIFO mode, the OE bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver.</p> <p>The data in the FIFO is retained and the data in the receive shift register is lost.</p> <ul style="list-style-type: none"> • 0 – no overrun error • 1 – overrun error <p>Reading the LSR clears the OE bit.</p> |
| 0 | R | DR | 0 | <p>Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO.</p> <ul style="list-style-type: none"> • 0 – no data ready • 1 – data ready <p>This bit is cleared when the RBR is read in non-FIFO mode, or when the receiver FIFO is empty, in FIFO mode.</p> |

1029 .Line Status Register Description

UART MODEM STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7 | R | DCD | 0 | <p>Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n. This bit is the complement of dcd_n.</p> <p>When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set.</p> <ul style="list-style-type: none"> • 0 – dcd_n input is de-asserted (logic 1) • 1 – dcd_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] set to 1), DCD is the same as MCR[3] (Out2).</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 6 | R | RI | 0 | <p>Ring Indicator. This is used to indicate the current state of the modem control line ri_n. This bit is the complement of ri_n.</p> <p>When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set.</p> <ul style="list-style-type: none"> • 0 – ri_n input is de-asserted (logic 1) • 1 – ri_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] set to 1), RI is the same as MCR[2] (Out1)</p> |
| 5 | R | DSR | 0 | <p>Data Set Ready. This is used to indicate the current state of the modem control line dsr_n. This bit is the complement of dsr_n.</p> <p>When the Data Set Ready input (dsr_n) is asserted it is an indication that the modem or data set is ready to establish communications with the uart.</p> <ul style="list-style-type: none"> • 0 – dsr_n input is de-asserted (logic 1) • 1 – dsr_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] set to 1), DSR is the same as MCR[0] (DTR).</p> |
| 4 | R | CTS | 0 | <p>This bit is the complement of cts_n. When the Clear to Send input (cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the uart.</p> <ul style="list-style-type: none"> • 0 – cts_n input is de-asserted (logic 1) • 1 – cts_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] = 1), CTS is the same as MCR[1] (RTS).</p> |
| 3 | R | DDCD | 0 | <p>Delta Data Carrier Detect. This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on dcd_n since last read of MSR • 1 – change on dcd_n since last read of MSR <p>Reading the MSR clears the DDCD bit. In Loopback Mode (MCR[4] = 1), DDCD reflects changes on MCR[3] (Out2).</p> <p>Note, if the DDCD bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit is set when the reset is removed if the dcd_n signal remains asserted.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 2 | R | TERI | 0 | <p>Trailing Edge of Ring Indicator. This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on ri_n since last read of MSR • 1 – change on ri_n since last read of MSR <p>Reading the MSR clears the TERI bit. In Loopback Mode (MCR[4] = 1), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.</p> |
| 1 | R | DDSR | 0 | <p>Delta Data Set Ready. This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on dsr_n since last read of MSR • 1 – change on dsr_n since last read of MSR <p>Reading the MSR clears the DDSR bit. In Loopback Mode (MCR[4] = 1), DDSR reflects changes on MCR[0] (DTR).</p> <p>Note, if the DDSR bit is not set and the dsr_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDSR bit is set when the reset is removed if the dsr_n signal remains asserted.</p> |
| 0 | R | DCTS | 0 | <p>Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on cts_n since last read of MSR • 1 – change on cts_n since last read of MSR <p>Reading the MSR clears the DCTS bit. In Loopback Mode (MCR[4] = 1), DCTS reflects changes on MCR[1] (RTS).</p> <p>Note, if the DCTS bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit is set when the reset is removed if the cts_n signal remains asserted.</p> |

1030 . Modem Status Register Description

UART SCRATCHPAD REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | Scratchpad Register | 0 | This register is for programmers to use as a temporary storage space. It has no defined purpose in the uart. |

1031 . Scratchpad Register Description

UART LOW POWER DIVISOR LATCH(LOW) REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | LPDLL | 0 | <p>This register makes up the lower 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART,</p> <p>which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver.</p> <p>The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> $\text{Low power baud rate} = (\text{serial clock frequency}) / (16 * \text{divisor})$ <p>Therefore, a divisor must be selected to give a baud rate of 115.2K.</p> <p>NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver.</p> <p>Also, once the LPDLL is set, at least eight clock cycles of the slowest uart clock should be allowed to pass before transmitting or receiving data.</p> |

1032 . Low Power Divisor Latch Low Register Description

UART LOW POWER DIVISOR LATCH(HIGH) REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 7:0 | R/W | LPDLH | 0 | <p>This register makes up the upper 8-bits of a 16-bit, read/write, Low Power Divisor Latch register that contains the baud rate divisor for the UART,</p> <p>which must give a baud rate of 115.2K. This is required for SIR Low Power (minimum pulse width) detection at the receiver. The output low-power baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows:</p> $\text{Low power baud rate} = (\text{serial clock frequency}) / (16 * \text{divisor})$ <p>Therefore, a divisor must be selected to give a baud rate of 115.2K.</p> <p>NOTE: When the Low Power Divisor Latch registers (LPDLL and LPDLH) are set to 0, the low-power baud clock is disabled and no low-power pulse detection (or any pulse detection) occurs at the receiver.</p> <p>Also, once the LPDLH is set, at least eight clock cycles of the slowest uart clock should be allowed to pass before transmitting or receiving data.</p> |

1033 . Low Power Divisor Latch High Register Description

UART SHADOW RECEIVE BUFFER REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--|-----------|---|
| 31:9 | R | Reserved | 0 | Reserved |
| 8 | R | Shadow Receive Buffer Register (MSB 9th bit) | 0 | <p>This is a shadow register for the RBR[8] bit.</p> <p>It is applicable only when UART_9BIT_DATA_EN=1.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---|-----------|--|
| 7:0 | R | Shadow Receive Buffer Register (LSB 8 bits) | 0 | <p>This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master.</p> <p>This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode.</p> <p>The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set.</p> <p>If in non-FIFO mode (FIFO_MODE = NONE) or FIFOs are disabled (FCR[0] set to 0), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an overrun error.</p> <p>If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to 1), this register accesses the head of the receive FIFO.</p> <p>If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO are preserved, but any incoming data is lost. An overrun error also occurs.</p> |

1034 . Shadow Receive Buffer Register Description

UART SHADOW TRANSMIT HOLDING REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--|-----------|--|
| 31:9 | R | Reserved | 0 | Reserved |
| 8 | W | Shadow Transmit Holding Register (MSB 9th bit) | 0 | <p>This is a shadow register for the THR[8] bit.</p> <p>It is applicable only when UART_9BIT_DATA_EN=1</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------------------|-----------|---|
| 7:0 | W | Shadow Transmit Holding Register | 0 | <p>This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master.</p> <p>This register contains data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode.</p> <p>Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set.</p> <p>If in non-FIFO mode or FIFOs are disabled (FCR[0] set to 0) and THRE is set, writing a single character to the THR clears the THRE.</p> <p>Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten.</p> <p>If in FIFO mode and FIFOs are enabled (FCR[0] set to 1) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full.</p> <p>The number x (default=16) is determined by the value of FIFO Depth that is set during configuration.</p> <p>Any attempt to write data when the FIFO is full results in the write data being lost.</p> |

1035 . Shadow Transmit Holding Register Description

UART FIFO ACCESS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | FIFO Access | 0 | <p>Writes have no effect when FIFO_ACCESS = No, always readable. This register is use to enable a FIFO access mode for testing,</p> <p>so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are implemented and enabled.</p> <p>When FIFOs are not implemented or not enabled it allows the RBR to be written by the master and the THR to be read by the master.</p> <ul style="list-style-type: none"> • 0 – FIFO access mode disabled • 1 – FIFO access mode enabled <p>Note, that when the FIFO access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFOs are treated as empty.</p> |

1036 . FIFO Access Register Description

UART TRANSMIT FIFO READ REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R | Transmit FIFO Read | 0 | <p>These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, reading this register gives the data at the top of the transmit FIFO.</p> <p>Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO.</p> <p>When FIFOs are not implemented or not enabled, reading this register gives the data in the THR.</p> |

1037 . Transmit FIFO Read Register Description

UART RECEIVE FIFO WRITE REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:10 | R | Reserved | 0 | Reserved |
| 9 | W | RFFE | 0 | <p>Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, this bit is used to write framing error detection information to the receive FIFO.</p> <p>When FIFOs are not implemented or not enabled, this bit is used to write framing error detection information to the RBR.</p> |
| 8 | W | RFPE | 0 | <p>Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, this bit is used to write parity error detection information to the receive FIFO.</p> <p>When FIFOs are not implemented or not enabled, this bit is used to write parity error detection information to the RBR.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 7:0 | W | RFWD | 0 | <p>Receive FIFO Write Data. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, the data that is written to the RFWD is pushed into the receive FIFO.</p> <p>Each consecutive write pushes the new data to the next write location in the receive FIFO.</p> <p>When FIFOs are not implemented or not enabled, the data that is written to the RFWD is pushed into the RBR.</p> |

1038 . Receive FIFO Write Register Description

UART STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:5 | R | Reserved | 0 | Reserved |
| 4 | R | RFF | 0 | <p>Receive FIFO Full. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the receive FIFO is completely full.</p> <ul style="list-style-type: none"> • 0 – Receive FIFO not full • 1 – Receive FIFO Full <p>This bit is cleared when the RX FIFO is no longer full.</p> |
| 3 | R | RFNE | 0 | <p>Receive FIFO Not Empty. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the receive FIFO contains one or more entries.</p> <ul style="list-style-type: none"> • 0 – Receive FIFO is empty • 1 – Receive FIFO is not empty <p>This bit is cleared when the RX FIFO is empty.</p> |
| 2 | R | TFE | 0 | <p>Transmit FIFO Empty. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the transmit FIFO is completely empty.</p> <ul style="list-style-type: none"> • 0 – Transmit FIFO is not empty • 1 – Transmit FIFO is empty <p>This bit is cleared when the TX FIFO is no longer empty.</p> |
| 1 | R | TFNF | 0 | <p>Transmit FIFO Not Full. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the transmit FIFO is not full.</p> <ul style="list-style-type: none"> • 0 – Transmit FIFO is full • 1 – Transmit FIFO is not full <p>This bit is cleared when the TX FIFO is full.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 0 | R | BUSY | 0 | <p>UART Busy. This bit indicates that a serial transfer is in progress; when cleared, indicates that the uart is idle or inactive.</p> <ul style="list-style-type: none"> • 0 – uart is idle or inactive • 1 – uart is busy (actively transferring data) <p>This bit will be set to 1 (busy) under any of the following conditions:</p> <ol style="list-style-type: none"> 1. Transmission in progress on serial interface 2. Transmit data present in THR, when FIFO access mode is not being used (FAR = 0) and the baud divisor is non-zero ($\{DLH,DLL\}$ does not equal 0) <ul style="list-style-type: none"> when the divisor latch access bit is 0 (LCR.DLAB = 0) 3. Reception in progress on the interface 4. Receive data present in RBR, when FIFO access mode is not being used (FAR = 0) <p>NOTE: It is possible for the UART Busy bit to be cleared even though a new character may have been sent from another device.</p> <p>That is, if the uart has no data in THR and RBR and there is no transmission in progress and a start bit of a new character has just reached the uart.</p> <p>This is due to the fact that a valid start is not seen until the middle of the bit period and this duration is dependent on the baud divisor that has been programmed.</p> <p>If a second system clock has been implemented, the assertion of this bit is also delayed by several cycles of the slower clock.</p> |

1039 . UART Status Register Description

UART TRANSMIT FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|--------------------|--------|---------------------|-----------|--|
| 31:FIFO_ADDR_WIDTH | R | Reserved | 0 | Reserved |
| FIFO_ADDR_WIDTH:0 | R | Transmit FIFO Level | 0 | This indicates the number of data entries in the transmit FIFO |

1040 . Transmit FIFO Level Register Description

UART RECEIVE FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|--------------------|--------|--------------------|-----------|--|
| 31:FIFO_ADDR_WIDTH | R | Reserved | 0 | Reserved |
| FIFO_ADDR_WIDTH:0 | R | Receive FIFO Level | 0 | This indicates the number of data entries in the receive FIFO. |

1041 . Receive FIFO Level Register Description

UART SOFTWARE RESET REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:3 | R | Reserved | 0 | Reserved |
| 2 | W | XFR | 0 | XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO. This resets the control portion of the transmit FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing'. It is not necessary to clear this bit. |
| 1 | W | RFR | 0 | RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO. This resets the control portion of the receive FIFO and treats the FIFO as empty. Note that this bit is 'self-clearing'. It is not necessary to clear this bit. |
| 0 | W | UR | 0 | UART Reset. This asynchronously resets the uart and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains are reset. |

1042 . Software Reset Register Description

UART SHADOW REQUEST TO SEND REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------------------|-----------|---|
| 0 | R/W | Shadow Request to Send | 0 | <p>This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to performing a read-modify-write on the MCR.</p> <p>This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the uart is ready to exchange data. When Auto RTS Flow Control is not enabled (MCR[5] = 0), the rts_n signal is set low by programming MCR[1] (RTS) to a high.</p> <p>In Auto Flow Control, (MCR[5] = 1) and FIFOs enable (FCR[0] = 1), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold) only when RTC Flow Trigger is disabled; otherwise it is gated by the receiver FIFO almost-full trigger, where “almost full” refers to two available slots in the FIFO (rts_n is inactive high when above the threshold).</p> <p>Note that in Loopback mode (MCR[4] = 1), the rts_n output is held inactive-high while the value of this location is internally looped back to an input.</p> |

1043 . Shadow Request to Send Register Description

UART SHADOW BREAK CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------------|-----------|--|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | Shadow Break Control Register | 0 | <p>This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to performing a read modify write on the LCR.</p> <p>This is used to cause a break condition to be transmitted to the receiving device.</p> <p>If set to 1, the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared.</p> <p>If (MCR[6] = 1), then sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver.</p> |

1044 . Shadow Break Control Register Description

UART SHADOW DMA MODE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | Shadow DMA Mode | 0 | <p>This is a shadow register for the DMA mode bit (FCR[3]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the DMA Mode bit gets updated.</p> <p>This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals.</p> <ul style="list-style-type: none"> • 0 – mode 0 • 1 – mode 1 |

1045 . Shadow DMA Mode Register Description

UART SHADOW FIFO ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | Shadow FIFO Enable | 0 | <p>Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the FIFO enable bit gets updated.</p> <p>This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. If this bit is set to 0 (disabled) after being enabled then both the XMIT and RCVR controller portion of FIFOs are reset.</p> |

1046 . Shadow FIFO Enable Register Description

UART SHADOW RCVR TRIGGER REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|--|
| 0 | R/W | Shadow RCVR Trigger0 | | <p>Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the RCVR trigger bit gets updated.</p> <p>This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. It also determines when the dma_rx_req_n signal is asserted when DMA Mode (FCR[3]) = 1.</p> <p>The following trigger levels are supported:</p> <ul style="list-style-type: none"> • 00 – 1 character in the FIFO • 01 – FIFO ¼ full • 10 – FIFO ½ full • 11 – FIFO 2 less than full |

1047 . Shadow RCVR Trigger Register Description

UART SHADOW TX EMPTY TRIGGER REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------|-----------|--|
| 31:2 | R | Reserved | 0 | Reserved |
| 1:0 | R/W | Shadow TX Empty Trigger | 0 | <p>Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the TX empty trigger bit gets updated.</p> <p>This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active.</p> <p>The following trigger levels are supported:</p> <ul style="list-style-type: none"> • 00 – FIFO empty • 01 – 2 characters in the FIFO • 10 – FIFO ¼ full • 11 – FIFO ½ full |

1048 . Shadow TX Empty Trigger Register Description

UART HALT TX REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 0 | R/W | HALT TX | 0 | <p>This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled.</p> <ul style="list-style-type: none"> • 0 – Halt TX disabled • 1 – Halt TX enabled <p>Note, if FIFOs are implemented and not enabled, the setting of the halt TX register has no effect on operation.</p> |

1049 . HALT TX Register Description

UART DMA SOFTWARE ACKNOWLEDGE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------------|-----------|--|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | W | DMA Software Acknowledge | 0 | <p>This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition.</p> <p>For example, if the DMA disables the channel, then the uart should clear its request.</p> <p>This causes the TX request, TX single, RX request and RX single signals to de-assert.</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |

1050 . DMA Software Acknowledgment Register Description

UART TRANSCEIVER CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:5 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------|-----------|--|
| 4:3 | R/W | XFER_MODE | 0 | <p>Transfer Mode</p> <ul style="list-style-type: none"> 0 : In this mode, transmit and receive can happen simultaneously. You can enable DE_EN and RE_EN at any point of time. Turn around timing as programmed in the TAT register is not applicable in this mode. 1 : In this mode, DE and RE are mutually exclusive. The hardware considers the turnaround timings that are programmed in the TAT register while switching from RE to DE or from DE to RE. Ensure that either DE or RE is expected to be enabled while programming. <p>For transmission, hardware waits if it is in the midst of receiving any transfer, before it starts transmitting.</p> <ul style="list-style-type: none"> 2 : In this mode, DE and RE are mutually exclusive. Once DE_EN or RE_EN is programmed, 're' is enabled by default and uart controller will be ready to receive. <p>If the user programs the TX FIFO with data, then uart, after ensuring no receive is in progress, disables the 're' and enables the 'de' signal.</p> <p>Once the TX FIFO becomes empty, the 're' signal gets enabled and the 'de' signal will be disabled. In this mode of operation, the hardware considers the turnaround timings that are programmed in the TAT register while switching from RE to DE or from DE to RE. In this mode, 'de' and 're' signals are strictly complementary to each other.</p> |
| 2 | R/W | DE_POL | 1 | <p>Driver Enable Polarity</p> <ul style="list-style-type: none"> 1: DE signal is active high 0: DE signal is active low <p>Reset Value: UART_DE_POL</p> |
| 1 | R/W | RE_POL | 1 | <p>Receiver Enable Polarity</p> <ul style="list-style-type: none"> 1: RE signal is active high 0: RE signal is active low <p>Reset Value: UART_RE_POL</p> |
| 0 | R/W | RS485_EN | 0 | <p>RS485 Transfer Enable</p> <ul style="list-style-type: none"> 0 : In this mode, the transfers are still in the RS232 mode. All other fields in this register are reserved and registers DE_EN, RE_EN, DET and TAT are reserved. 1: In this mode, the transfers will happen in RS485 mode. All other fields of this register are applicable. |

1051 . Transceiver Control Register Description

UART DRIVER OUTPUT ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|--|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | DE Enable | 0 | <p>DE Enable control</p> <p>The 'DE Enable' register bit is used to control assertion and de-assertion of 'de' signal.</p> <ul style="list-style-type: none"> • 0: De-assert 'de' signal • 1: Assert 'de' signal |

1052 . Driver Output Enable Register Description

UART RECEIVER OUTPUT ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-----------|--|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | RE Enable | 0 | <p>RE Enable control</p> <p>The 'RE Enable' register bit is used to control assertion and de-assertion of 're' signal.</p> <ul style="list-style-type: none"> • 0: De-assert 're' signal • 1: Assert 're' signal |

1053 . Receiver Output Enable Register Description

UART DRIVER OUTPUT ENABLE TIMING REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------|-----------|--|
| 31:24 | R | Reserved | 0 | Reserved |
| 23:16 | R/W | DE de-assertion time | 0 | <p>Driver enable de-assertion time. This field controls the amount of time (in terms of number of serial clock periods) between the end of stop bit on the serial output (sout) to the falling edge of Driver output enable signal</p> |
| 15:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | DE assertion time | 0 | <p>Driver enable assertion time. This field controls the amount of time (in terms of number of serial clock periods) between the assertion of rising edge of Driver output enable signal to serial transmit enable. Any data in transmit buffer, will start on serial output (sout) after the transmit enable.</p> |

1054 . Driver Output Enable Timing Register Description

UART TURNAROUND TIMING REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:16 | R/W | RE to DE | 0 | <p>Receiver Enable to Driver Enable Turn Around time. Turnaround time (in terms of serial clock) for RE de-assertion to DE assertion.</p> <p>Note:</p> <ul style="list-style-type: none"> • If the DE assertion time in the DET register is 0, then the actual value is the programmed value + 3. • If the DE assertion time in the DET register is 1, then the actual value is the programmed value + 2. • If the DE assertion time in the DET register is greater than 1, then the actual value is the programmed value + 1. |
| 15:0 | R/W | DE to RE | 0 | <p>Driver Enable to Receiver Enable Turn Around time. Turnaround time (in terms of serial clock) for DE de-assertion to RE assertion.</p> <p>Note: The actual time is the programmed value + 1.</p> |

1055 . TurnAround Timing Register Description

UART DIVISOR LATCH FRACTIONAL VALUE REGISTER

| Bit | Access | Function | POR Value | Description |
|-------------|--------|----------|-----------|-------------|
| 31:DLF_SIZE | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------|----------|------------------|-----------|---|--|--|-----------|----------|------------------|------|------|--------|------|------|--------|------|------|-------|------|------|--------|------|------|------|------|------|--------|------|------|-------|------|------|--------|------|------|-----|------|------|--------|------|-------|-------|------|-------|--------|------|-------|------|------|-------|--------|------|-------|-------|------|-------|--------|
| DLF_SIZE-1:0 | R/W | DLF | 0 | Fractional part of divisor. The fractional value is added to integer value set by DLH, DLL. Fractional value is determined by $(\text{Divisor Fraction value})/(2^{\text{DLF_SIZE}})$ The DLF Values to be programmed for DLF_SIZE=4 are <table> <thead> <tr> <th>DLF_Value</th> <th>Fraction</th> <th>Fractional Value</th> </tr> </thead> <tbody> <tr><td>0000</td><td>0/16</td><td>0.0000</td></tr> <tr><td>0001</td><td>1/16</td><td>0.0625</td></tr> <tr><td>0010</td><td>2/16</td><td>0.125</td></tr> <tr><td>0011</td><td>3/16</td><td>0.1875</td></tr> <tr><td>0100</td><td>4/16</td><td>0.25</td></tr> <tr><td>0101</td><td>5/16</td><td>0.3125</td></tr> <tr><td>0110</td><td>6/16</td><td>0.375</td></tr> <tr><td>0111</td><td>7/16</td><td>0.4375</td></tr> <tr><td>1000</td><td>8/16</td><td>0.5</td></tr> <tr><td>1001</td><td>9/16</td><td>0.5625</td></tr> <tr><td>1010</td><td>10/16</td><td>0.625</td></tr> <tr><td>1011</td><td>11/16</td><td>0.6875</td></tr> <tr><td>1100</td><td>12/16</td><td>0.75</td></tr> <tr><td>1101</td><td>13/16</td><td>0.8125</td></tr> <tr><td>1110</td><td>14/16</td><td>0.875</td></tr> <tr><td>1111</td><td>15/16</td><td>0.9375</td></tr> </tbody> </table> | | | DLF_Value | Fraction | Fractional Value | 0000 | 0/16 | 0.0000 | 0001 | 1/16 | 0.0625 | 0010 | 2/16 | 0.125 | 0011 | 3/16 | 0.1875 | 0100 | 4/16 | 0.25 | 0101 | 5/16 | 0.3125 | 0110 | 6/16 | 0.375 | 0111 | 7/16 | 0.4375 | 1000 | 8/16 | 0.5 | 1001 | 9/16 | 0.5625 | 1010 | 10/16 | 0.625 | 1011 | 11/16 | 0.6875 | 1100 | 12/16 | 0.75 | 1101 | 13/16 | 0.8125 | 1110 | 14/16 | 0.875 | 1111 | 15/16 | 0.9375 |
| DLF_Value | Fraction | Fractional Value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0000 | 0/16 | 0.0000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0001 | 1/16 | 0.0625 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0010 | 2/16 | 0.125 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0011 | 3/16 | 0.1875 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0100 | 4/16 | 0.25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0101 | 5/16 | 0.3125 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0110 | 6/16 | 0.375 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0111 | 7/16 | 0.4375 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000 | 8/16 | 0.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1001 | 9/16 | 0.5625 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1010 | 10/16 | 0.625 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1011 | 11/16 | 0.6875 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1100 | 12/16 | 0.75 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1101 | 13/16 | 0.8125 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1110 | 14/16 | 0.875 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1111 | 15/16 | 0.9375 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

1056 . Divisor Latch Fraction Register Description

UART RECEIVE ADDRESS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 7:0 | R/W | RAR | 0 | <p>This is an address matching register during receive mode. If the 9-th bit is set in the incoming character then the remaining 8-bits will be checked against this register value. If the match happens then subsequent characters with 9-th bit set to 0 will be treated as data byte until the next address byte is received.</p> <p>Note:</p> <ul style="list-style-type: none"> • This register is applicable only when 'ADDR_MATCH' (LCR_EXT[1]) and 'DLS_E' (LCR_EXT[0]) bits are set to 1. • RAR should be programmed only when UART is not busy. • RAR can be programmed at any point of the time. However, user must not change this register value when any receive is in progress. |

1057 . Receive Address Register Description

UART TRANSMIT ADDRESS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | TAR | 0 | <p>This is an address matching register during transmit mode. If DLS_E (LCR_EXT[0]) bit is enabled, then uart sends the 9-bit character with 9-th bit set to 1 and remaining 8-bit address will be sent from this register provided 'SEND_ADDR' (LCR_EXT[2]) bit is set to 1.</p> <p>NOTE:</p> <ul style="list-style-type: none"> • This register is used only to send the address. The normal data should be sent by programming THR register. • Once the address is started to send on the uart serial lane, then 'SEND_ADDR' bit will be auto-cleared by the hardware. |

1058 . Transmit Address Register Description

UART LINE EXTENDED CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:4 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------|-----------|--|
| 3 | R/W | TRANSMIT_MODE | 0 | <p>Transmit mode control bit. This bit is used to control the type of transmit mode during 9-bit data transfers.</p> <ul style="list-style-type: none"> • 1 : In this mode of operation, Transmit Holding Register (THR) and Shadow Transmit Holding Register (STHR) are 9-bit wide. The user must ensure that the THR/STHR register is written correctly for address/data. <p>Address: 9th bit is set to 1, Data: 9th bit is set to 0.</p> <p>NOTE: Transmit address register (TAR) is not applicable in this mode of operation.</p> <ul style="list-style-type: none"> • 0 : In this mode of operation, Transmit Holding Register (THR) and Shadow Transmit Holding register (STHR) are 8-bit wide. The user needs to program the address into Transmit Address Register (TAR) and data into the THR/STHR register. <p>SEND_ADDR bit is used as a control knob to indicate the uart on when to send the address.</p> |
| 2 | R/W | SEND_ADDR | 0 | <p>Send address control bit. This bit is used as a control knob for the user to determine when to send the address during transmit mode.</p> <ul style="list-style-type: none"> • 1 - 9-bit character will be transmitted with 9-th bit set to 1 and the remaining 8-bits will match to what is being programmed in "Transmit Address Register". • 0 - 9-bit character will be transmitted with 9-th bit set to 0 and the remaining 8-bits will be taken from the TxFIFO which is programmed through 8-bit wide THR/STHR register. <p>NOTE:</p> <ol style="list-style-type: none"> 1. This bit is auto-cleared by the hardware, after sending out the address character. User is not expected to program this bit to 0. 2. This field is applicable only when DLS_E bit is set to 1 and TRANSMIT_MODE is set to 0. |

| Bit | Access | Function | POR Value | Description |
|-----|--------|------------|-----------|---|
| 1 | R/W | ADDR_MATCH | 0 | <p>Address Match Mode. This bit is used to enable the address match feature during receive.</p> <ul style="list-style-type: none"> • 1 - Address match mode; uart will wait until the incoming character with 9-th bit set to 1. And, further checks to see if the address matches with what is programmed in "Receive Address Match Register". If match is found, then subsequent characters will be treated as valid data and uart starts receiving data. • 0 - Normal mode; uart will start to receive the data and 9-bit character will be formed and written into the receive RxFIFO. <p>User is responsible to read the data and differentiate b/n address and data.</p> <p>NOTE: This field is applicable only when DLS_E is set to 1.</p> |
| 0 | R/W | DLS_E | 0 | <p>Extension for DLS. This bit is used to enable 9-bit data for transmit and receive transfers.</p> <ul style="list-style-type: none"> • 1 = 9 bits per character • 0 = Number of data bits selected by DLS |

1059 . Line Extended Control Register Description

UART COMPONENT PARAMETER REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------------|-----------|---|
| 31:24 | R | Reserved | 0 | Reserved |
| 23:16 | R | FIFO_MODE | 0x01 | <p>0x00 = 0</p> <p>0x01 = 16</p> <p>0x02 = 32</p> <p>to</p> <p>0x80 = 2048</p> <p>0x81- 0xff = reserved</p> |
| 15:14 | R | Reserved | 0 | Reserved |
| 13 | R | DMA_EXTRA | 0x1 | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 12 | R | UART_ADD_ENCODED_PARAMS | 0x0 | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 11 | R | SHADOW | 0x0 | <p>0 – FALSE</p> <p>1 – TRUE</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------|-----------|--|
| 10 | R | FIFO_STAT | 0x1 | 0 – FALSE 1 – TRUE |
| 9 | R | FIFO_ACCESS | 0x0 | 0 – FALSE 1 – TRUE |
| 8 | R | ADDITIONAL_FEAT | 0x1 | 0 – FALSE 1 – TRUE |
| 7 | R | SIR_LP_MODE | 0x1 | 0 – FALSE 1 – TRUE |
| 6 | R | SIR_MODE | 0x1 | 0 – FALSE 1 – TRUE |
| 5 | R | THRE_MODE | 0x1 | 0 – FALSE 1 – TRUE |
| 4 | R | AFCE_MODE | 0x1 | 0 – FALSE 1 – TRUE |
| 3:2 | R | Reserved | 0 | Reserved |
| 1:0 | R | APB_DATA_WIDTH | 0x2 | 00 – 8 bits 01 – 16 bits 10 – 32 bits 11 – reserved |

1060 . Component Parameter Register Description

UART COMPONENT VERSION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------|--------------|--|
| 31:0 | R | UART Component Version | 32'h3430302a | This register contains UART Component Version. |

1061 . UART Component Version Register Description

UART COMPONENT TYPE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------|------------|--|
| 31:0 | R | Peripheral ID | 0x44570110 | This register contains the peripherals identification code |

1062 . Component Type Register Description

16.18 USART

16.18.1 General Description

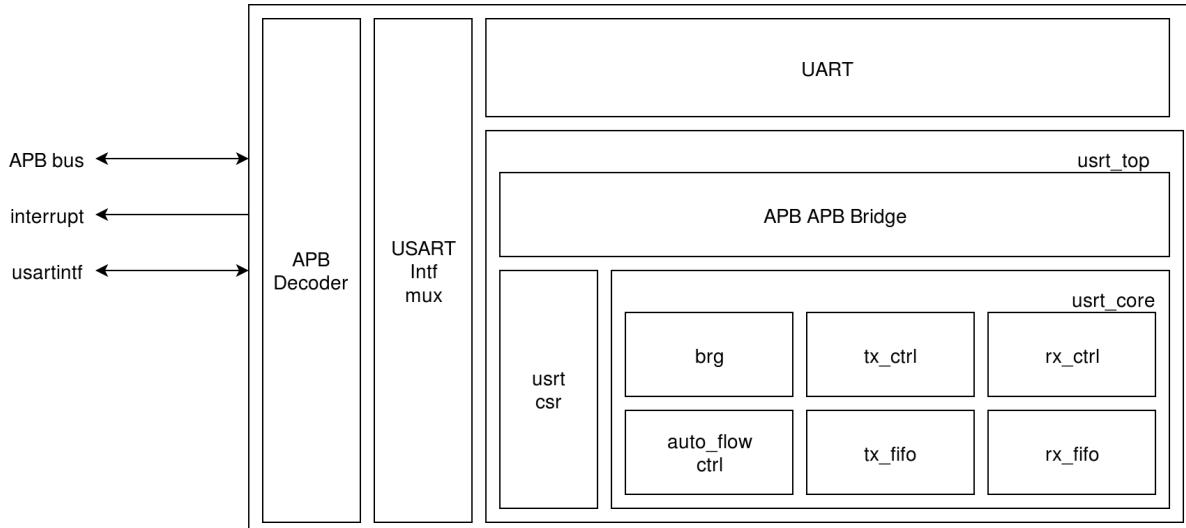
USART is used in communication through wired medium in both Synchronous and Asynchronous fashion. In Synchronous mode both the full duplex and half duplex (single wire) modes are supported

16.18.2 Features

The following features are supported by the USART controller in the MCU HP peripherals (USART1):

- Support for both Synchronous and Asynchronous modes.
- Supports Full duplex and half duplex (single wire) mode of communication.
- 1-9 bit wide character support.
- Support all the baud rates available. Maximum supported baud rate is 7,372,800 (around 7.3Mbps) with 118MHz UART input clock. It supports upto 20Mbps in USART mode.
- Programmable FIFO thresholds with maximum FIFO depth of 16 and support for DMA
- Supports generation of interrupt for different events.

16.18.3 Functional Description



USART block diagram

USART supports both USART and USART functionality. USART is already explained in other sections. USART is going to be explained in this section. USART module contains, Control and Status Registers(CSR), baud rate generator, auto flow control, tx and rx FIFOs, tx controller and rx controller.

This interface MUX muxes the interface signals from USART and USART. In sync_mode USART signal are given out on interface and interface signals are given to USART module and inputs to USART will be blocked. In async_mode USART signal are given out on interface and interface signals are given to USART module and inputs to USART will be blocked.

USART is going to work in two modes. They are continuous clock mode and discontinuous clock mode.

Continuous clock mode:

Data from the external USART comes on *rxd* pin. Clock from the external USART comes on *clk* pin. These two are synchronized on using undivided clock (*usart_clk*). *clks* output of the synchronizer is registered once and passed through a multiplexer. In case of same edge sample and drive mode, this multiplexer is disabled to make sure the synchronized *rxd* (*rxd_s*) holds the valid data. The *clks* is passed through edge det logic where rising and falling edges are generated. These outputs are passed through a multiplexer to select the sampling edge. This edge is given to start/stop det logic where the *rxd_s* is also connected. For every pulse on the sampling signal *rxd_s* is seen

and if it is matched with start_bit (usually 0), data_latch_en is path is activated and the sampling pulses from that point onwards are given to rx_shift_reg and rx_bit_cnt counter where rxd_s is sampled and bit cnt is incremented. When the number of bits is matched to the programmed value a fifo_wr signal is generated to capture the rx_shift_reg value into rx_fifo. Stop bit is detected by the start/stop det logic and given to bit_char_cnt. If stop bit is seen even before all the bits reception a frame error is generated and the rx_shift_reg is flushed and FIFO will not be filled. Second stop bit detection is ignored if two stop bits are adjacent (in case of 1.5/2 stop bits case, 2 stop bits might be seen some times). In master mode case internally divided clock is used as input to edge det logic where as in slave mode clk_s is used. Start/stop bit detection can be bypassed as to receive continuous data characters to increase the throughput.

Discontinuous mode:

In this mode of operation, start/stop bits are not required and every data bit sampled on the sampling edge of clk is a valid data bit. In this mode, start/stop det logic can be fully bypassed. Fully divided clock is the actual baud clock where as fractional divided clock is the clock which is corrected clock to limit the error % with the actual baud rate.

16.18.4 Procedures to use USART

Full duplex mode

- Set sync_mode.
- Write data to TX FIFO.
- Write count to RX_CNT.
- Set auto_flow_ctrl if controlled transfers on interface.

Half duplex mode

RX mode

- Reset full_duplex_mode.
- Set rx_mode.
- Reset tx_mode.
- Write RX_CNT with number of bytes to be received.
- Read data from RX_FIFO by polling rx FIFO empty/aempty status.

TX mode

- Reset full_duplex_mode.
- Set tx_mode.
- Reset rx_mode.
- Write data to TX_FIFO by polling the FIFO full/afull status.

16.18.5 Register Summary

UART Base Address: 0x4400_0000

USART Base Address: 0x4400_0100

| Register Name | Offset | Description |
|---------------|--------|------------------------------|
| USART_RBR | 0x0 | Receive Buffer Register |
| USART_THR | 0x0 | Transmit Holding Register |
| USART_DLL | 0x0 | Divisor Latch (Low) Register |

| Register Name | Offset | Description |
|----------------------|---------------|-----------------------------------|
| USART_DLH | 0x4 | Divisor Latch (High) Register |
| USART_IER | 0x4 | Interrupt Enable Register |
| USART_IIR | 0x8 | Interrupt Identification Register |
| USART_FCR | 0x8 | FIFO Control Register |
| USART_LCR | 0xC | Line Control Register |
| USART_MCR | 0x10 | Modem Control Register |
| USART_LSR | 0x14 | Line Status Register |
| USART_MSR | 0x18 | Modem Status Register |
| USART_SCR | 0x1C | Scratchpad Register |
| USART_FDR | 0x28 | Reserved |
| USART_HDEN | 0x40 | Hardware Enable register |
| USART_SMCR | 0x58 | Control register |
| USART_FAR | 0x70 | FIFO Access Register |
| USART_TFR | 0x74 | Transmit FIFO Read Register |
| USART_RFW | 0x78 | Receive FIFO Write Register |
| USART_USR | 0x7C | UART Status Register |
| USART_TFL | 0x80 | Transmit FIFO Level Register |
| USART_RFL | 0x84 | Receive FIFO Level Register |
| USART_SRR | 0x88 | Software Reset Register |
| USART_SRTS | 0x8C | Shadow Request to Send Register |
| USART_SBCR | 0x90 | Shadow Break Control Register |
| USART_SDMAM | 0x94 | Shadow DMA Mode Register |
| USART_SFE | 0x98 | Shadow FIFO Enable Register |
| USART_SRT | 0x9C | Shadow RCVR Trigger Register |
| USART_STET | 0xA0 | Shadow TX Empty Trigger Register |
| USARTHTX | 0xA4 | Halt TX Register |
| USART_DMASA | 0xA8 | DMA Software Acknowledge Register |
| USART_CPR | 0xF4 | Component Parameter Register |
| UCV | 0xF8 | UART Component Version Register |
| USART_CTR | 0xFC | Component Type Register |

1063 . Register Summary Table

16.18.6 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write

USART RECEIVE BUFFER REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-------------------------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R | Receive Buffer Register | 0x0 | <p>Data byte received on the serial input port (sin) in UART mode, or the serial infrared input (sir_in) in infrared mode.</p> <p>The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LSR) is set.</p> <p>If in non-FIFO mode (FIFO_MODE = NONE) or FIFOs are disabled (FCR[0] set to 0), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error.</p> <p>If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to 1), this register accesses the head of the receive FIFO.</p> <p>If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.</p> |

1064 . Receive Buffer Register Description

USART TRANSMIT HOLDING REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:8 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------------|-----------|--|
| 7:0 | W | Transmit Holding Register | 0x0 | <p>Data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode.</p> <p>Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set.</p> <p>If in non-FIFO mode or FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the THR clears the THRE.</p> <p>Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten.</p> <p>If in FIFO mode and FIFOs are enabled (FCR[0] = 1) and THRE is set, x number of characters of data may be written to the THR before the FIFO is full.</p> <p>The number x (default=16) is determined by the value of FIFO Depth that is set during configuration.</p> <p>Any attempt to write data when the FIFO is full results in the write data being lost.</p> |

1065 . Transmit Holding Register Description

USART DIVISOR LATCH LOW REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | Divisor Latch (Low) | 0 | <p>Lower 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.</p> <p>The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (CLOCK_MODE = Enabled) frequency divided by sixteen times</p> <p>the value of the baud rate divisor, as follows:</p> <p>baud rate = (serial clock freq) / (16 * divisor).</p> <p>Note that with the Divisor Latch Registers (DLL and DLH) set to 0, the baud clock is disabled and no serial communications occur.</p> <p>Also, once the DLL is set, at least 8 clock cycles of the slowest uart clock should be allowed to pass before transmitting or receiving data.</p> |

1066 . Divisor Latch Low Register Description

USART DIVISOR LATCH HIGH REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | Divisor Latch (High) | 0 | <p>Upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.</p> <p>The output baud rate is equal to the serial clock (pclk if one clock design, sclk if two clock design (CLOCK_MODE = Enabled) frequency divided by sixteen times</p> <p>the value of the baud rate divisor, as follows:</p> $\text{baud rate} = (\text{serial clock freq}) / (16 * \text{divisor}).$ <p>Note that with the Divisor Latch Registers (DLL and DLH) set to 0, the baud clock is disabled and no serial communications occur.</p> <p>Also, once the DLH is set, at least 8 clock cycles of the slowest uart clock should be allowed to pass before transmitting or receiving data.</p> |

1067 . Divisor Latch High Register Description

USART INTERRUPT ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7 | R/W | PTIME | 0 | <p>Programmable THRE Interrupt Mode Enable that can be written to only when THRE_MODE_USER = Enabled, always readable.</p> <p>This is used to enable/disable the generation of THRE Interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |
| 6:4 | R | Reserved | 0 | Reserved |
| 3 | R/W | EDSSI | 0 | <p>Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 2 | R/W | ELSI | 0 | <p>Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |
| 1 | R/W | ETBEI | 0 | <p>Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt.</p> <p>This is the third highest priority interrupt.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |
| 0 | R/W | ERBFI | 0 | <p>Enable Received Data Available Interrupt.</p> <p>This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFOs enabled).</p> <p>These are the second highest priority interrupts.</p> <ul style="list-style-type: none"> • 0 – disabled • 1 – enabled |

1068 . Interrupt Enable Register Description

USART INTERRUPT IDENTIFICATION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:6 | R | FIFOs Enabled (or FIFOSE) | 0 | <p>This is used to indicate whether the FIFOs are enabled or disabled.</p> <ul style="list-style-type: none"> • 00 – disabled • 11 – enabled |
| 5:4 | R | Reserved | 0 | Reserved |
| 3:0 | R | Interrupt ID (or IID) | 1 | <p>This indicates the highest priority pending interrupt which can be one of the following types:</p> <ul style="list-style-type: none"> • 0000 – modem status • 0001 – no interrupt pending • 0010 – THR empty • 0100 – received data available • 0110 – receiver line status • 0111 – busy detect • 1100 – character timeout <p>Bit 3 indicates an interrupt can only occur when the FIFOs are enabled and used to distinguish a Character Timeout condition interrupt.</p> |

1069 . Interrupt Identity Register Description

USART FIFO CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------------------------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:6 | W | RCVR Trigger (or RT) | 0 | <p>RCVR Trigger. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated.</p> <p>In auto flow control mode it is used to determine when the rts_n signal is de-asserted.</p> <p>It also determines when the dma_rx_req_n signal is asserted in certain modes of operation. The following trigger levels are supported:</p> <ul style="list-style-type: none"> • 00 – 1 character in the FIFO • 01 – FIFO 1/4 full • 10 – FIFO 1/2 full • 11 – FIFO 2 less than full |
| 5:4 | W | TX Empty Trigger (or TET) | 0 | <p>TX Empty Trigger. Writes have no effect when THRE_MODE_USER = Disabled.</p> <p>This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active.</p> <p>It also determines when the dma_tx_req_n signal is asserted when in certain modes of operation.</p> <p>The following trigger levels are supported:</p> <ul style="list-style-type: none"> • 00 – FIFO empty • 01 – 2 characters in the FIFO • 10 – FIFO 1/4 full • 11 – FIFO 1/2 full |
| 3 | W | DMA Mode (or DMAM) | 0 | <p>DMA Mode. This determines the DMA signaling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected (DMA_EXTRA = No).</p> <ul style="list-style-type: none"> • 0 – mode 0 • 1 – mode 1 |
| 2 | W | XMIT FIFO Reset (or XFIFOR) | 0 | <p>This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also de-asserts the DMA TX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA = YES).</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-----------------------------|-----------|--|
| 1 | W | RCVR FIFO Reset (or RFIFOR) | 0 | <p>This resets the control portion of the receive FIFO and treats the FIFO as empty. This also de-asserts the DMA RX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA = YES).</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |
| 0 | W | FIFO Enable (or FIFOE) | 0 | <p>This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs.</p> <p>Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.</p> |

1070 . FIFO Control Register Description

USART LINE CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7 | R/W | DLAB | 0 | <p>Divisor Latch Access Bit. If UART_16550_COMPATIBLE = NO, then writeable only when UART is not busy (USR[0] is 0); otherwise always writable, always readable.</p> <p>This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH/LPDLL and LPDLH) to set the baud rate of the UART.</p> <p>This bit must be cleared after initial baud rate setup in order to access other registers.</p> |
| 6 | R/W | Break (or BC) | 0 | <p>Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to 1, the serial output is forced to the spacing (logic 0) state.</p> <p>When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared.</p> <p>If SIR_MODE = Enabled and active (MCR[6] set to 1) the sir_out_n line is continuously pulsed.</p> <p>When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------|-----------|---|
| 5 | R/W | Stick Parity | 0 | <p>If UART_16550_COMPATIBLE = NO, then writeable only when UART is not busy (USR[0] is 0); otherwise always writable, always readable.</p> <p>This bit is used to force parity value. When PEN, EPS, and Stick Parity are set to 1, the parity bit is transmitted and checked as logic 0.</p> <p>If PEN and Stick Parity are set to 1 and EPS is a logic 0, then parity bit is transmitted and checked as a logic 1. If this bit is set to 0, Stick Parity is disabled.</p> |
| 4 | R/W | EPS | 0 | <p>Even Parity Select. If UART_16550_COMPATIBLE = NO, then writeable only when UART is not busy (USR[0] is 0); otherwise always writable, always readable.</p> <p>This is used to select between even and odd parity, when parity is enabled (PEN set to 1). If set to 1, an even number of logic 1s is transmitted or checked.</p> <p>If set to 0, an odd number of logic 1s is transmitted or checked.</p> |
| 3 | R/W | PEN | 0 | <p>Parity Enable. If UART_16550_COMPATIBLE = NO, then writeable only when UART is not busy (USR[0] is 0); otherwise always writable, always readable.</p> <p>This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively.</p> <ul style="list-style-type: none"> • 0 – parity disabled • 1 – parity enabled |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------------------|-----------|---|
| 2 | R/W | STOP | 0 | <p>Number of stop bits. If UART_16550_COMPATIBLE = NO, then writable only when UART is not busy (USR[0] is 0); otherwise always writable, always readable.</p> <p>This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to 0, one stop bit is transmitted in the serial data.</p> <p>If set to 1 and the data bits are set to 5 (LCR[1:0] set to 0) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted.</p> <p>Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit.</p> <ul style="list-style-type: none"> • 0 – 1 stop bit • 1 – 1.5 stop bits when DLS (LCR[1:0]) is 0, else 2 stop bit <p>NOTE: The STOP bit duration implemented by uart may appear longer due to idle time inserted between characters for some configurations and baud clock divisor values in the transmit direction.</p> |
| 1:0 | R/W | DLS (or CLS, as used in legacy) | 0 | <p>Data Length Select. If UART_16550_COMPATIBLE = NO, then writable only when UART is not busy (USR[0] is 0); otherwise always writable and readable.</p> <p>When DLS_E in LCR_EXT is set to 0, this register is used to select the number of data bits per character that the peripheral transmits and receives.</p> <p>The number of bits that may be selected are as follows:</p> <ul style="list-style-type: none"> • 00 – 5 bits • 01 – 6 bits • 10 – 7 bits • 11 – 8 bits |

1071 .Line Control Register Description

USART MODEM CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:7 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|---------------------|-----------|---|
| 6 | R/W | SIRE | 0 | <p>SIR Mode Enable. Writeable only when SIR_MODE = Enabled, always readable. This is used to enable/disable the IrDA SIR Mode features</p> <ul style="list-style-type: none"> • 0 – IrDA SIR Mode disabled • 1 – IrDA SIR Mode enabled <p>Note: To enable SIR mode, write the appropriate value to the MCR register before writing to the LCR register</p> |
| 5 | R/W | AFCE | 0 | <p>Auto Flow Control Enable. Writeable only when AFCE_MODE = Enabled, always readable. When FIFOs are enabled and the Auto Flow Control Enable (AFCE) bit is set,</p> <p>Auto Flow Control features are enabled</p> <ul style="list-style-type: none"> • 0 – Auto Flow Control Mode disabled • 1 – Auto Flow Control Mode enabled |
| 4 | R/W | LoopBack (or LB) | 0 | <p>LoopBack Bit. This is used to put the UART into a diagnostic mode for test purposes. If operating in UART mode (SIR_MODE != Enabled or not active, MCR[6] set to 0),</p> <p>data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally.</p> <p>If operating in infrared mode (SIR_MODE = Enabled AND active, MCR[6] set to 1), data on the sir_out_n line is held low, while serial data output is inverted and looped back to the sir_in line.</p> |
| 3 | R/W | OUT2 | 0 | <p>OUT2. This is used to directly control the user-designated Output2 (out2_n) output.</p> <p>The value written to this location is inverted and driven out on out2_n, that is:</p> <ul style="list-style-type: none"> • 0 – out2_n de-asserted (logic 1) • 1 – out2_n asserted (logic 0) <p>Note that in Loopback mode (MCR[4] set to 1), the out2_n output is held inactive high while the value of this location is internally looped back to an input.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 2 | R/W | OUT1 | 0 | <p>OUT1. This is used to directly control the user-designated Output1 (out1_n) output.</p> <p>The value written to this location is inverted and driven out on out1_n, that is:</p> <ul style="list-style-type: none"> • 0 – out1_n de-asserted (logic 1) • 1 – out1_n asserted (logic 0) <p>Note that in Loopback mode (MCR[4] set to 1), the out1_n output is held inactive high while the value of this location is internally looped back to an input.</p> |
| 1 | R/W | RTS | 0 | <p>Request to Send. This is used to directly control the Request to Send (rts_n) output.</p> <p>The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data.</p> <p>When Auto RTS Flow Control is not enabled (MCR[5] set to 0), the rts_n signal is set low by programming MCR[1] (RTS) to a high. In Auto Flow Control, AFCE_MODE = Enabled and active (MCR[5] set to 1) and FIFOs enable (FCR[0] set to 1), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold) only when the RTC Flow Trigger is disabled; otherwise it is gated by the receiver FIFO almost-full trigger, where “almost full” refers to two available slots in the FIFO (rts_n is inactive high when above the threshold). The rts_n signal is de-asserted when MCR[1] is set low.</p> <p>Note that in Loopback mode (MCR[4] set to 1), the rts_n output is held inactive high while the value of this location is internally looped back to an input.</p> |
| 0 | R/W | DTR | 0 | <p>Data Terminal Ready. This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is:</p> <ul style="list-style-type: none"> • 0 – dtr_n de-asserted (logic 1) • 1 – dtr_n asserted (logic 0) <p>The Data Terminal Ready output is used to inform the modem or data set that the UART is ready to establish communications. Note that in Loopback mode (MCR[4] set to 1), the dtr_n output is held inactive high while the value of this location is internally looped back to an input.</p> |

1072 . ModemControl Register Description

USART LINE STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7 | R | RFE | 0 | <p>Receiver FIFO Error bit. This bit is only relevant when FIFO_MODE != NONE AND FIFOs are enabled (FCR[0] set to 1). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO.</p> <ul style="list-style-type: none"> • 0 – no error in RX FIFO • 1 – error in RX FIFO <p>This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO.</p> |
| 6 | R | TEM _T | 1 | <p>Transmitter Empty bit. If in FIFO mode (FIFO_MODE != NONE) and FIFOs enabled (FCR[0] set to 1), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty.</p> <p>If in non-FIFO mode or FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.</p> |
| 5 | R | THRE | 1 | <p>Transmit Holding Register Empty bit. If THRE_MODE_USER = Disabled or THRE mode is disabled (IER[7] set to 0) and regardless of FIFO's being implemented/enabled or not,</p> <p>this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled.</p> <p>If THRE_MODE_USER = Enabled AND FIFO_MODE != NONE and both modes are active (IER[7] set to 1 and FCR[0] set to 1 respectively), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 4 | R | BI | 0 | <p>Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data.</p> <p>If in UART mode (SIR_MODE = Disabled), it is set whenever the serial input, sir_in, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop bits.</p> <p>If in infrared mode (SIR_MODE = Enabled), it is set whenever the serial input, sir_in, is continuously pulsed to logic '0' for longer than the sum of start time + data bits + parity + stop bits.</p> <p>A break condition on serial input causes one and only one character, consisting of all 0s, to be received by the UART.</p> <p>In FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO.</p> <p>Reading the LSR clears the BI bit. In non-FIFO mode, the BI indication occurs immediately and persists until the LSR is read.</p> <p>NOTE: If a FIFO is full when a break condition is received, a FIFO overrun occurs. The break condition and all the information associated with it—parity and framing errors—is discarded;</p> <p>any information that a break character was received is lost.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 3 | R | FE | 0 | <p>Framing Error bit. This is used to indicate the occurrence of a framing error in the receiver.</p> <p>A framing error occurs when the receiver does not detect a valid STOP bit in the received data.</p> <p>In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO.</p> <p>When a framing error occurs, the uart tries to resynchronize.</p> <p>It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit; that is, data, and/or parity and stop.</p> <p>It should be noted that the Framing Error (FE) bit (LSR[3]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]).</p> <p>This happens because the break character implicitly generates a framing error by holding the sin input to logic 0 for longer than the duration of a character.</p> <ul style="list-style-type: none"> • 0 – no framing error • 1 – framing error <p>Reading the LSR clears the FE bit.</p> |
| 2 | R | PE | 0 | <p>Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set.</p> <p>In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO.</p> <p>It should be noted that the Parity Error (PE) bit (LSR[2]) can be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]).</p> <p>In this situation, the Parity Error bit is set if parity generation and detection is enabled (LCR[3]=1) and the parity is set to odd (LCR[4]=0).</p> <ul style="list-style-type: none"> • 0 – no parity error • 1 – parity error <p>Reading the LSR clears the PE bit.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 1 | R | OE | 0 | <p>Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read.</p> <p>In the non-FIFO mode, the OE bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten.</p> <p>In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost.</p> <ul style="list-style-type: none"> • 0 – no overrun error • 1 – overrun error <p>Reading the LSR clears the OE bit.</p> |
| 0 | R | DR | 0 | <p>Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO.</p> <ul style="list-style-type: none"> • 0 – no data ready • 1 – data ready <p>This bit is cleared when the RBR is read in non-FIFO mode, or when the receiver FIFO is empty, in FIFO mode.</p> |

1073 . Line Status Register Description

USART MODEM STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7 | R | DCD | 0 | <p>Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n. This bit is the complement of dcd_n. When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set.</p> <ul style="list-style-type: none"> • 0 – dcd_n input is de-asserted (logic 1) • 1 – dcd_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] set to 1), DCD is the same as MCR[3] (Out2).</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 6 | R | RI | 0 | <p>Ring Indicator. This is used to indicate the current state of the modem control line ri_n. This bit is the complement of ri_n. When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set.</p> <ul style="list-style-type: none"> • 0 – ri_n input is de-asserted (logic 1) • 1 – ri_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] set to 1), RI is the same as MCR[2] (Out1)</p> |
| 5 | R | DSR | 0 | <p>Data Set Ready. This is used to indicate the current state of the modem control line dsr_n.</p> <p>This bit is the complement of dsr_n. When the Data Set Ready input (dsr_n) is asserted it is an indication that the modem or data set is ready to establish communications with the uart.</p> <ul style="list-style-type: none"> • 0 – dsr_n input is de-asserted (logic 1) • 1 – dsr_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] set to 1), DSR is the same as MCR[0] (DTR).</p> |
| 4 | R | CTS | 0 | <p>This bit is the complement of cts_n. When the Clear to Send input (cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the uart.</p> <ul style="list-style-type: none"> • 0 – cts_n input is de-asserted (logic 1) • 1 – cts_n input is asserted (logic 0) <p>In Loopback Mode (MCR[4] = 1), CTS is the same as MCR[1] (RTS).</p> |
| 3 | R | DDCD | 0 | <p>Delta Data Carrier Detect. This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on dcd_n since last read of MSR • 1 – change on dcd_n since last read of MSR <p>Reading the MSR clears the DDCD bit. In Loopback Mode (MCR[4] = 1), DDCD reflects changes on MCR[3] (Out2).</p> <p>Note, if the DDCD bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit is set when the reset is removed if the dcd_n signal remains asserted.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 2 | R | TERI | 0 | <p>Trailing Edge of Ring Indicator. This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on ri_n since last read of MSR • 1 – change on ri_n since last read of MSR <p>Reading the MSR clears the TERI bit. In Loopback Mode (MCR[4] = 1), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.</p> |
| 1 | R | DDSR | 0 | <p>Delta Data Set Ready. This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on dsr_n since last read of MSR • 1 – change on dsr_n since last read of MSR <p>Reading the MSR clears the DDSR bit. In Loopback Mode (MCR[4] = 1), DDSR reflects changes on MCR[0] (DTR).</p> <p>Note, if the DDSR bit is not set and the dsr_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDSR bit is set when the reset is removed if the dsr_n signal remains asserted.</p> |
| 0 | R | DCTS | 0 | <p>Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read.</p> <ul style="list-style-type: none"> • 0 – no change on cts_n since last read of MSR • 1 – change on cts_n since last read of MSR <p>Reading the MSR clears the DCTS bit. In Loopback Mode (MCR[4] = 1), DCTS reflects changes on MCR[1] (RTS).</p> <p>Note, if the DCTS bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit is set when the reset is removed if the cts_n signal remains asserted.</p> |

1074 . Modem Status Register Description

USART SCRATCHPAD REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------------|-----------|--|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R/W | Scratchpad Register | 0 | This register is for programmers to use as a temporary storage space. It has no defined purpose in the UART. |

1075 . Scratchpad Register Description

USART FDR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------|-----------|---|
| 31:0 | R/W | FDR register | 0 | This register is just written and read but not used anywhere in functionality. This is kept to make it compatible with USART standard |

1076 .FD Register Description

USART HDEN REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------|-----------|--|
| 31:2 | R | Reserved | 0 | Reserved |
| 1 | R/W | tx_mode/rx_mode | 0 | This signal is valid when full_duplex_mode is disabled 0 – tx_mode 1 – rx_mode |
| 0 | R/W | full_duplex_mode | 0 | 0 – Full duplex mode enable 1 – Full duplex mode disable |

1077 .HDEN Register Description

USART SMCR REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------|-----------|--|
| 31:6 | R | Reserved | 0 | Reserved |
| 5 | R/W | start_stop_en | 0 | 1 – Enable start stop 0 – Disable start stop |
| 4 | R/W | conti_clk_mode0 | | 1 – Continuous clock mode 0 – Non-continuous clock mode |
| 3:2 | R | Reserved | 0 | Reserved |
| 1 | R/W | mst_mode | 0 | 1 – MST mode 0 – Non-MST mode |
| 0 | R/W | sync_mode | 0 | 1 – Sync mode 0 – Non-Sync mode |

1078 .SMCR Register Description

USART FIFO ACCESS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------|-----------|--|
| 0 | R/W | FIFO Access | 0 | <p>Writes have no effect when FIFO_ACCESS = No, always readable. This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are implemented and enabled.</p> <p>When FIFOs are not implemented or not enabled it allows the RBR to be written by the master and the THR to be read by the master.</p> <ul style="list-style-type: none"> • 0 – FIFO access mode disabled • 1 – FIFO access mode enabled <p>Note, that when the FIFO access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFOs are treated as empty.</p> |

1079 . FIFO Access Register Description

USART TRANSMIT FIFO READ REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|--------------------|-----------|---|
| 31:8 | R | Reserved | 0 | Reserved |
| 7:0 | R | Transmit FIFO Read | 0 | <p>These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, reading this register gives the data at the top of the transmit FIFO.</p> <p>Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO.</p> <p>When FIFOs are not implemented or not enabled, reading this register gives the data in the THR.</p> |

1080 . Transmit FIFO Read Register Description

USART RECEIVE FIFO WRITE REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|-------------|
| 31:10 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|---|
| 9 | W | RFFE | 0 | <p>Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, this bit is used to write framing error detection information to the receive FIFO.</p> <p>When FIFOs are not implemented or not enabled, this bit is used to write framing error detection information to the RBR.</p> |
| 8 | W | RFPE | 0 | <p>Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, this bit is used to write parity error detection information to the receive FIFO.</p> <p>When FIFOs are not implemented or not enabled, this bit is used to write parity error detection information to the RBR.</p> |
| 7:0 | W | RFWD | 0 | <p>Receive FIFO Write Data. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to 1).</p> <p>When FIFOs are implemented and enabled, the data that is written to the RFWD is pushed into the receive FIFO.</p> <p>Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFOs are not implemented or not enabled, the data that is written to the RFWD is pushed into the RBR.</p> |

1081 . Receive FIFO Write Register Description

USART STATUS REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:5 | R | Reserved | 0 | Reserved |
| 4 | R | RFF | 0 | <p>Receive FIFO Full. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the receive FIFO is completely full.</p> <ul style="list-style-type: none"> • 0 – Receive FIFO not full • 1 – Receive FIFO Full <p>This bit is cleared when the RX FIFO is no longer full.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 3 | R | RFNE | 0 | <p>Receive FIFO Not Empty. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the receive FIFO contains one or more entries.</p> <ul style="list-style-type: none"> • 0 – Receive FIFO is empty • 1 – Receive FIFO is not empty <p>This bit is cleared when the RX FIFO is empty.</p> |
| 2 | R | TFE | 0 | <p>Transmit FIFO Empty. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the transmit FIFO is completely empty.</p> <ul style="list-style-type: none"> • 0 – Transmit FIFO is not empty • 1 – Transmit FIFO is empty <p>This bit is cleared when the TX FIFO is no longer empty.</p> |
| 1 | R | TFNF | 0 | <p>Transmit FIFO Not Full. This bit is only valid when FIFO_STAT = YES. This is used to indicate that the transmit FIFO is not full.</p> <ul style="list-style-type: none"> • 0 – Transmit FIFO is full • 1 – Transmit FIFO is not full <p>This bit is cleared when the TX FIFO is full.</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------|-----------|--|
| 0 | R | BUSY | 0 | <p>UART Busy. This bit is valid only when UART_16550_COMPATIBLE = NO and indicates that a serial transfer is in progress; when cleared, indicates that the uart is idle or inactive.</p> <ul style="list-style-type: none"> • 0 – uart is idle or inactive • 1 – uart is busy (actively transferring data) <p>This bit will be set to 1 (busy) under any of the following conditions:</p> <ol style="list-style-type: none"> 1. Transmission in progress on serial interface 2. Transmit data present in THR, when FIFO access mode is not being used (FAR = 0) and the baud divisor is non-zero ($\{DLH,DLL\}$ does not equal 0) when the divisor latch access bit is 0 (LCR.DLAB = 0) 3. Reception in progress on the interface 4. Receive data present in RBR, when FIFO access mode is not being used (FAR = 0) <p>NOTE: It is possible for the UART Busy bit to be cleared even though a new character may have been sent from another device.</p> <p>That is, if the uart has no data in THR and RBR and there is no transmission in progress and a start bit of a new character has just reached the uart.</p> <p>This is due to the fact that a valid start is not seen until the middle of the bit period and this duration is dependent on the baud divisor that has been programmed.</p> <p>If a second system clock has been implemented (CLOCK_MODE = Enabled), the assertion of this bit is also delayed by several cycles of the slower clock.</p> |

1082 . UART Status Register Description

USART TRANSMIT FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|---------------------|--------|---------------------|-----------|---|
| 31:FIFO_ADDR_WIDT H | R | Reserved | 0 | Reserved |
| FIFO_ADDR_WIDTH:0 | R | Transmit FIFO Level | 0 | This indicates the number of data entries in the transmit FIFO. |

1083 . Transmit FIFO Level Register Description

USART RECEIVE FIFO LEVEL REGISTER

| Bit | Access | Function | POR Value | Description |
|--------------------|--------|--------------------|-----------|--|
| 31:FIFO_ADDR_WIDTH | R | Reserved | 0 | Reserved |
| FIFO_ADDR_WIDTH:0 | R | Receive FIFO Level | 0 | This indicates the number of data entries in the receive FIFO. |

1084 . Receive FIFO Level Register Description

USART SOFTWARE RESET REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|--|
| 31:3 | R | Reserved | 0 | Reserved |
| 2 | W | XFR | 0 | <p>XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO. This resets the control portion of the transmit FIFO and treats the FIFO as empty.</p> <p>This also de-asserts the DMA TX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA = YES).</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |
| 1 | W | RFR | 0 | <p>RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO. This resets the control portion of the receive FIFO and treats the FIFO as empty.</p> <p>This also de-asserts the DMA RX request and single signals when additional DMA handshaking signals are selected (DMA_EXTRA = YES).</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |
| 0 | W | UR | 0 | UART Reset. This asynchronously resets the uart and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains are reset. |

1085 . Software Reset Register Description

USART SHADOW REQUEST TO SEND REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | Shadow Request to Send | 0 | <p>This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to perform a read-modify-write on the MCR.</p> <p>This is used to directly control the Request to Send (rts_n) output.</p> <p>The Request To Send (rts_n) output is used to inform the modem or data set that the uart is ready to exchange data.</p> <p>When Auto RTS Flow Control is not enabled (MCR[5] = 0), the rts_n signal is set low by programming MCR[1] (RTS) to a high.</p> <p>In Auto Flow Control, AFCE_MODE = Enabled and active (MCR[5] = 1) and FIFOs enable (FCR[0] = 1), the rts_n output is controlled in the same way,</p> <p>but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold) only when RTC Flow Trigger is disabled; otherwise</p> <p>it is gated by the receiver FIFO almost-full trigger, where “almost full” refers to two available slots in the FIFO (rts_n is inactive high when above the threshold).</p> <p>Note that in Loopback mode (MCR[4] = 1), the rts_n output is held inactive-high while the value of this location is internally looped back to an input.</p> |

1086 . Shadow Request to Send Register Description

USART SHADOW BREAK CONTROL REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------------------|-----------|--|
| 0 | R/W | Shadow Break Control Register | 0 | <p>This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to perform a read modify write on the LCR. This is used to cause a break condition to be transmitted to the receiving device.</p> <p>If set to 1, the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared.</p> <p>If SIR_MODE = Enabled and active (MCR[6] = 1) the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver.</p> |

1087 . Shadow Break Control Register Description

USART SHADOW DMA MODE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | Shadow DMA Mode | 0 | <p>This is a shadow register for the DMA mode bit (FCR[3]). This can be used to remove the burden of having to store the previously written value to the FCR</p> <p>in memory and having to mask this value so that only the DMA Mode bit gets updated. This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected (DMA_EXTRA = NO).</p> <ul style="list-style-type: none"> • 0 – mode 0 • 1 – mode 1 |

1088 . Shadow DMA Mode Register Description

USART SHADOW FIFO ENABLE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------|-----------|---|
| 0 | R/W | Shadow FIFO Enable | 0 | <p>Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the FIFO enable bit gets updated. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs.</p> <p>If this bit is set to 0 (disabled) after being enabled then both the XMIT and RCVR controller portion of FIFOs are reset.</p> |

1089 . Shadow FIFO Enable Register Description

USART SHADOW RCVR TRIGGER REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------------------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | Shadow RCVR Trigger0 | 0 | <p>Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the RCVR trigger bit gets updated.</p> <p>This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated.</p> <p>It also determines when the dma_rx_req_n signal is asserted when DMA Mode (FCR[3]) = 1.</p> <p>The following trigger levels are supported:</p> <ul style="list-style-type: none"> • 00 – 1 character in the FIFO • 01 – FIFO ¼ full • 10 – FIFO ½ full • 11 – FIFO 2 less than full |

1090 . Shadow RCVR Trigger Register Description

USART SHADOW TX EMPTY TRIGGER REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:2 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|-------------------------|-----------|---|
| 1:0 | R/W | Shadow TX Empty Trigger | 0 | <p>Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]). This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the TX empty trigger bit gets updated.</p> <p>This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. The following trigger levels are supported:</p> <ul style="list-style-type: none"> • 00 – FIFO empty • 01 – 2 characters in the FIFO • 10 – FIFO ¼ full • 11 – FIFO ½ full |

1091 . Shadow TX Empty Trigger Register Description

USART HALT TX REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|---|
| 31:1 | R | Reserved | 0 | Reserved |
| 0 | R/W | HALT TX | 0 | <p>This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled.</p> <ul style="list-style-type: none"> • 0 – Halt TX disabled • 1 – Halt TX enabled <p>Note, if FIFOs are implemented and not enabled, the setting of the halt TX register has no effect on operation.</p> |

1092 . HALT TX Register Description

USART DMA SOFTWARE ACKNOWLEDGE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-----------|-------------|
| 31:1 | R | Reserved | 0 | Reserved |

| Bit | Access | Function | POR Value | Description |
|-----|--------|--------------------------|-----------|--|
| 0 | W | DMA Software Acknowledge | 0 | <p>This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition.</p> <p>For example, if the DMA disables the channel, then the uart should clear its request.</p> <p>This causes the TX request, TX single, RX request and RX single signals to de-assert.</p> <p>Note that this bit is 'self-clearing'. It is not necessary to clear this bit.</p> |

1093 . DMA Software Acknowledgment Register Description

USART COMPONENT PARAMETER REGISTER

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------------------|-----------|---|
| 31:24 | R | Reserved | 0 | Reserved |
| 23:16 | R | FIFO_MODE | 0x01 | <p>0x00 = 0</p> <p>0x01 = 16</p> <p>0x02 = 32</p> <p>to</p> <p>0x80 = 2048</p> <p>0x81- 0xff = reserved</p> |
| 15:14 | R | Reserved | 0 | Reserved |
| 13 | R | DMA_EXTRA | 0x1 | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 12 | R | UART_ADD_ENCODED_PARAMS0x0 | | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 11 | R | SHADOW | 0x0 | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 10 | R | FIFO_STAT | 0x1 | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 9 | R | FIFO_ACCESS | 0x0 | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 8 | R | ADDITIONAL_FEAT | 0x1 | <p>0 – FALSE</p> <p>1 – TRUE</p> |
| 7 | R | SIR_LP_MODE | 0x1 | <p>0 – FALSE</p> <p>1 – TRUE</p> |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------|-----------|--|
| 6 | R | SIR_MODE | 0x1 | 0 – FALSE 1 – TRUE |
| 5 | R | THRE_MODE | 0x1 | 0 – FALSE 1 – TRUE |
| 4 | R | AFCE_MODE | 0x1 | 0 – FALSE 1 – TRUE |
| 3:2 | R | Reserved | 0 | Reserved |
| 1:0 | R | APB_DATA_WIDTH | 0x2 | 00 – 8 bits 01 – 16 bits 10 – 32 bits 11 – reserved |

1094 . Component Parameter Register Description

UART COMPONENT VERSION REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|------------------------|------------|--|
| 31:0 | R | UART Component Version | 0x3430302a | ASCII value for each number in the version, followed by *. |

1095 . UART Component Version Register Description

USART COMPONENT TYPE REGISTER

| Bit | Access | Function | POR Value | Description |
|------|--------|---------------|------------|--|
| 31:0 | R | Peripheral ID | 0x44570110 | This register contains the peripherals identification code |

1096 . Component Type Register Description

17 MCU ULP Peripherals

17.1 IR_Decoder

17.1.1 General Description

This is a general purpose Infrared receiver, which can decode all IR protocol(RC5,NEC,.. etc) with Software intervention.

17.1.2 Features

- IR Decoder is General purpose Infrared Receiver.
- IR Decoder clocked by a low power 32 KHz RC clock.
- Programmable Active and Sleep window duration for IR data monitoring.
- Wakeup source to exiting low power sleep state.
- It will take IR pulses from external IR sensor connected through GPIO's

17.1.3 Functional Description

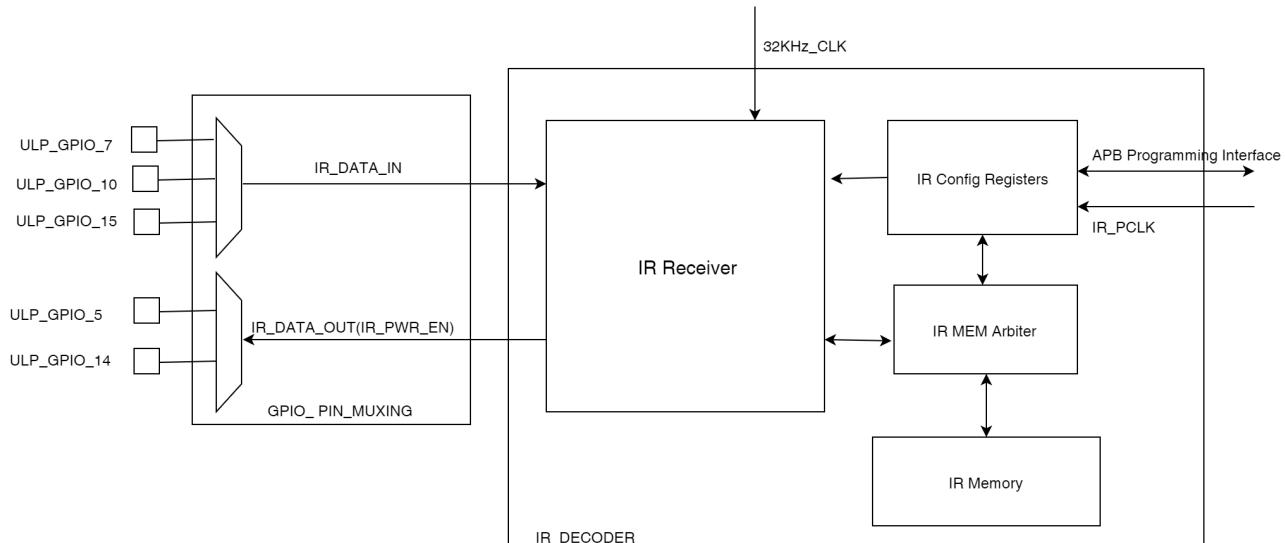
IR Decoder sample the input data starting from each rising and falling edge where a bit period counter (period counter) is reset. At each sample point clocked ad 32KHz clk the period counter is incremented. This gives a number that represents the period of each high or low part of the signal.



Note

The period counter is 15 bits long hence there is no counter wrap issue for a long input sequence. Received IR data is stored in a memory for software to read data and interpret IR Command.

Block Diagram



Block Diagram of IR Decoder

Clock Selection

There are two clock going to IR Decoder, One is a low power 32KHz CLK sourced from a 32KHz RC clock and a IR_PCLK for programming configuration registers.

Please refer to ULPSS Clock Architecture section for configuring IR_PCLK and 32KH_CLK.

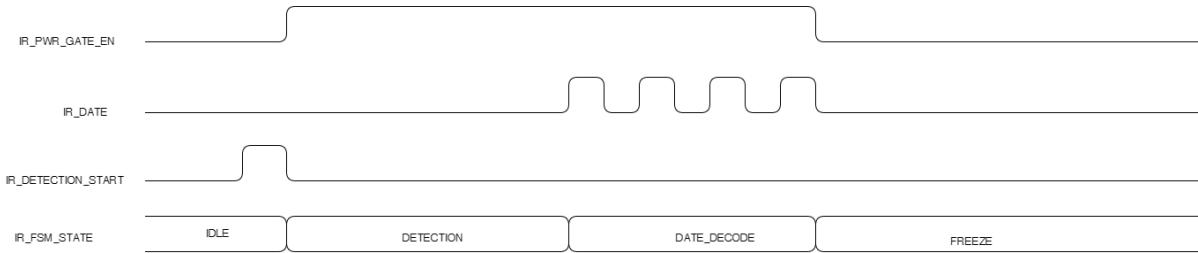
GPIO Configuration

Please refer to GPIO pin mixing section for allocating GPIO's for IR detection.

Operating Mode

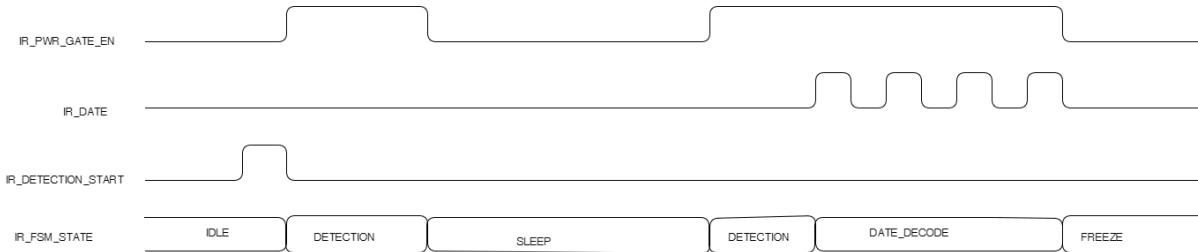
There are two operating mode on IR decode

- **Continuous Mode :** In this mode, the external IR sensor power gate enable is turned on always so that IR data pattern can be monitored continuously. This can be achieved by setting register bit EN_CONT_IR_DET in register address **IR_CONFIG**.



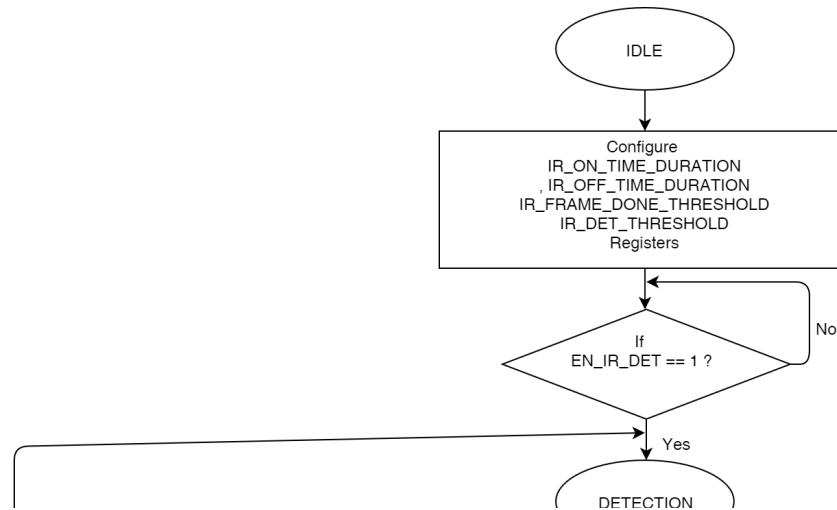
Continuous Mode

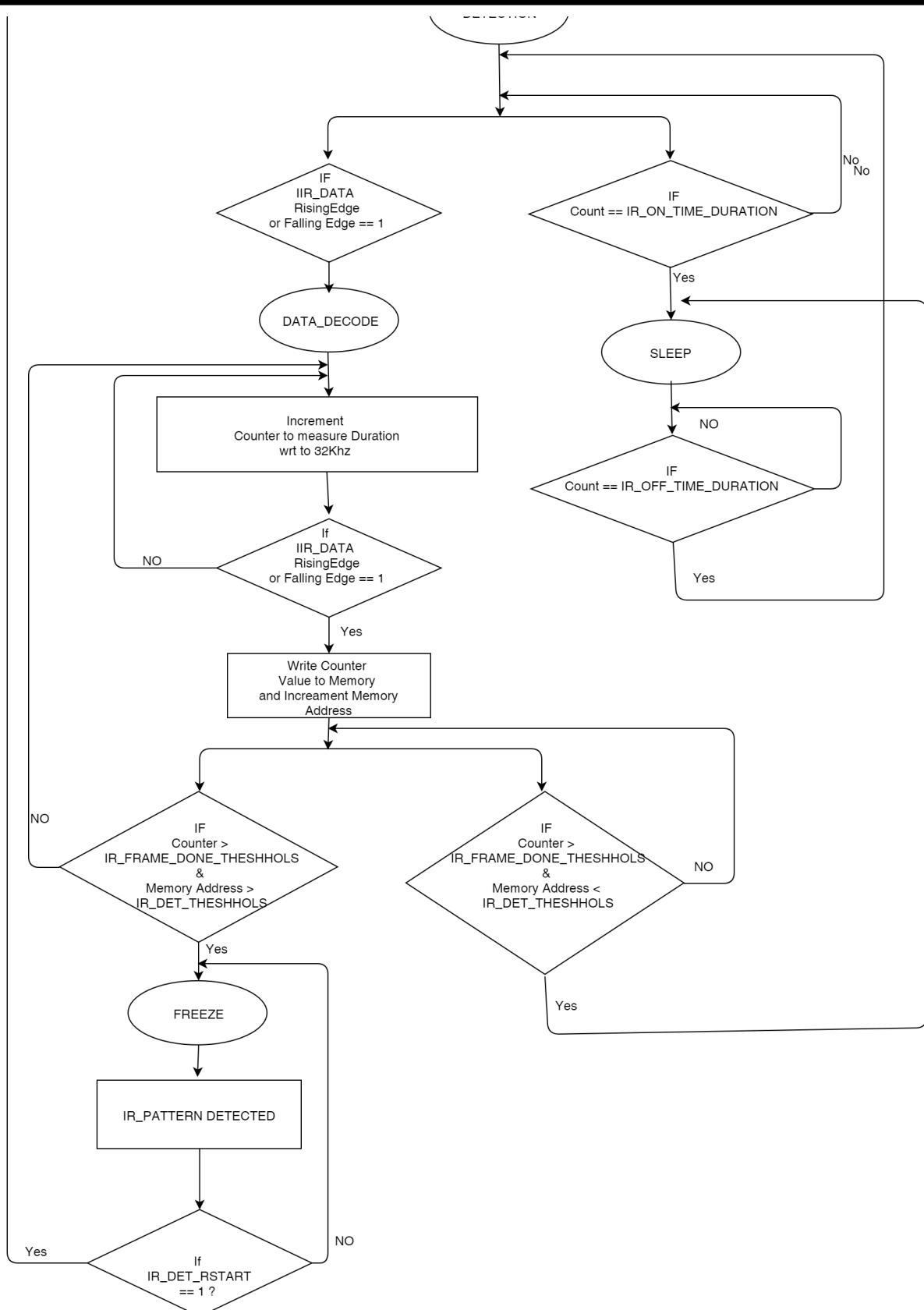
- **Window Mode:** In this mode, There are 2 states, Sleep and Active. In Sleep state, external IR sensor power gate enable is turned off and in Active state external IR sensor power gate enable is turned on for monitoring IR Data.



Window Mode

IR Decoding Flow Diagram





IR Decoding Flow Diagram

Programming Sequence for Configuration

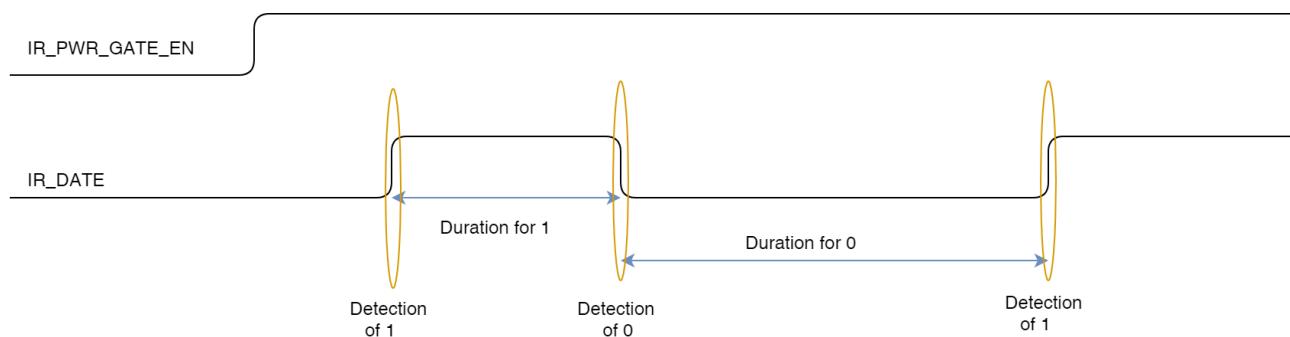
1. Configure 32KHz and IR_PCLK by Clock Selection section.
2. Configure **IR_OFF_TIME_DURATION** register
3. Configure **IR_ON_TIME_DURATION** register
4. Configure **IR_FRAME_DONE_THRESHOLD** register
5. Configure **IR_DET_THRESHOLD** register
6. Enable **EN_CLK_IR_CORE** bit in register **IR_CONFIG**
7. Based on the Mode of Operation (Continuous Mode or Window Mode), Configure **EN_CONT_IR_DET** bit in register **IR_CONFIG**
8. Enable **EN_IR_DET** bit in register **IR_CONFIG** to trigger IR Decoder

Programming Sequence for Reading IR Data

1. Up On IR detection in NVIC , wait for 100us.
2. Read **IR_DATA_MEM_DEPTH** bit in register **IR_MEM_READ** , which indicated how many memory location need to read.
3. Write **IR_MEM_ADDR** bits in register **IR_MEM_ADDR_ACCESS** with value 0
4. Enable **IR_MEM_RD_EN** bit in register **IR_MEM_ADDR_ACCESS**
5. Read **IR_MEM_DATA_OUT** bit in register **IR_MEM_READ** to read IR data.
6. Increment **IR_MEM_ADDR** bits in register **IR_MEM_ADDR_ACCESS**
7. Repeat steps 4 and 5.
8. Repeat step 6 till **IR_MEM_ADDR** reaches **IR_DATA_MEM_DEPTH**
9. For re-initiating IR decoding, program **IR_DET_RSTSTART** bit in register **IR_CONFIG**.
10. Wait for new IR detection in NVIC.

IR Memory Data Representation

IR memory Data is a 16-bit(15:0) value. Where Bit-15, represents polarity of IR Pulse (1 or 0) as described in the below diagram and Bits-14 to Bits-0 represents duration for pulse with respect to clock ticks to 32KHz clock.



IR Memory Data Representation

17.1.4 Register Summary

Base Address: 0x2404_0C00

| Register Name | Offset | Description | |
|--------------------------------------|--------|-------------|--|
| IR_OFF_TIME_DURATION | 0x00 | | |
| IR_ON_TIME_DURATION | 0x04 | | |

| Register Name | Offset | Description |
|-------------------------|--------|-------------|
| IR_FRAME_DONE_THRESHOLD | 0x08 | |
| IR_DET_THRESHOLD | 0x0C | |
| IR_CONFIG | 0x10 | |
| IR_MEM_ADDR_ACCESS | 0x14 | |
| IR_MEM_READ | 0x18 | |

1097 . IR Decoder Register Summary

17.1.5 Register Description

IR_OFF_TIME_DURATION

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------|-------------|---|
| 31:17 | | | | |
| 16:0 | R/W | IR_OFF_TIME_DURATION | 0 | IR Sleep duration timer value. Programmable value for OFF duration for power cycling on External IR Sensor.Count to be programmed wrt to clock ticks of 32KHz clock. |

1098 . IR_OFF_TIME_DURATION

IR_ON_TIME_DURATION

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|--|
| 31:12 | | | | |
| 11:0 | R/W | IR_ON_TIME_DURATION | 0 | IR Detection duration timer value. Programmable value for ON duration for power cycling on External IR Sensor.Count to be programmed wrt to clock ticks of 32KHz clock. |

1099 . IR_OFF_TIME_DURATION

IR_FRAME_DONE_THRESHOLD

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------|-------------|---|
| 31:15 | | | | |
| 14:0 | R/W | IR_FRAME_DONE_THRESHOLD | 0 | Count with respect to 32KHz clock after not more toggle are expected to a given pattern . |

1100 . IR_FRAME_DONE_THRESHOLD

IR_DET_THRESHOLD

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------------|-------------|---|
| 31:7 | | | | |
| 6:0 | R/W | IR_DET_THRESHOLD0 OLD | 0 | Minimum Number of edges to detected during on-time failing which IR detection is re-stated. |

1101 .IR_DET_THRESHOLD

IR_CONFIG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|---|
| 31:17 | | | | |
| 16 | R/W | SREST_IR_CORE | 0 | Soft reset IR core block |
| 15:9 | | | | |
| 8 | R/W | EN_CONT_IR_DET | 0 | Enable Continuous IR detection. When enabled there will be no power cycling on External IR Sensor. |
| 7:3 | | | | |
| 2 | R/W | EN_CLK_IR_CORE | 0 | Enable 32KHz clock to IR Core 1 - Clock gating Disable 0 - Clock Gating Enable |
| 1 | R/W | IR_DET_RSTART | 0 | Enable IR detection Re-start Logic. 1 - Re-Starts IR detection. Note : Self clearing register |
| 0 | R/W | EN_IR_DET | 0 | Enable IR detection Logic 1 - Enable 0 - Disable |

1102 .IR_CONFIG

IR_MEM_ADDR_ACCESS

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------|-------------|-----------------------|
| 31:10 | | | | |
| 9 | R/W | IR_MEM_RD_EN | 0 | IR memory read enable |
| 8:7 | | | | |
| 6:0 | R/W | IR_MEM_ADDR | 0 | IR read address |

1103 .IR_MEM_ADDR_ACCESS

IR_MEM_READ

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|---|
| 31 | | | | |
| 30:24 | R | IR_DATA_MEM_DEPT H | -- | Indicated valid number of IR Address in the memory to be read |
| 23:16 | | | | |
| 15:0 | R | IR_MEM_DATA_OUT | -- | IR Read data from memory |

1104 . IR MEM READ

17.2 Sensor Data Collector (SDC)

17.2.1 General Description

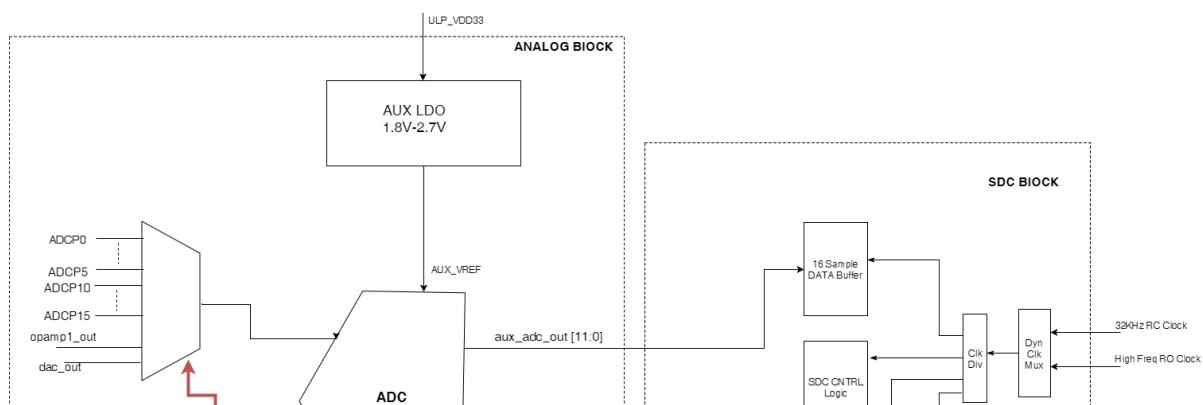
Sensor Data Collector(SDC) is a low energy sensor sampled collection mode where AUX-ADC is used for sample collection and it also has option for utilizes on chip analog peripheral such as OPAMP to perform measurements. The result from measurement will be stored in a buffer of 16 samples to be used by MCU for further processing.

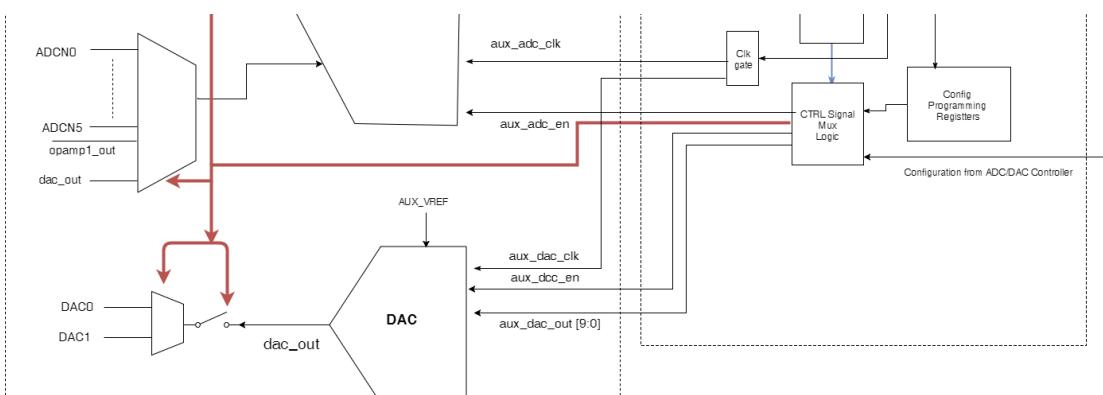
17.2.2 Features

- Low energy sensor block which used ADC which utilizes on chip OPAMP for measurement
- Support for 14 Signal-Ended mode Configuration
 - 12 External Inputs
 - 2 Internal Inputs
 - Opamp
 - DAC output as voltage reference
- Support for 7 Differential Configuration.
- Stores 16 sampled of ADC date in internal Buffer for 1 Channel.
- Stores 8 sampled of ADC date in internal Buffer for 2 Channel.
- Stores 4 sampled of ADC date in internal Buffer for 4 Channel.
- Initiates Interrupt/wakeup once Internal Buffer threshold has reached.

17.2.3 Functional Description

Block Diagram





Block Diagram of Sensor Data Collector (SDC)

GPIO MUXING

Please refer to **ULP GPIO Pin Multiplexing** section in **Pin Multiplexing for MCU and WiseMCU** products chapter for more information on Analog Pin Muxing.

Clock Section

Configure SDC clock source should be selected for initiating programming for configuration register. There are 2 clock option for SDC, 32MHz RC clock and High Frequency RO clock.

The following procedure should be used for clock source selection.

- Configure **SDCSS_CLK_SEL** register to select 32MHz RC clock in register address **MCU_FSM_REF_CLK**
- Enable **SDCSS_CLK_EN** in register address **MCU_FSM_REF_CLK**
- Enable **SDCSS_STATIC_CLK_EN** in register address **MCU_FSM_REF_CLK**

Once the configuration register are programmed. Program **SDCSS_STATIC_CLK_EN** as '0' in register address **MCU_FSM_REF_CLK** to disable free running clock. Clock will be enabled once the configured trigger even occurs and it will remain till sampling operation is completed.

Programming Sequence

1. Configure SDC clock source to initiate programming for SDC Configuration.
 - a. Configure **SDCSS_CLK_SEL** register to select 32MHz RC clock in register address **MCU_FSM_REF_CLK**
 - b. Enable **SDCSS_CLK_EN** in register address **MCU_FSM_REF_CLK**
 - c. Enable **SDCSS_STATIC_CLK_EN** in register address **MCU_FSM_REF_CLK**
2. Configure SDCSS Parameters
 - a. Program **SAMP_THRESH** bits in register **SDC_GEN_CONFIG_1** for number of sample required after which wakeup is initiated.
 - b. Program **NUM_CH_SEL** bits in register **SDC_GEN_CONFIG_2** for selecting number of channel that need to be swepted for sample collection.
 - c. Program **CNT_TRIG_EVNT** bits in register **SDC_GEN_CONFIG_4**, which indicated the number of trigger even that need to be skipped.
 - d. Select Trigger even on which AUX-ADC data should be sampled by programming **SAMP_TRIG_SEL** bits in register **SDC_GEN_CONFIG_3**.
3. AUX-ADC Should be calibrated before triggering SDC block. Refer to Analog to Digital converter section in Analog peripheral chapter for calibration procedure.
4. Configure Aux-ADC parameters

- a. Program SDC_AUXADC_INPUT_N_SEL_CH1 , SDC_AUXADC_INPUT_P_SEL_CH1 , SDC_AUXADC_DIFF_MODE_CH1 bits in register **SDC_AUXADC_CONFIG_1** for Channel-1 ADC selection. Refer to Analog to Digital converter section in Analog peripheral chapter for AUX-ADC Channel selection.
 - b. Program SDC_AUXADC_INPUT_N_SEL_CH2 , SDC_AUXADC_INPUT_P_SEL_CH2 , SDC_AUXADC_DIFF_MODE_CH2 bits in register **SDC_AUXADC_CONFIG_2** for Channel-2 ADC selection. Refer to Analog to Digital converter section in Analog peripheral chapter for AUX-ADC Channel selection.
 - c. Program SDC_AUXADC_INPUT_N_SEL_CH3 , SDC_AUXADC_INPUT_P_SEL_CH3 , SDC_AUXADC_DIFF_MODE_CH3 bits in register **SDC_AUXADC_CONFIG_3** for Channel-3 ADC selection. Refer to Analog to Digital converter section in Analog peripheral chapter for AUX-ADC Channel selection.
 - d. Program SDC_AUXADC_INPUT_N_SEL_CH4 , SDC_AUXADC_INPUT_P_SEL_CH4 , SDC_AUXADC_DIFF_MODE_CH4 bits in register **SDC_AUXADC_CONFIG_4** for Channel-4 ADC selection. Refer to Analog to Digital converter section in Analog peripheral chapter for AUX-ADC Channel selection.
 - e. Program SDC_AUXADC_EN bits in register **SDC_AUXADC_CONFIG_1** for enabling AUX-ADC through SDC block.
 - f. Program SDC_ADC_CONFIG_EN bits in register **SDC_AUXADC_CONFIG_1** for enabling SDC AUX-ADC Configuration .
5. Configure OP-AMP parameters if Op-AMP output need to feed to ADC.
- a. Program SDC_OPAMP_IN_N_SEL in register **SDC_AUXOPAMP_CONFIG_1** for selecting N-Channel of OP-AMP1. N Chennel is common for all 4 Channel. Refer to OPAMP section in Analog peripheral chapter for OPAMP Channel selection.
 - b. Program SDC_OPAMP_IN_P_SEL_CH1, SDC_OPAMP_EN_CH1 in register **SDC_AUXOPAMP_CONFIG_1** for selecting P-Channel of OP-AMP1.
 - c. Program SDC_OPAMP_IN_P_SEL_CH2, SDC_OPAMP_EN_CH2 in register **SDC_AUXOPAMP_CONFIG_2** for selecting P-Channel of OP-AMP1.
 - d. Program SDC_OPAMP_IN_P_SEL_CH3, SDC_OPAMP_EN_CH3 in register **SDC_AUXOPAMP_CONFIG_3** for selecting P-Channel of OP-AMP1.
 - e. Program SDC_OPAMP_IN_P_SEL_CH4, SDC_OPAMP_EN_CH4 in register **SDC_AUXOPAMP_CONFIG_4** for selecting P-Channel of OP-AMP1.
 - f. Program SDC_OPAMP_CONFIG_EN in register **SDC_AUXOPAMP_CONFIG_1** for enabling SDC OPAMP Configuration .
6. Configure ADC Clock Division factor to generated clock to ADC by programming SDC_CLK_DIV bits in register **SDC_GEN_CONFIG_3**.
7. Enable Data Collection
- a. Program SDC_SAMP_EN bits in register **SDC_GEN_CONFIG_2**
8. DDisable **SDCSS_STATIC_CLK_EN** in register address **MCU_FSM_REF_CLK**
9. Initiate Sleep sequence for entering PS1 with SDC a wakeup source.
10. Upon Wakeup read register SDC_DATA_REG0 to SDC_DATA_REG15. These register will contain sample collected from AUX-ADC and sampled need moved to ULP SRAM Banks.
11. Re-Initiating the SDC block
- a. Program RST_WRT_PTR in register **SDC_GEN_CONFIG_1** to reset the write pointer of FIFO.
 - b. Program INTR_STATUS_CLEAR in register **SDC_GEN_CONFIG_0** to clear interrupt to NVIC.
12. For Collecting mode sample Re-Initiate Sleep Sequence.

17.2.4 Register Summary

Base Address: 0x2404_8100

| Register Name | Offset | Description |
|---------------------------------|--------|-------------|
| MCU_FSM_REF_CLK | 0x1C | |

[1105 . SDC Clock Register Summary](#)

Base Address: 0x2404_2400

| Register Name | Offset | Description |
|-----------------------|--------|-------------|
| SDC_GEN_CONFIG_0 | 0x00 | |
| SDC_GEN_CONFIG_1 | 0x04 | |
| SDC_GEN_CONFIG_2 | 0x08 | |
| SDC_GEN_CONFIG_3 | 0x12 | |
| SDC_AUXADC_CONFIG_1 | 0x18 | |
| SDC_AUXDAC_CONFIG_1 | 0x1C | |
| SDC_AUXLDO_CONFIG | 0x20 | |
| SDC_AUXOPAMP_CONFIG_1 | 0x24 | |
| SDC_AUXADC_CONFIG_2 | 0x28 | |
| SDC_AUXADC_CONFIG_3 | 0x2C | |
| SDC_AUXADC_CONFIG_4 | 0x30 | |
| SDC_AUXOPAMP_CONFIG_2 | 0x34 | |
| SDC_DATA_REG0 | 0x38 | |
| SDC_DATA_REG1 | 0x3C | |
| SDC_DATA_REG2 | 0x40 | |
| SDC_DATA_REG3 | 0x44 | |
| SDC_DATA_REG4 | 0x48 | |
| SDC_DATA_REG5 | 0x4C | |
| SDC_DATA_REG6 | 0x50 | |
| SDC_DATA_REG7 | 0x54 | |
| SDC_DATA_REG8 | 0x58 | |
| SDC_DATA_REG9 | 0x5C | |
| SDC_DATA_REG10 | 0x60 | |
| SDC_DATA_REG11 | 0x64 | |
| SDC_DATA_REG12 | 0x68 | |
| SDC_DATA_REG13 | 0x6C | |
| SDC_DATA_REG14 | 0x70 | |
| SDC_DATA_REG15 | 0x74 | |

1106 . SDC Register Summary

17.2.5 Register Description

MCU_FSM_REF_CLK

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|---|
| 31 | R/W | SDCSS_STATIC_CLK_EN | 0 | To enable static clk for sensor data collector subsystem |
| 30 | R/W | SDCSS_CLK_EN | 0 | To enable dynamic clock for sdcss |
| 29:28 | R/W | SDCSS_CLK_SEL | 0 | SDCSS Clock Selection to be used for Configuration 01 – 32MHz RC Clock 10 – High Frequency RO Clock |
| 27:25 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------------|-------------|--|
| 24 | R/W | ULPSS_REF_CLK_CLNR_1 ON | | Clk cleaner On signal for ulpss ref clock |
| 23 | R/W | ULPSS_REF_CLK_CLNR_0 OFF | | clk cleaner Off signal for ulpss ref clock |
| 22:19 | | | | |
| 18:16 | R/W | ULPSS_REF_CLK_SEL | 1 | Dynamic Reference Clock Mux select of ULPSS 0 : Clock will be gated at dynamic mux output of ULPSS 1 : ulp_32mhz_rc_byp_clk 2 : ulp_32mhz_rc_clk 3 : rf_ref_clk 4 : mems_ref_clk 5 : ulp_20mhz_ringosc_clk 6 : ulp_doubler_clk 7 : ref_byp_clk to TASS |
| 15 | | | | |
| 14:12 | R/W | TASS_REF_CLK_SEL | 1 | Dynamic Reference Clock Mux select of TASS controlled by M4. 0 : Clock will be gated at dynamic mux output of TASS 1 : ulp_32mhz_rc_byp_clk 2 : ulp_32mhz_rc_clk 3 : rf_ref_clk 4 : mems_ref_clk 5 : ulp_20mhz_ringosc_clk 6 : ref_byp_clk to TASS |
| 11:9 | | | | |
| 8 | R/W | M4SS_REF_CLK_CLNR_1 ON | 1 | Enable clk cleaner for m4ss reference clock |
| 7 | R/W | M4SS_REF_CLK_CLNR_0 OFF | 0 | Disable signal for m4ss reference clock |
| 6:3 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------|-------------|--|
| 2:0 | R/W | M4SS_REF_CLK_SEL | 1 | Dynamic Reference Clock Mux select of M4SS 0 : Clock will be gated at dynamic mux output of M4SS 1 : ulp_32mhz_rc_byp_clk 2 : ulp_32mhz_rc_clk 3 : rf_ref_clk 4 : mems_ref_clk 5 : ulp_20mhz_ringosc_clk 6 : ulp_doubler_clk 7 : ref_byp_clk to TASS |

1107 . MCU FSM REF CLK

SDC_GEN_CONFIG_0

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:1 | | | | |
| 0 | R/W | INTR_STATUS_CLEAR | 0 | Reading this register will returns SDC's interrupt status. Writing 1 to this register will clear interrupt. |

1108 . SDC GEN CONFIG 0

SDC_GEN_CONFIG_1

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------|-------------|---|
| 31:9 | | | | |
| 8:5 | R/W | SAMP_THRESH | 0 | Number of data sampled to be collected from Aux-ADC and stored in Buffer before interrupt is raised/wakeup is initiated |
| 4:1 | R | WRT_PTR | 0 | Write pointer Value |
| 0 | R/W | RST_WRT_PTR | 0 | Writing 1 to this register will resets the write pointer so that new samples can be filled in Buffer. |

1109 . SDC GEN CONFIG 1

SDC_GEN_CONFIG_2

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------|-------------|--|
| 31:4 | | | | |
| 3:1 | R/W | NUM_CH_SEL | 0 | <p>Number of Channels to be used</p> <p>0 - Single channel is used</p> <p>1 - Two channels are used</p> <p>2 - Three channels are used</p> <p>3 - Four channels are used</p> <p>Ex: If 0 , Only 1 sample is stored into the buffer for Channel-1's configuration on trigger event.</p> <p>Ex: If 1, 2 sampled will be stored into the buffer. 1st sample will be for Channel-1's configuration and 2nd sampled will be for Channel-2's Configuration</p> |
| 0 | R/W | SDC_SAMP_EN | 0 | <p>SDC Data Sampling mode</p> <p>1 - Enable</p> <p>0 - Disable</p> |

1110 . SDC_GEN_CONFIG_2

SDC_GEN_CONFIG_3

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|--|
| 31:21 | | | | |
| 20:11 | R/W | SDC_CLK_DIV | 0 | <p>SDCSS clock division factor</p> <p>0: Passthrough</p> <p>N: Division by 2N</p> <p>Note: Value should not exceed 160 value equal for generating 20Khz clock.</p> |

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------|-------------|--|
| 10:1 | R/W | CNT_TRIG_EVNT | 0 | <p>The register will indicate in which trigger event AUX-ADC Data will sampled</p> <p>0 - Sample Data on every Trigger Event</p> <p>1 - Sample Data on every alternate Trigger Event</p> <p>2 - Sample Data on every 3rd Trigger Event</p> <p>3 - Sample Data on every 4th Trigger Event</p> <p>and so on.</p> |
| 0 | R/W | SAMP_TRIG_SEL | 0 | <p>The register is used to select the trigger event on which AUX-ADC Data is sampled</p> <p>1 – 1ms Pulse will be used as Trigger event</p> <p>0 – 1sec Pulse will be used as Trigger event</p> <p>Note : Calendar/RTC Block should be configured for generating 1 milli-sec and 1 sec pulse/</p> |

1111 . SDC_GEN_CONFIG_3

SDC_AUXADC_CONFIG_1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------------|-------------|---|
| 31:12 | | | | |
| 11 | R/W | SDC_ADC_CONFIG_EN | 0 | On Enabling this register, SDC ADC Configuration will be Applied. |
| 10 | R/W | SDC_AUXADC_EN | 0 | AUXADC Enable from SDC Block |
| 9 | R/W | SDC_AUXADC_DIFF_MODE_CH1 | 0 | Enable Differential Mode in AUX ADC for Channel -1 |
| 8:5 | R/W | SDC_AUXADC_INPUT_N_SEL_CH1 | 0 | <p>AUXADC's Negative Input Mux Select for Channel-1</p> <p>Please refer to Input selection section in Analog to Digital conversion chapter for programming options.</p> |
| 4:0 | R/W | SDC_AUXADC_INPUT_P_SEL_CH1 | 0 | <p>AUXADC's Positive Input Mux Select for Channel-1</p> <p>Please refer to Input selection section in Analog to Digital conversion chapter for programming options.</p> |

1112 . SDC_AUXADC_CONFIG_1

SDC_AUXDAC_CONFIG_1

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|---------------------|--------------------|---|
| 31:1 2 | | | | |
| 14 | R/W | SDC_DAC_CONFIG_EN | 0 | On Enabling this register, SDC DAC Configuration will be Applied. |
| 13:4 | R/W | SDC_DAC_DATA | 0 | SDC Aux DAC Data |
| 3 | | | | |
| 2 | R/W | SDC_DAC_OUT_MUX_SEL | 0 | <p>Programming register for choosing GPIO in which DAC Output is connected</p> <p>0 – Connect DAC Output to DAC_0 (ULP_GPIO_4)</p> <p>1 – Connect DAC Output to DAC_1 (ULP_GPIO_15)</p> <p>Please refer to Analog Functions mapping section in Pin Multiplexing for MCU and WiSemCU chapter for more description.</p> |
| 1 | R/W | SDC_DAC_OUT_MUX_EN | 0 | <p>Enable signal for Connecting DAC Output to GPIO</p> <p>0 – DAC Output is Not Connected to GPIO.</p> <p>1 – DAC Output is Connected to GPIO. Please refer to SDC_DAC_OUT_MUX_SEL to know to which GPIO DAC output is connected</p> |
| 0 | R/W | SDC_DAC_EN | 0 | Enable signal DAC |

1113 . SDC_AUXDAC_CONFIG_1

SDC_AUXLDO_CONFIG

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|----------------------|--------------------|---|
| 31:12 | | | | |
| 7 | R/W | SDC_AUXLDO_CONFIG_EN | 0 | SDC Aux LDO Configuration Control Enable |
| 6 | R/W | SDC_AUXLDO_EN | 0 | Turn-On AUX LDO 1 - Turn-ON 0 - Turn-OFF |
| 5 | R/W | SDC_AUXLDO_BYP_EB | 0 | Configure AUXLDO in Bypass mode. When Enabled, Ouput supply of LDO will be same as Input supply. |
| 4 | | | | |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|----------------------|--------------------|--|
| 3:0 | R/W | SDC_AUXLDO_VOLT_CTRL | 0 | SDC AUX LDO Voltage Control Selection 0 - 1.60v 1 - 1.68v 2 - 1.76v 3 - 1.84v 4 - 1.92v 5 - 2.00v 6 - 2.08v 7 - 2.16v 8 - 2.24v 9 - 2.32v 10 - 2.4v 11 - 2.48v 12 - 2.56v 13 - 2.64v 14 - 2.72v 15 - 2.90v |

1114 . SDC_AUXDAC_CONFIG_1

SDC_AUXOPAMP_CONFIG_1

| Bi | Ac | ce | ss | Function | Reset Value | Description |
|-----------|-----------|-----------|-----------|-----------------------|--------------------|---|
| 31 | | | | | | |
| 30 | R/W | | | SDC_OPAMP_CONFIG_0_EN | 0 | On Enabling this register, SDC OPAMP Configuration will be Applied. |
| 29:26 | R/W | | | SDC_VREF_MUX_4_SEL | 0 | Selection register for choosing Voltage reference to external use on GPIO(ULP_GPIO_15) 1 – AUX_Vref 0 – 1.0v |
| 28 | R/W | | | SDC_VREF_MUX_3_SEL | 0 | Selection register for choosing Voltage reference to external use on GPIO(ULP_GPIO_4) 1 – AUX_Vref 0 – 1.0v |
| 27 | R/W | | | SDC_VREF_MUX_2_SEL | 0 | Selection register for choosing Voltage reference to external use on GPIO(ULP_GPIO_3) 1 – AUX_Vref 0 – 1.0v |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------------|--------------------|---|
| 26 | R/W | SDC_VREF_MUX_1_SEL | 0 | Selection register for choosing Voltage reference to external use on GPIO(ULP_GPIO_1) 1 – AUX_Vref 0 – 1.0v |
| 25 | | | | |
| 24 | R/W | SDC_VREF_MUX_4_EN | 0 | Enable for connecting Low Drive strength Voltage reference to GPIO for external use. 1 – Connect Voltage reference to GPIO(ULP_GPIO_15) 0 – No Connect to GPIO Please refer to Analog Functions mapping section in Pin Multiplexing for MCU and WiSeMCU chapter for more description. |
| 23 | R/W | SDC_VREF_MUX_3_EN | 0 | Enable for connecting Low Drive strength Voltage reference to GPIO for external use. 1 – Connect Voltage reference to GPIO(ULP_GPIO_4) 0 – No Connect to GPIO Please refer to Analog Functions mapping section in Pin Multiplexing for MCU and WiSeMCU chapter for more description. |
| 22 | R/W | SDC_VREF_MUX_3_EN | 0 | Enable for connecting Low Drive strength Voltage reference to GPIO for external use. 1 – Connect Voltage reference to GPIO(ULP_GPIO_3) 0 – No Connect to GPIO Please refer to Analog Functions mapping section in Pin Multiplexing for MCU and WiSeMCU chapter for more description. |
| 21 | R/W | SDC_VREF_MUX_1_EN | 0 | Enable for connecting Low Drive strength Voltage reference to GPIO for external use. 1 – Connect Voltage reference to GPIO(ULP_GPIO_1) 0 – No Connect to GPIO Please refer to Analog Functions mapping section in Pin Multiplexing for MCU and WiSeMCU chapter for more description. |
| 20 | R/W | SDC_OPAMP_OUT_MUX_SEL | 0 | Configuration register for connecting OPAMP1 output to GPIO 0 – Connect OPAMP1 Output to ULP_GPIO_4 1 – Connect OPAMP1 Output to ULP_GPIO_15 Please refer to Analog Functions mapping section in Pin Multiplexing for MCU and WiSeMCU chapter for more description. |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------|-------------|---|
| 19:16 | R/W | SDC_OPAMP_IN_P_SEL_L_CH1 | 0 | Configuration register for selecting P Input of OPAMP1 for Channel-1 Please refer to Input Selection in OPAMP chapter. |
| 15:13 | R/W | SDC_OPAMP_IN_N_SEL_L | 0 | Configuration register for selecting N Input of OPAMP1. This selection will be common for all Channels Please refer to Input Selection in OPAMP chapter. |
| 12 | R/W | SDC_OPAMP_OUT_MUX_EN | 0 | On Configuring this register, OPAMP1 Output will be connected to GPIO 1 – Output is connected to GPIO 0 – Output is Not connected. |
| 11 | R/W | SDC_OPAMP_RES_TO_OUT_VDD | 0 | Configuration register for Connecting R2 Resistor Ladder input 0 – Connect R2 to Opamp1 Output 1 – Connect R2 to VDD(AUX_Vref) |
| 10:8 | R/W | SDC_OPAMP_RES_MUX_SEL | 0 | Configuration register for Connecting R1 Resistor Ladder input Please refer to Input Selection in OPAMP chapter. |
| 7 | R/W | SDC_OPAMP_RES_BANK_EN | 0 | Configuration register for controlling Resistor Bank of OPAMP 0 – Disable 1 – Enable |
| 6:4 | R/W | SDC_OPAMP_R2_SEL | 0 | Configuration for Resistor Ladder R2 of OPAMP1 for controlling its gain. Please refer to Resistor banks |
| 3:2 | R/W | SDC_OPAMP_R1_SEL | 0 | Configuration for Resistor Ladder R1 of OPAMP1 for controlling its gain. Please refer to Resistor banks |
| 1 | R/W | SDC_OPAMP_LP_MODE | 0 | Configuration of OPAMP1 Operation mode 0 – Normal mode of Operation 1 – Low power mode of Operation |
| 0 | R/W | SDC_OPAMP_ENABLE_CH1 | 0 | Enable signal for turning OPAMP to be used for Channel-1 Operation 1 – Enable 0 – Disable |

1115 . SDC_AUXOPAMP_CONFIG_1

SDC_AUXADC_CONFIG_2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:10 | | | | |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|---------------------------------|--------------------|--|
| 9 | R/W | SDC_AUXADC_DIFF_MO DE_CH2 | 0 | Enable Differential Mode in AUX ADC for Channel -2 1 - AUX ADC Operates in Differential mode 0 - AUX ADC Operates in Single Ended mode |
| 8:5 | R/W | SDC_AUXADC_INPUT_N_0 SEL_CH2 | | AUXADC's Negative Input Mux Select for Channel-2 Please refer to Input selection section in Analog to Digital conversion chapter for programming options. |
| 4:0 | R/W | SDC_AUXADC_INPUT_P_0 SEL_CH2 | | AUXADC's Positive Input Mux Select for Channel-2 Please refer to Input selection section in Analog to Digital conversion chapter for programming options. |

1116 . SDC_AUXADC_CONFIG_2

SDC_AUXADC_CONFIG_3

| Bi | Acc | Function | Reset | Description |
|-----------|------------|---------------------------------|--------------|--|
| 31:10 | | | | |
| 9 | R/W | SDC_AUXADC_DIFF_MOD E_CH3 | 0 | Enable Differential Mode in AUX ADC for Channel 3 1 - AUX ADC Operates in Differential mode 0 - AUX ADC Operates in Single Ended mode |
| 8:5 | R/W | SDC_AUXADC_INPUT_N_0 SEL_CH3 | | AUXADC's Negative Input Mux Select for Channel-3 Please refer to Input selection section in Analog to Digital conversion chapter for programming options. |
| 4:0 | R/W | SDC_AUXADC_INPUT_P_0 SEL_CH3 | | AUXADC's Positive Input Mux Select for Channel-3 Please refer to Input selection section in Analog to Digital conversion chapter for programming options. |

1117 . SDC_AUXADC_CONFIG_3

SDC_AUXADC_CONFIG_4

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--------------------|
| 31:10 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------------------------------|-------------|--|
| 9 | R/W | SDC_AUXADC_DIFF_MO DE_CH4 | 0 | Enable Differential Mode in AUX ADC for Channel -4 1 - AUX ADC Operates in Differential mode 0 - AUX ADC Operates in Single Ended mode |
| 8:5 | R/W | SDC_AUXADC_INPUT_N 0 _SEL_CH4 | | AUXADC's Negative Input Mux Select for Channel-4 Please refer to Input selection section in Analog to Digital conversion chapter for programming options. |
| 4:0 | R/W | SDC_AUXADC_INPUT_P 0 _SEL_CH4 | | AUXADC's Positive Input Mux Select for Channel-4 Please refer to Input selection section in Analog to Digital conversion chapter for programming options. |

1118 . SDC_AUXADC_CONFIG_4

SDC_AUXOPAMP_CONFIG_2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------------|-------------|---|
| 31:15 | | | | |
| 14:11 | R/W | SDC_OPAMP_IN_P_SEL_CH4 | 0 | Configuration register for selecting P Input of OPAMP1 for Channel-4 Please refer to Input Selection in OPAMP chapter. |
| 10 | R/W | SDC_OPAMP_EN_CH4 | 0 | Enable signal for turning OPAMP to used for Channel-4 Operation 1 – Enable 0 – Disable |
| 9:6 | R/W | SDC_OPAMP_IN_P_SEL_CH3 | 0 | Configuration register for selecting P Input of OPAMP1 for Channel-3 Please refer to Input Selection in OPAMP chapter. |
| 5 | R/W | SDC_OPAMP_EN_CH3 | 0 | Enable signal for turning OPAMP to used for Channel-4 Operation 1 – Enable 0 – Disable |
| 4:1 | R/W | SDC_OPAMP_IN_P_SEL_CH2 | 0 | Configuration register for selecting P Input of OPAMP1 for Channel-2 Please refer to Input Selection in OPAMP chapter. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------|-------------|--|
| 0 | R/W | SDC_OPAMP_EN_CH2 | 0 | Enable signal for turning OPAMP to used for Channel-2 Operation 1 – Enable 0 – Disable |

1119 . SDC_AUXOPAMP_CONFIG_2

SDC_DATA_REG0

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_0 | 0 | Channel iD for sample 0 |
| 11:0 | R | SDC_DATA_SAMPLE_0 | 0 | Sample 0 collected from Sensor through Aux ADC. |

1120 . SDC_DATA_REG0

SDC_DATA_REG1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_1 | 0 | Channel iD for sample 1 |
| 11:0 | R | SDC_DATA_SAMPLE_1 | 0 | Sample 1 collected from Sensor through Aux ADC. |

1121 . SDC_DATA_REG1

SDC_DATA_REG2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_2 | 0 | Channel iD for sample 2 |
| 11:0 | R | SDC_DATA_SAMPLE_2 | 0 | Sample 2 collected from Sensor through Aux ADC. |

1122 . SDC_DATA_REG2

SDC_DATA_REG3

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_3 | 0 | Channel iD for sample 3 |
| 11:0 | R | SDC_DATA_SAMPLE_3 | 0 | Sample 3 collected from Sensor through Aux ADC. |

1123 . SDC_DATA_REG3

SDC_DATA_REG4

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_4 | 0 | Channel iD for sample 4 |
| 11:0 | R | SDC_DATA_SAMPLE_4 | 0 | Sample 4 collected from Sensor through Aux ADC |

1124 . SDC_DATA_REG4

SDC_DATA_REG5

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_5 | 0 | Channel iD for sample 5 |
| 11:0 | R | SDC_DATA_SAMPLE_5 | 0 | Sample 5 collected from Sensor through Aux ADC |

1125 . SDC_DATA_REG5

SDC_DATA_REG6

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_6 | 0 | Channel iD for sample 6 |
| 11:0 | R | SDC_DATA_SAMPLE_6 | 0 | Sample 6 collected from Sensor through Aux ADC |

1126 . SDC_DATA_REG6

SDC_DATA_REG7

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:16 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 13:12 | R | SMP_ID_CH_7 | 0 | Channel iD for sample 7 |
| 11:0 | R | SDC_DATA_SAMPLE_7 | 0 | Sample 7 collected from Sensor through Aux ADC. |

1127 . SDC_DATA_REG7

SDC_DATA_REG8

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_8 | 0 | Channel iD for sample 8 |
| 11:0 | R | SDC_DATA_SAMPLE_8 | 0 | Sample 8 collected from Sensor through Aux ADC. |

1128 . SDC_DATA_REG8

SDC_DATA_REG9

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_9 | 0 | Channel iD for sample 9 |
| 11:0 | R | SDC_DATA_SAMPLE_9 | 0 | Sample 9 collected from Sensor through Aux ADC. |

1129 . SDC_DATA_REG9

SDC_DATA_REG10

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | channel_id_10 | 0 | Channel iD for sample 10 |
| 11:0 | R | SDC_DATA_SAMPLE_10 | 0 | Sample 10 collected from Sensor through Aux ADC. |

1130 . SDC_DATA_REG10

SDC_DATA_REG11

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------|-------------|--------------------------|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_11 | 0 | Channel iD for sample 11 |

| Bit | Access | Function | Reset Value | Description |
|------|--------|--------------------|-------------|--|
| 11:0 | R | SDC_DATA_SAMPLE_11 | 0 | Sample 11 collected from Sensor through Aux ADC. |

1131 . SDC DATA REG11

SDC_DATA_REG12

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_12 | 0 | Channel iD for sample 12 |
| 11:0 | R | SDC_DATA_SAMPLE_12 | 0 | Sample 12 collected from Sensor through Aux ADC. |

1132 . SDC DATA REG12

SDC_DATA_REG13

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_13 | 0 | Channel iD for sample 13 |
| 11:0 | R | SDC_DATA_SAMPLE_13 | 0 | Sample 13 collected from Sensor through Aux ADC. |

1133 . SDC DATA REG13

SDC_DATA_REG14

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_14 | 0 | Channel iD for sample 14 |
| 11:0 | R | SDC_DATA_SAMPLE_14 | 0 | Sample 14 collected from Sensor through Aux ADC. |

1134 . SDC DATA REG14

SDC_DATA_REG15

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|--|
| 31:16 | | | | |
| 13:12 | R | SMP_ID_CH_15 | 0 | Channel ID for sample 15 |
| 11:0 | R | SDC_DATA_SAMPLE_15 | 0 | Sample 15 collected from Sensor through Aux ADC. |

1135 . SDC DATA REG15

17.3 AUX ADC/DAC Controller

17.3.1 General Description

The ADC-DAC Controller works on a ADC with a resolution of 12bits at 10Maga sample per second when ADC reference Voltage is greater than 2.8v or 5Maga sample per second when ADC reference Voltage is less than 2.8v.

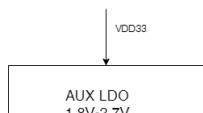
17.3.2 AUX ADC Features

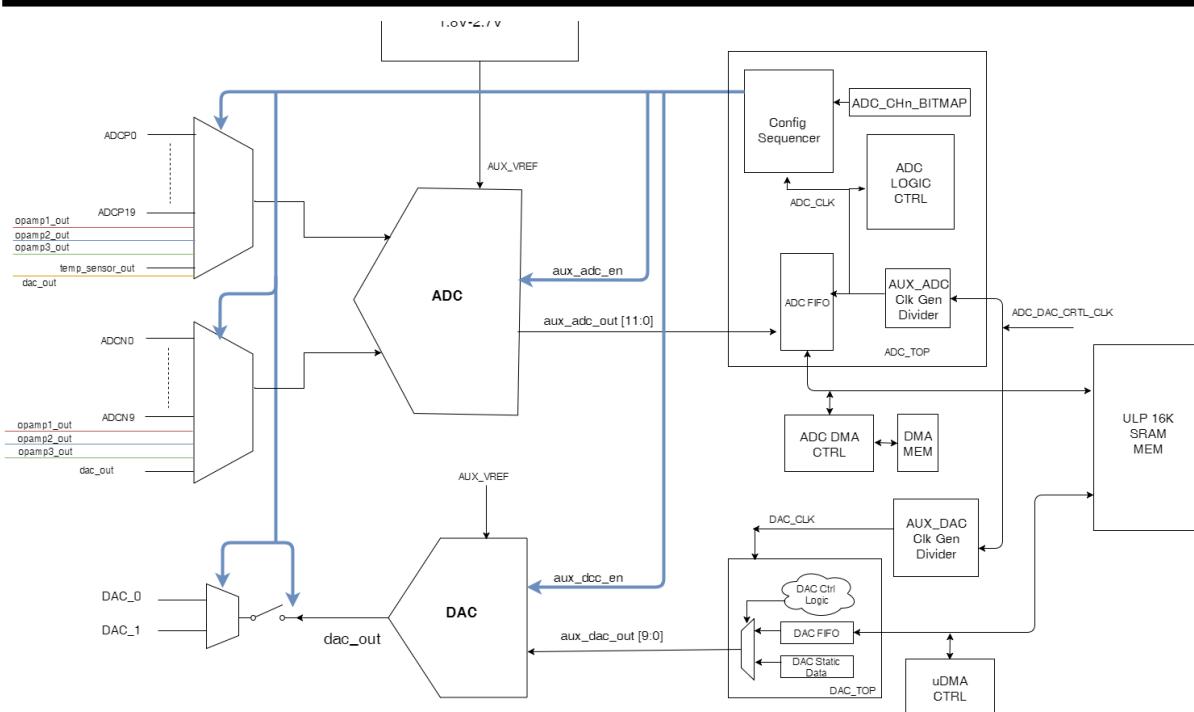
- Programmable clock divider to getting ADC_CLK with Option of controlling its Duty-cycle
- 12 bit ADC Output in 2's complement representation
- GPIO's in High Power mode mode for ADC Operation
 - Signal Ended Mode
 - 20 External configuration selection
 - 5 Internal configuration selection
 - Internal Temperature sensor
 - 3 Opamps Outputs
 - DAC output for internal reference
 - Differential Mode
 - 10 external differential mode configuration selection
 - 4 Internal configuration selection.
 - 3 Opamps Outputs
 - DAC output for internal reference
- GPIO's in Low Power mode for ADC Operation
 - Signal Ended Mode
 - 14 External configuration selection.
 - 5 Internal configuration selection.
 - Internal Temperature sensor.
 - 3 Opamps Outputs
 - DAC output for internal reference
 - Differential Mode
 - 6 external differential mode configuration selection.
 - 4 Internal configuration selection.
 - 3 Opamps Outputs
 - DAC output for internal reference
- 10MHz to 32KHz allowed ADC_CLK
- Configurable DAM to support 16 channels for storing AUXADC data. Date is ULP SRAM.
- Measurement range 0 to AUXADC_VREF(1.8v to 3.3v)

17.3.3 AUX DAC Features

- Programmable clock divider to getting DAC_CLK
- 10bit resolution
- Single ended DAC
- Monotonic by design
- Max sampling frequency is 5MHz for DAC_CLK
- Functional Description

Block Diagram





Block Diagram of AUX ADC/DAC Controller

GPIO MUXING

Please refer to **ULP GPIO Pin Multiplexing** section in **Pin Multiplexing for MCU and WiseMCU** products chapter for more information Analog Pin Muxing.

Clock Selection

For configuring the AUX ADC-DAC controller, clock source need to be selected before initiating configuration sequence. Please refer to ULPSS Clock Architecture section for selection options

AUXADC CLK Selection

Clock to ADC is generated by programming register **AUXADC_CLK_DIV_FAC** and enabling the clock circuit to ADC by programming **EN_ADC_CLK** bit in register **AUXADC_CTRL_1**.

In register **AUXADC_CLK_DIV_FAC**, there are programming option for controlling On-Duration of the ADC clock and Total duration of ADC clock. With these option we can generate a duty-cycled clock for better time optimization. On-Duration of the ADC clock is controlled by **ADC_CLK_ON_DUR** bits and Total duration of clock is controlled by programming **ADC_CLK_DIV_FAC** bits.

Ex: If the AUX-ADC controller clock is selected as 32MHz and the required ADC clock is 4MHz with 50% duty cycle. Then program 4 to **ADC_CLK_ON_DUR** and 8 to **ADC_CLK_DIV_FAC**.

AUXDAC CLK Selection

Clock to DAC is generated by programming register **AUXDAC_CLK_DIV_FAC**.

AUXDAC Controller

There are three operating modes AUXDAC controller.

- Static Mode
- FIFO Mode
- Reference voltage for ADC

Static Mode

In static mode, DAC will give out constant voltage output for a programmed DAC.

Programming Sequence :

1. Program **AUXDAC_CLK_DIV_FAC** register to select clock frequency on which DAC should be operating.
2. Program **AUXDAC_DYN_MODE** bit in register **AUXDAC_CONFIG_1** to 0.
3. Program DAC parameter in AUXDAC_CONIG_1 register.
 - a. AUXDAC_OUT_MUX_EN bit when set will enable DAC output to be seen on GPIO.
 - b. AUXDAC_OUT_MUX_SEL bit will choose on which GPIO DAC Output data is feed.
 - c. AUXDAC_DATA_S value will decide DAC output voltage in static mode.
- d. AUXDAC_EN_S bit will Enable DAC operation in Static mode.

FIFO Mode

DAC controller can be kept in FIFO mode to play continuously digital word on DAC. This mode can be used for playing Single tone waveform in DAC.

Programming Sequence:

1. Program **AUXDAC_CLK_DIV_FAC** register to select clock frequency on which DAC should be operating.
2. Program **AUXDAC_DYN_MODE** bit in register **AUXDAC_CONFIG_1** to 1.
3. Configure DAC FIFO Parameter in register **AUXDAC_CTRL_1**
 - a. Program DAC_FIFO_THRESHOLD, DAC_FIFO_FLUSH, ENDAC_FIFO_CONFIG bit in register **AUXDAC_CONFIG_1**.
4. Load DAC digital in ULP SRAM and Configure **uDAM controller** to send data from ULP memories to AUXDAC_DATA register. Please refer to uDAM Controller section for configuration details.
5. Program **AUXDAC_EN_F** bit in register **AUXDAC_CONFIG_1** to 1 Enable DAC operation in FIFO mode.

Note: In FIFO mode, DAC output is available only on AGPIO4

Reference Voltage Mode

DAC controller can be kept in a mode where DAC output will be used as reference voltage to ADC channel input or OPAMP channel input.

Programming Sequence:

1. Program **AUXDAC_CLK_DIV_FAC** register to select clock frequency on which DAC should be operating.
2. Program **AUXDAC_DYN_MODE** bit in register **AUXDAC_CONFIG_1** to 1.
3. Program AUXDAC_EN_F bit to Enable DAC operation.
4. Configure CHANNEL_BITMAP registers, for generating reference voltage for the required channel by programming DAC Digital words.
5. Program DAC_CTRL_TO_ADC bit in register AUXDAC_CTRL_1

ADC Controller

There are three operating modes AUXADC controller.

- Static Mode
- DMA Mode

Static Mode

In static mode, ADC will sample data from a signal configured channel.

Programming Sequence:

DMA Mode

Channel BITMAP

| BITMAP Location | Signal Name |
|-----------------|-----------------------|
| 16 - 0 | Reserved |
| 21 - 17 | aux_adc_ip_sel |
| 25 - 22 | aux_adc_in_sel |
| 26 | aux_adc_diffmode |
| 27 | Reserved |
| 28 | aux_dac_en |
| 29 | aux_dac_out_mux_en |
| 30 | aux_dac_out_mux_sel |
| 40 - 31 | aux_dac_data |
| 41 | opamp1_enable |
| 42 | opamp1_lp_mode |
| 44 - 43 | opamp1_R1_sel |
| 47 - 45 | opamp1_R2_sel |
| 48 | opamp1_en_res_bank |
| 51 - 49 | opamp1_res_mux_sel |
| 52 | opamp1_res_to_out_vdd |
| 53 | opamp1_out_mux_en |
| 56 - 54 | opamp1_inn_sel |
| 60 - 57 | opamp1_inp_sel |
| 61 | opamp1_out_mux_sel |
| 62 | opamp2_enable |
| 63 | opamp2_lp_mode |
| 65 - 64 | opamp2_R1_sel |
| 68 - 66 | opamp2_R2_sel |
| 69 | opamp2_en_res_bank |
| 72 - 70 | opamp2_res_mux_sel |
| 74 - 73 | opamp2_res_to_out_vdd |
| 75 | opamp2_out_mux_en |
| 77 - 76 | opamp2_inn_sel |
| 80 - 78 | opamp2_inp_sel |
| 81 | opamp3_enable |
| 82 | opamp3_lp_mode |
| 84 - 83 | opamp3_R1_sel |
| 87 - 85 | opamp3_R2_sel |
| 88 | opamp3_en_res_bank |

| BITMAP Location | Signal Name |
|-----------------|-----------------------|
| 91 - 89 | opamp3_res_mux_sel |
| 92 | opamp3_res_to_out_vdd |
| 93 | opamp3_out_mux_en |
| 95 - 94 | opamp3_inn_sel |
| 98 - 96 | opamp3_inp_sel |
| 99 | ldo_bypass |
| 100 | ldo_enable |

DAM Memory Configuration

There is a dedicated ADC DMA to support 16 channels. The DMA is enabled by setting bit **Internal_DMA_Enable** in register **INTERNAL_DMA_CH_ENABLE** to 1. Internal ADC DMA has a small memory where destination address for storing Aux-ADC sample in ULP SRAM is written along with number of samples to be stored. There are 32 locations (2 per channel) in the DMA memory for storing page1(ping) and page2(pong) memory location are stored per channel. Ex: Address 0,1 is used for storing destination memory address to store Aux-ADC sampled for Channel-1 and Address 2,3 is used for storing destination memory address to store Aux-ADC sampled for Channel-2 and so on.

First internal RAM should be written by the user through APB with the information of "start_addr_1, buf_len_1, valid_1, start_addr_2, buf_len_2, valid_2" (mentioned in fig.5) for each channel out of total 16 channels. These "start_addr_1, buf_len_1, valid_1, start_addr_2, buf_len_2, valid_2" information corresponds to the addresses of external memory in which the data from the corresponding channels are stored through AHB Bus.

DMA mode supports dual buffer cyclic mode to avoid loss of data when buffer is full. In dual buffer cyclic mode, if buffer 1 is full for particular channel, incoming sampled data is written into buffer 2 such that, samples from buffer 1 are read back by controller during this time. That's why there are two start addresses, two buffer lengths and two valid signals for each channel.

Note that, buffer-1 should be read while the controller is populating buffer-2 after memory switch, otherwise after buffer-2 getting full, the controller again writes to buffer-1 and as the data of buffer-1 will be overwritten this time, user should make sure that data in buffer-1 was already read.

This information is written into internal RAM through registers "adc_int_mem_1" and "adc_int_mem_2". Two consecutive address of RAM corresponds to one channel information. For example RAM address "0" and "1" holds information regarding "Channel-1", RAM addresses "2" and "3" holds information regarding "Channel-2" and so on. Bit locations `adc_int_mem_2[14:10]` are to be programmed with the RAM address depending on which channel information the user wants to program. For example if user wants to program the information regarding channel-1, first make `adc_int_mem_2[14:10]` to 5'd0 and should program information through remaining bits of "adc_int_mem_2" and "adc_int_mem_1" (as mentioned in section.7) and then make "adc_int_mem_2[14:10]" to 5'd1 and should program information through remaining bits of "adc_int_mem_2" and "adc_int_mem_1" (as mentioned in section.7).

Per channel memory address information from internal RAM can be read back through APB interface valid bit is used to indicate if buffer address is valid or not in case of dual buffer mode.

Note that in multi-channel mode, as the multiple channels are sensed, after sensing each channel, information related to "buffer length" etc. of that particular channel are rewritten into the corresponding location of internal memory for the next use when same channel is sensed again.

Out of two memory chunks allocated for each of the channel to enable "ping-pong" operation in DMA mode, `memory_switch` interrupt will be asserted every time a ping or pong occurs i.e. every time either of the chunk fills up. So, if first memory chunk is filled and if the interrupt is asserted, then the responsibility of clearing that interrupt lies with the processor before second memory chunk fills up so that interrupt corresponding to second memory chunk will be asserted.

This corresponding interrupt of memory switch can be either a pulse or level. Whether this interrupt is a pulse or a level is decided by a bit in ADC control register which specifies "FIMQ" is active or not. If FIMQ is active, then this interrupt will be a pulse and if it is inactive then the interrupt will be level.

Note: Note that the ports to outer memory will be of ULI interface and not AHB bus and the architecture will be as below

Aux-ADC Input Selection

Please refer to Input selection section in Analog to Digital Converter sub-chapter in Analog peripheral chapter for selecting input to AUXADC. If OPAMP's are selected to input to AUXADC, Please refer to Input selection section of OPAMP sub-chapter in Analog peripheral chapter.

17.3.4 Register Summary

Base Address: 0x24043800

| Register Name | Offset | Description |
|--------------------------|--------|-------------|
| AUXDAC_CTRL_1 | 0x00 | |
| AUXADC_CTRL_1 | 0x04 | |
| AUXDAC_CLK_DIV_FAC | 0x08 | |
| AUXADC_CLK_DIV_FAC | 0x0C | |
| AUXDAC_DATA | 0x10 | |
| AUXADC_DATA | 0x14 | |
| AUXADC_DET_THR_CTRL_0 | 0x18 | |
| AUXADC_DET_THR_CTRL_1 | 0x1C | |
| INTR_CLEAR_REG | 0x20 | |
| INTR_MASK_REG | 0x24 | |
| INTR_STATUS_REG | 0x28 | |
| INTR_MASKED_STATUS_REG | 0x2C | |
| FIFO_STATUS_REG | 0x30 | |
| Reserved | 0x34 | |
| ADC_CH1_BIT_MAP_CONFIG_0 | 0x38 | |
| ADC_CH1_BIT_MAP_CONFIG_1 | 0x3C | |
| ADC_CH1_BIT_MAP_CONFIG_2 | 0x40 | |
| ADC_CH1_BIT_MAP_CONFIG_3 | 0x44 | |
| ADC_CH2_BIT_MAP_CONFIG_0 | 0x48 | |
| ADC_CH2_BIT_MAP_CONFIG_1 | 0x4C | |
| ADC_CH2_BIT_MAP_CONFIG_2 | 0x50 | |
| ADC_CH2_BIT_MAP_CONFIG_3 | 0x54 | |
| ADC_CH3_BIT_MAP_CONFIG_0 | 0x58 | |
| ADC_CH3_BIT_MAP_CONFIG_1 | 0x5C | |
| ADC_CH3_BIT_MAP_CONFIG_2 | 0x60 | |
| ADC_CH3_BIT_MAP_CONFIG_3 | 0x64 | |
| ADC_CH4_BIT_MAP_CONFIG_0 | 0x68 | |
| ADC_CH4_BIT_MAP_CONFIG_1 | 0x6C | |
| ADC_CH4_BIT_MAP_CONFIG_2 | 0x70 | |
| ADC_CH4_BIT_MAP_CONFIG_3 | 0x74 | |
| ADC_CH5_BIT_MAP_CONFIG_0 | 0x78 | |
| ADC_CH5_BIT_MAP_CONFIG_1 | 0x7C | |
| ADC_CH5_BIT_MAP_CONFIG_2 | 0x80 | |
| ADC_CH5_BIT_MAP_CONFIG_3 | 0x84 | |
| ADC_CH6_BIT_MAP_CONFIG_0 | 0x88 | |
| ADC_CH6_BIT_MAP_CONFIG_1 | 0x8C | |
| ADC_CH6_BIT_MAP_CONFIG_2 | 0x90 | |

| Register Name | Offset | Description |
|---------------------------|--------|-------------|
| ADC_CH6_BIT_MAP_CONFIG_3 | 0x94 | |
| ADC_CH7_BIT_MAP_CONFIG_0 | 0x98 | |
| ADC_CH7_BIT_MAP_CONFIG_1 | 0x9C | |
| ADC_CH7_BIT_MAP_CONFIG_2 | 0xA0 | |
| ADC_CH7_BIT_MAP_CONFIG_3 | 0xA4 | |
| ADC_CH8_BIT_MAP_CONFIG_0 | 0xA8 | |
| ADC_CH8_BIT_MAP_CONFIG_1 | 0xAC | |
| ADC_CH8_BIT_MAP_CONFIG_2 | 0xB0 | |
| ADC_CH8_BIT_MAP_CONFIG_3 | 0xB4 | |
| ADC_CH9_BIT_MAP_CONFIG_0 | 0xB8 | |
| ADC_CH9_BIT_MAP_CONFIG_1 | 0xBC | |
| ADC_CH9_BIT_MAP_CONFIG_2 | 0xC0 | |
| ADC_CH9_BIT_MAP_CONFIG_3 | 0xC4 | |
| ADC_CH10_BIT_MAP_CONFIG_0 | 0xC8 | |
| ADC_CH10_BIT_MAP_CONFIG_1 | 0xCC | |
| ADC_CH10_BIT_MAP_CONFIG_2 | 0xD0 | |
| ADC_CH10_BIT_MAP_CONFIG_3 | 0xD4 | |
| ADC_CH11_BIT_MAP_CONFIG_0 | 0xD8 | |
| ADC_CH11_BIT_MAP_CONFIG_1 | 0xDC | |
| ADC_CH11_BIT_MAP_CONFIG_2 | 0xE0 | |
| ADC_CH11_BIT_MAP_CONFIG_3 | 0xE4 | |
| ADC_CH12_BIT_MAP_CONFIG_0 | 0xE8 | |
| ADC_CH12_BIT_MAP_CONFIG_1 | 0xEC | |
| ADC_CH12_BIT_MAP_CONFIG_2 | 0xF0 | |
| ADC_CH12_BIT_MAP_CONFIG_3 | 0xF4 | |
| ADC_CH13_BIT_MAP_CONFIG_0 | 0xF8 | |
| ADC_CH13_BIT_MAP_CONFIG_1 | 0xFC | |
| ADC_CH13_BIT_MAP_CONFIG_2 | 0x100 | |
| ADC_CH13_BIT_MAP_CONFIG_3 | 0x104 | |
| ADC_CH14_BIT_MAP_CONFIG_0 | 0x108 | |
| ADC_CH14_BIT_MAP_CONFIG_1 | 0x10C | |
| ADC_CH14_BIT_MAP_CONFIG_2 | 0x110 | |
| ADC_CH14_BIT_MAP_CONFIG_3 | 0x114 | |
| ADC_CH15_BIT_MAP_CONFIG_0 | 0x118 | |
| ADC_CH15_BIT_MAP_CONFIG_1 | 0x11C | |
| ADC_CH15_BIT_MAP_CONFIG_2 | 0x120 | |
| ADC_CH15_BIT_MAP_CONFIG_3 | 0x124 | |
| ADC_CH16_BIT_MAP_CONFIG_0 | 0x128 | |
| ADC_CH16_BIT_MAP_CONFIG_1 | 0x12C | |
| ADC_CH16_BIT_MAP_CONFIG_2 | 0x130 | |
| ADC_CH16_BIT_MAP_CONFIG_3 | 0x134 | |
| ADC_CH1_OFFSET | 0x138 | |
| ADC_CH2_OFFSET | 0x13C | |
| ADC_CH3_OFFSET | 0x140 | |
| ADC_CH4_OFFSET | 0x144 | |
| ADC_CH5_OFFSET | 0x148 | |
| ADC_CH6_OFFSET | 0x14C | |
| ADC_CH7_OFFSET | 0x150 | |

| Register Name | Offset | Description |
|------------------------|-------------|-------------|
| ADC_CH8_OFFSET | 0x154 | |
| ADC_CH9_OFFSET | 0x158 | |
| ADC_CH10_OFFSET | 0x15C | |
| ADC_CH11_OFFSET | 0x160 | |
| ADC_CH12_OFFSET | 0x164 | |
| ADC_CH13_OFFSET | 0x168 | |
| ADC_CH14_OFFSET | 0x16C | |
| ADC_CH15_OFFSET | 0x170 | |
| ADC_CH16_OFFSET | 0x174 | |
| ADC_CH1_FREQ | 0x178 | |
| ADC_CH2_FREQ | 0x17C | |
| ADC_CH3_FREQ | 0x180 | |
| ADC_CH4_FREQ | 0x184 | |
| ADC_CH5_FREQ | 0x188 | |
| ADC_CH6_FREQ | 0x18C | |
| ADC_CH7_FREQ | 0x190 | |
| ADC_CH8_FREQ | 0x194 | |
| ADC_CH9_FREQ | 0x198 | |
| ADC_CH10_FREQ | 0x19C | |
| ADC_CH11_FREQ | 0x1A0 | |
| ADC_CH12_FREQ | 0x1A4 | |
| ADC_CH13_FREQ | 0x1A8 | |
| ADC_CH14_FREQ | 0x1AC | |
| ADC_CH15_FREQ | 0x1B0 | |
| ADC_CH16_FREQ | 0x1B4 | |
| Reserved | 0x1B8-0x1C0 | |
| AUXADC_CONFIG_1 | 0x1C4 | |
| Reserved | 0x1C8 | |
| ADC_SEQ_CTRL | 0x1CC | |
| VAD_BBP_ID | 0x1D0 | |
| ADC_INT_MEM_1 | 0x1D4 | |
| ADC_INT_MEM_2 | 0x1D8 | |
| INTERNAL_DMA_CH_ENABLE | 0x1DC | |
| TS_PTAT_ENABLE | 0x1E0 | |
| Reserved | 0x1E4-0x204 | |
| AUXADC_CONFIG_2 | 0x208 | |
| AUXDAC_CONFIG_1 | 0x20C | |

1136 . Base Address: 0x24043800

17.3.5 Register Description

AUXDAC_CTRL_1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:10 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 9 | R/W | DAC_TO_CTRL_ADC | 0x0 | <p>When set, AUX-DAC control is handed over to Aux ADC-DAC controller. By default, AUX-DAC is under the control of baseband.</p> <p>When this bit is set, AUX-DAC clock and other DAC controls from this controller are routed to the AUX-DAC.</p> <p>In order to ensure that the clock from this controller is properly routed to the DAC, the AUX-DAC clock from BBP should be present prior to the setting of this bit.(BBP clock would be present if BBP is out of soft reset and clocks to BBP are present)</p> <p>In case BBP will never be released from reset in the chip, the setting of this bit is not dependent on the presence of BBP clock.</p> |
| 8:7 | | | | |
| 6 | R/W | DAC_ENABLE_F | 0x0 | <p>This bit is used to enable AUX DAC controller. (Valid only when dac_enable is set)</p> <p>1 – Enable DAC Controller 0 – Disable DAC Controller</p> |
| 5:3 | R/W | DAC_FIFI_THRESHOL D | 0x4 | These bits control the DAC FIFO threshold. When used by DMA, this will act as almost full threshold. For TA, it acts as almost empty threshold. |
| 2 | R/W | DAC_FIFO_FLUSH | 0x0 | <p>This bit is used to flush the DAC FIFO</p> <p>'1' – Flush dac FIFO</p> <p>'0' – Do not flush</p> <p>This bit is self-clearing</p> |
| 1 | R/W | DAC_STATIC_MODE | 0x0 | <p>This bit is used to select non-FIFO mode in DAC.</p> <p>'1' – Static mode is enabled. Data written to the DAC_DATA_REG will not be written to the FIFO. It will be played on DAC directly. Only single sample can be held at a time.</p> <p>'0' – FIFO mode enabled. Data written to the DAC_DATA_REG is written to the FIFO in this mode.</p> <p>In either of these modes, data will be driven to the DAC only when dac_enable is set.</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------|-------------|---|
| 0 | R/W | ENDAC_FIFO_CONFING | 0x0 | <p>This bit activates the DAC path in Aux ADC-DAC controller. Data samples will be played on DAC only when this bit is set.</p> <p>1 – Enable 0 - Disable</p> |

1137 .AUXDAC_CTRL_1 Register Description

AUXADC_CTRL_1

| B i t | A c c e s s | Function | R e s et V a l u e | Description |
|-------------|----------------------------|----------------------|--|--|
| 31:2 8 | | | | |
| 27 | R/W | ADC_NUM_P HASE | 0x0 | |
| 26:1 4 | | | | |
| 13:1 2 | R/W | ADC_CH_SEL _LS | 0x0 | <p>Aux ADC channel number from which the data has to be sampled</p> <p>2'b00 – channel 0 2'b01 – channel 1 2'b10 – channel 2 2'b11 – channel 3</p> <p>This is valid only when adc_multiple_chan_active is ‘0’. When channel number is greater than 3, upper bits should also be programmed ADC_CH_SEL_MS to bits in this register.</p> |
| 11 | | | | |
| 10 | R/W | EN_ADC_CLK | 0x0 | Enable AUX ADC Divider output clock. |
| 9 | R/W | BYPADD NOI SE_AVG | 0x0 | ADC in Bypass noise avg mode |
| 8:7 | R/W | ADC_CH_SEL _MS | 0x0 | Upper 2-bits of adc chan select. When the channel number is greater than 3, these have to be used. The hardware uses {{8:7},{13:12}} as 4-bit channel select |

| B i t | A c e s s | Function | R es et V al u e | Description |
|----------------------|----------------------------------|---------------------------|---|---|
| 6 | R/W | ADC_MULTIPLER_CHAN_ACTIVE | 0x0 | <p>This bit is used to control the auxadcsel signal going to the Aux ADC.</p> <p>'1' – Data will be sampled from four ADC channels in sequential order and written to the receive FIFO in the same order.</p> <p>'0' – Data will be sampled from the programmed ADC channel.(determined by auxadcsel in this register)</p> <p>*Applicable only in FIFO mode.</p> |
| 5:3 | R/W | ADC_FIFO_THRESHOLD | 0x4 | These bits control the ADC FIFO threshold. When used by DMA, this will act as almost empty threshold. For TA, it acts as almost full threshold. |
| 2 | R/W | ADC_FIFO_FLUSH | 0x0 | <p>This bit is used to flush the ADC FIFO</p> <p>'1' – Flush adc FIFO</p> <p>'0' – Do not flush</p> <p>This bit is self clearing</p> |
| 1 | R/W | ADC_STATIC_MODE | 0x0 | <p>This bit is used to select non-FIFO mode in ADC.</p> <p>'1' – Static mode is enabled. ADC data input will be sampled and written to a register in this mode. It will not be written to the FIFO.</p> <p>'0' – FIFO mode enabled. ADC data input will be sampled and written to the ADC FIFO in this mode.</p> <p>In either of these modes, input data from ADC will be sampled only when adc_enable is set. Reading the ADC_DATA_REG provides the value of the sampled data in both the modes.</p> |
| 0 | R/W | ADC_ENABLE | 0x0 | <p>This bits activates the ADC path in Aux ADC-DAC controller. Data will be sampled from ADC only when this bit is set.</p> <p>1 – Enable</p> <p>0 - Disable</p> |

1138 .AUXADC_CTRL_1 Register Description

AUXDAC_CLK_DIV_FAC

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--------------------|
| 31:10 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 9:0 | R/W | AUXDAC_CLK_DIV_FA C | 0x0 | These bits control the DAC clock division factor . clock_freq = input_clock_freq/(2*division factor) Note: '0' value will not generate clock. |

1139 .AUXDAC_CLK_DIV_FAC Register Description

AUXADC_CLK_DIV_FAC

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|---|
| 31:25 | | | | |
| 24:16 | R/W | ADC_CLK_ON_DUR | 0x0 | These bits control the On-Duration of the ADC clock. |
| 15:10 | | | | |
| 9:0 | R/W | ADC_CLK_DIV_FAC | 0x0 | These bits control the Total-Duration of the ADC clock. Note : 'EN_ADC_CLK' bit need to be enable to get clock divider output. |

1140 .AUXADC_CLK_DIV_FAC Register Description

AUXDAC_DATA

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------|-------------|---|
| 31:10 | | | | |
| 9:0 | W | AUXDAC_DATA | 0x0 | Writing to this register will fill DAC FIFO for streaming Data to DAC |

1141 .AUXDAC_DATA Register Description

AUXADC_DATA

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------|-------------|---|
| 31:16 | | | | |
| 15:12 | R | AUXADC_CH_ID | 0 | Channel ID of AUX DATA sample, Valid only in FIFO mode. |
| 11:0 | R | AUXADC_DATA | 0 | AUXADC Data Read through Register. |

1142 .AUXADC_DATA Register Description

ADC_DET_THR_CTRL_0

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:14 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------------------|-------------|---|
| 13:12 | R/W | ADC input detection threshold 0 | 0x0 | Carries upper two bits of ADC detection threshold |
| 11 | R/W | range_comparison_enable | 0x0 | When set, Aux ADC-DAC controller raises an interrupt to processor when the Aux ADC output falls within the range specified in AUX_ADC_DET_THRESHOLD_0 & AUX_ADC_DET_THRESHOLD_1. |
| 10 | R/W | cmp_eq_en | 0x0 | When set, Aux ADC-DAC controller raises an interrupt to processor when the Aux ADC output is equal to the programmed Aux ADC detection threshold. (Bits [7:0] of this register) |
| | | | | When range comparison is set, interrupt will be raised only if the comparison conditions specified AUX_ADC_DET_THRESHOLD_1 are also met. |
| 9 | R/W | cmp_grtr_than_en | 0x0 | When set, Aux ADC-DAC controller raises an interrupt to processor when the Aux ADC output is greater than the programmed Aux ADC detection threshold. (Bits [7:0] of this register) |
| | | | | When range comparison is set, interrupt will be raised only if the comparison conditions specified AUX_ADC_DET_THRESHOLD_1 are also met. |
| 8 | R/W | cmp_less_than_en | 0x0 | When set, Aux ADC-DAC controller raises an interrupt to processor when the Aux ADC output falls below the programmed Aux ADC detection threshold.(Bits [7:0] of this register) |
| | | | | When range comparison is set, interrupt will be raised only if the comparison conditions specified AUX_ADC_DET_THRESHOLD_1 are also met. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------------------|-------------|--|
| 7:0 | R/W | ADC input detection threshold 1 | 0x0 | <p>The value against which the ADC output has to be compared is to be programmed in this register. Bits {13:12,7:0} represent the 10-bit comparison value mode.</p> <p>Interrupt will be raised to the processor as soon as any of the comparison conditions becomes valid.</p> <p>With the existing comparison conditions, the following can be achieved: $>$, $<$, $=$, \geq, \leq</p> <p>When range comparison is enabled, the following can be achieved:</p> <p>$\text{threshold1} < \text{adc output} < \text{threshold2}$</p> <p>$\text{threshold1} \leq \text{adc output} < \text{threshold2}$</p> <p>$\text{threshold1} \leq \text{adc output} \leq \text{threshold2}$</p> <p>Note: This is valid in adc_static_mode only</p> |

1143 .ADC_DET_THR_CTRL_0 Register Description

ADC_DET_THR_CTRL_1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------------------|-------------|---|
| 31:13 | | | | |
| 12:11 | R/W | ADC detection threshold 2 upper bits | 0x0 | Upper 2 bits of ADC detection threshold 2 |
| 10 | R/W | cmp_eq | 0x0 | <p>When set, Aux ADC-DAC controller raises an interrupt to TA when the Aux ADC output is equal to the programmed Aux ADC detection threshold.</p> <p>This is valid only when range comparison is enabled.</p> |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|---------------------------------|-------------|--|
| 9 | R/W | cmp_grtr_than | 0x0 | <p>When set, Aux ADC-DAC controller raises an interrupt to TA when the Aux ADC output is greater than the programmed Aux ADC detection threshold.</p> <p>This is valid only when range comparison is enabled.</p> |
| 8 | R/W | cmp_less_than | 0x0 | <p>When set, Aux ADC-DAC controller raises an interrupt to TA when the Aux ADC output falls below the programmed Aux ADC detection threshold.</p> <p>This is valid only when range comparison is enabled.</p> |
| 7:0 | R/W | ADC input detection threshold 2 | 0x0 | <p>The value against which the ADC output has to be compared is to be programmed in this register. Bits {12:11,7:0} represent the 10-bit comparison value.</p> <p>Interrupt will be raised to the processor as soon as any of the comparison conditions becomes valid.</p> <p>With the existing comparison conditions, the following can be achieved: > , < , = , >= , <=</p> <p>Note: This is valid in adc_static_mode and when range comparison is enabled</p> |

1144 .ADC_DET_THR_CTRL_1 Register Description

INTR_CLEAR_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------|-------------|---|
| 31:24 | | | | |
| 23:8 | R | intr_clr_reg | 0x0 | If enabled, corresponding first_mem_switch_intr bits will be cleared. |
| 7:1 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------|-------------|---|
| 0 | R/WC | Clear_intr | 0x0 | <p>This bit is used to clear threshold detection interrupt</p> <p>'1' – Clear the interrupt</p> <p>'0' – No effect</p> <p>This bit is self-clearing.</p> <p>Upon read, this provides the status of the ADC detection threshold interrupt.</p> |

1145 .INTR_CLEAR_REG Register Description

INTR_MASK_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------------------------|-------------|--|
| 31:23 | | | | |
| 22:7 | R/W | DMA_Channel_intr_mask[0xFFFF 15:0] | | When Cleared, first_mem_switch_intr will be unmasked |
| 6 | R/W | dac_fifo_underrun_intr_ mask | 0x1 | When Cleared, dac FIFO underrun interrupt will be unmasked |
| 5 | R/W | adc_fifo_overflow_intr_m ask | 0x1 | When Cleared, adc FIFO overflow interrupt will be unmasked |
| 4 | R/W | adc_fifo_afull_intr_mask | 0x1 | When Cleared, adc FIFO afull interrupt will be unmasked |
| 3 | R/W | adc_fifo_full_intr_mask | 0x1 | When Cleared, adc FIFO full interrupt will be unmasked |
| 2 | R/W | dac_fifo_aempty_intr_ma sk | 0x1 | When Cleared, dac FIFO aempty interrupt will be unmasked |
| 1 | R/W | dac_fifo_empty_intr_mas k | 0x1 | When Cleared, dac_FIFO_empty interrupt will be unmasked. |
| 0 | R/W | threshold detection intr en | 0x1 | When Cleared, threshold detection interrupt will be unmasked |

1146 .INTR_MASK_REG Register Description

INTR_STATUS_REG

| Bit | Acc ess | Function | Rese t Valu e | Description |
|-------|---------|----------|---------------|-------------|
| 31:23 | | | | |

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------------------------|-------------|---|
| 22:7 | R | DMA_Channel_0x0_intr[15:0] | 0x0 | Interrupt indicating the first memory has been filled and the DMA write is being shifted to second memory chunk for ping-pong operation. Each bit is for each channel. For example, 15 th bit is for 15 th channel and so on. |
| 6 | R | Dac_fifo_underrun | 0x0 | Set when a read is done on DAC FIFO when the FIFO is empty. This happens when the FIFO is empty at the driving edge of the AUX DAC clock. This bit gets cleared as soon as the FIFO is written. |
| 5 | R | Adc_fifo_overflow | 0x0 | Set when a write attempt is made to ADC FIFO when the FIFO is already full. This happens when the FIFO is full at the sampling edge of the AUX ADC clock. This bit gets cleared as soon as the FIFO is read. |
| 4 | R | adc_fifo_afull | 0x0 | Set when ADC FIFO occupancy >= ADC FIFO threshold. This bit gets cleared as soon as the FIFO level falls below the threshold. |
| 3 | R | adc_fifo_full | 0x0 | Set when ADC FIFO is full. This bit gets cleared when data is read from the FIFO. |
| 2 | R | Dac_fifo_aempty | 0x1 | Set when the FIFO occupancy <= DAC FIFO threshold. This bit gets cleared as soon as soon as DAC FIFO occupancy level crosses the programmed threshold. |
| 1 | R | dac_fifo_empty | 0x1 | Set when DAC FIFO is empty. This bit gets cleared when the DAC FIFO at least a single sample is available in DAC FIFO. |
| 0 | R | Adc_threshold_detection_interrupt | 0x0 | This bit is set when ADC threshold matches with the programmed conditions. This will be cleared as soon as this interrupt is acknowledged by processor. |

1147 . INTR_STATUS_REG Register Description

INTR_MASKED_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------------|-------------|---|
| 31:23 | | | | |
| 22:7 | R | DMA_Channel_intr_ma_sked[15:0] | 0x0 | Masked Interrupt status indicating the first memory has been filled and the DMA write is being shifted to second memory chunk for ping-pong operation. Each bit is for each channel. For example, 15 th bit is for 15 th channel and so on. |
| 6 | R | Dac_fifo_underrun_ma_sked | 0x0 | Masked Interrupt. Set when a read is done on DAC FIFO when the FIFO is empty. This happens when the FIFO is empty at the driving edge of the AUX DAC clock. This bit gets cleared as soon as the FIFO is written. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|--------------------------------------|-------------|--|
| 5 | R | Adc_fifo_overflow_masked | 0x0 | Masked Interrupt. Set when a write attempt is made to ADC FIFO when the FIFO is already full. This happens when the FIFO is full at the sampling edge of the AUX ADC clock. This bit gets cleared as soon as the FIFO is read. |
| 4 | R | adc_fifo_afull_masked | 0x0 | Masked Interrupt. Set when ADC FIFO occupancy >= ADC FIFO threshold. This bit gets cleared as soon as the FIFO level falls below the threshold. |
| 3 | R | adc_fifo_full_masked | 0x0 | Masked Interrupt. Set when ADC FIFO is full. This bit gets cleared when data is read from the FIFO. |
| 2 | R | Dac_fifo_aempty_masked | 0x1 | Masked Interrupt. Set when the FIFO occupancy <= DAC FIFO threshold. This bit gets cleared as soon as soon as DAC FIFO occupancy level crosses the programmed threshold. |
| 1 | R | dac_fifo_empty_masked | 0x1 | Masked Interrupt. Set when DAC FIFO is empty. This bit gets cleared when the DAC FIFO at least a single sample is available in DAC FIFO. |
| 0 | R | Adc_threshold_detecti on_intr_masked | 0x0 | Masked Interrupt. This bit is set when ADC threshold matches with the programmed conditions. This will be cleared as soon as this interrupt is acknowledged by processor. |

1148 . INTR_MASKED_STATUS_REG Register Description

FIFO_STATUS_REG

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|--|
| 31:8 | | | | |
| 7 | R | adc_fifo_afull | 0x0 | Set when ADC FIFO occupancy >= ADC FIFO threshold. This bit gets cleared as soon as the FIFO level falls below the threshold. |
| 6 | R | adc_fifo_full | 0x0 | Set when ADC FIFO is full. This bit gets cleared when data is read from the FIFO. |
| 5 | R | Dac_fifo_aempty | 0x1 | Set when the FIFO occupancy <= DAC FIFO threshold. This bit gets cleared as soon as soon as DAC FIFO occupancy level crosses the programmed threshold. |
| 4 | R | dac_fifo_empty | 0x1 | Set when FIFO is empty. This bit gets cleared when the DAC FIFO is not empty. |
| 3 | R | Adc_fifo_aempty | 0x1 | Set when the FIFO occupancy <= ADC FIFO threshold. This bit gets cleared as soon as soon as ADC FIFO occupancy level crosses the programmed threshold. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 2 | R | Adc_fifo_empty | 0x1 | Set when FIFO is empty. This bit gets cleared when the ADC FIFO is not empty. In word mode, FIFO will be shown as non-empty only when occupancy >= 2. |
| 1 | R | Dac_fifo_afull | 0x0 | Set when DAC FIFO occupancy >= FIFO threshold. This bit gets cleared as soon as the FIFO level falls below the threshold. |
| 0 | R | Dac_fifo_full | 0x0 | Set when DAC FIFO is full. In word mode, FIFO will be shown as full unless there is space for 16-bits. This bit gets cleared when data is read from the FIFO and launched to the AUX DAC. |

1149 .FIFO_STATUS_REG Register Description

ADC_CH1_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------------|---------------|--------------------------|----------------------|--------------------------|
| 31:0 | R/W | Channel_1_BitMap [31:0]- | | Refer Bitmap Description |

1150 .ADC_CH1_BIT_MAP_CONFIG_0 Register Description

ADC_CH1_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------------|---------------|---------------------------|----------------------|--------------------------|
| 31:0 | R/W | Channel_1_BitMap [63:32]- | | Refer Bitmap Description |

1151 .ADC_CH1_BIT_MAP_CONFIG_1 Register Description

ADC_CH1_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------------|---------------|---------------------------|----------------------|--------------------------|
| 31:0 | R/W | Channel_1_BitMap [95:64]- | | Refer Bitmap Description |

1152 .ADC_CH1_BIT_MAP_CONFIG_2 Register Description

ADC_CH1_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------------|---------------|---------------------------|----------------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_1_BitMap [98:96]- | | Refer Bitmap Description |

1153 .ADC_CH1_BIT_MAP_CONFIG_3 Register Description

ADC_CH2_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_2_BitMap [31:0]- | | Refer Bitmap Description |

1154 .ADC_CH2_BIT_MAP_CONFIG_0 Register Description

ADC_CH2_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_2_BitMap [63:32]- | | Refer Bitmap Description |

1155 .ADC_CH2_BIT_MAP_CONFIG_1 Register Description

ADC_CH2_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_2_BitMap [95:64]- | | Refer Bitmap Description |

1156 .ADC_CH2_BIT_MAP_CONFIG_2 Register Description

ADC_CH2_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_2_BitMap [98:96]- | | Refer Bitmap Description |

1157 .ADC_CH2_BIT_MAP_CONFIG_3 Register Description

ADC_CH3_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_3_BitMap [31:0]- | | Refer Bitmap Description |

1158 .ADC_CH3_BIT_MAP_CONFIG_0 Register Description

ADC_CH3_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_3_BitMap [63:32]- | | Refer Bitmap Description |

1159 .ADC_CH3_BIT_MAP_CONFIG_1 Register Description

ADC_CH3_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_3_BitMap [95:64]- | | Refer Bitmap Description |

1160 .ADC_CH3_BIT_MAP_CONFIG_2 Register Description

ADC_CH3_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_3_BitMap [98:96]- | | Refer Bitmap Description |

1161 .ADC_CH3_BIT_MAP_CONFIG_3 Register Description

ADC_CH4_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_4_BitMap [31:0]- | | Refer Bitmap Description |

1162 .ADC_CH4_BIT_MAP_CONFIG_0 Register Description

ADC_CH4_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_4_BitMap [63:32]- | | Refer Bitmap Description |

1163 .ADC_CH4_BIT_MAP_CONFIG_1 Register Description

ADC_CH4_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_4_BitMap [95:64]- | | Refer Bitmap Description |

1164 .ADC_CH4_BIT_MAP_CONFIG_2 Register Description

ADC_CH4_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_4_BitMap [98:96]- | | Refer Bitmap Description |

1165 .ADC_CH4_BIT_MAP_CONFIG_3 Register Description

ADC_CH5_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_5_BitMap [31:0]- | | Refer Bitmap Description |

1166 .ADC_CH5_BIT_MAP_CONFIG_0 Register Description

ADC_CH5_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_5_BitMap [63:32]- | | Refer Bitmap Description |

1167 .ADC_CH5_BIT_MAP_CONFIG_1 Register Description

ADC_CH5_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_5_BitMap [95:64]- | | Refer Bitmap Description |

1168 .ADC_CH5_BIT_MAP_CONFIG_2 Register Description

ADC_CH5_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_5_BitMap [98:96]- | | Refer Bitmap Description |

1169 .ADC_CH5_BIT_MAP_CONFIG_3 Register Description

ADC_CH6_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_6_BitMap [31:0]- | | Refer Bitmap Description |

1170 .ADC_CH6_BIT_MAP_CONFIG_0 Register Description

ADC_CH6_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_6_BitMap [63:32]- | | Refer Bitmap Description |

1171 .ADC_CH6_BIT_MAP_CONFIG_1 Register Description

ADC_CH6_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_6_BitMap [95:64]- | | Refer Bitmap Description |

1172 .ADC_CH6_BIT_MAP_CONFIG_2 Register Description

ADC_CH6_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_6_BitMap [98:96]- | | Refer Bitmap Description |

1173 .ADC_CH6_BIT_MAP_CONFIG_3 Register Description

ADC_CH7_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_7_BitMap [31:0]- | | Refer Bitmap Description |

1174 .ADC_CH7_BIT_MAP_CONFIG_0 Register Description

ADC_CH7_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_7_BitMap [63:32]- | | Refer Bitmap Description |

1175 .ADC_CH7_BIT_MAP_CONFIG_1 Register Description

ADC_CH7_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_7_BitMap [95:64]- | | Refer Bitmap Description |

1176 .ADC_CH7_BIT_MAP_CONFIG_2 Register Description

ADC_CH7_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_7_BitMap [98:96]- | | Refer Bitmap Description |

1177 .ADC_CH7_BIT_MAP_CONFIG_3 Register Description

ADC_CH8_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_2_BitMap [31:0]- | | Refer Bitmap Description |

1178 .ADC_CH8_BIT_MAP_CONFIG_0 Register Description

ADC_CH8_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_2_BitMap [63:32]- | | Refer Bitmap Description |

1179 .ADC_CH8_BIT_MAP_CONFIG_1 Register Description

ADC_CH8_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_2_BitMap [95:64]- | | Refer Bitmap Description |

1180 .ADC_CH8_BIT_MAP_CONFIG_2 Register Description

ADC_CH8_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_2_BitMap [98:96]- | | Refer Bitmap Description |

1181 .ADC_CH8_BIT_MAP_CONFIG_3 Register Description

ADC_CH9_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_9_BitMap [31:0]- | | Refer Bitmap Description |

1182 .ADC_CH9_BIT_MAP_CONFIG_0 Register Description

ADC_CH9_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_9_BitMap [63:32]- | | Refer Bitmap Description |

1183 .ADC_CH9_BIT_MAP_CONFIG_1 Register Description

ADC_CH9_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_9_BitMap [95:64]- | | Refer Bitmap Description |

1184 .ADC_CH9_BIT_MAP_CONFIG_2 Register Description

ADC_CH9_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_9_BitMap [98:96]- | | Refer Bitmap Description |

1185 .ADC_CH9_BIT_MAP_CONFIG_3 Register Description

ADC_CH10_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_10_BitMap [31:0]- | | Refer Bitmap Description |

1186 .ADC_CH10_BIT_MAP_CONFIG_0 Register Description

ADC_CH10_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_10_BitMap [63:32]- | | Refer Bitmap Description |

1187 .ADC_CH10_BIT_MAP_CONFIG_1 Register Description

ADC_CH10_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_10_BitMap [95:64]- | | Refer Bitmap Description |

[1188 .ADC_CH10_BIT_MAP_CONFIG_2 Register Description](#)

ADC_CH10_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_10_BitMap [98:96]- | | Refer Bitmap Description |

[1189 .ADC_CH10_BIT_MAP_CONFIG_3 Register Description](#)

ADC_CH11_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_11_BitMap [31:0]- | | Refer Bitmap Description |

[1190 .ADC_CH11_BIT_MAP_CONFIG_0 Register Description](#)

ADC_CH11_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_11_BitMap [63:32]- | | Refer Bitmap Description |

[1191 .ADC_CH11_BIT_MAP_CONFIG_1 Register Description](#)

ADC_CH11_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_11_BitMap [95:64]- | | Refer Bitmap Description |

[1192 .ADC_CH11_BIT_MAP_CONFIG_2 Register Description](#)

ADC_CH11_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_11_BitMap [98:96]- | | Refer Bitmap Description |

[1193 .ADC_CH11_BIT_MAP_CONFIG_3 Register Description](#)

ADC_CH12_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_12_BitMap [31:0]- | | Refer Bitmap Description |

[1194 .ADC_CH12_BIT_MAP_CONFIG_0 Register Description](#)

ADC_CH12_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_12_BitMap [63:32]- | | Refer Bitmap Description |

1195 .ADC_CH12_BIT_MAP_CONFIG_1 Register Description

ADC_CH12_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_12_BitMap [95:64]- | | Refer Bitmap Description |

1196 .ADC_CH12_BIT_MAP_CONFIG_2 Register Description

ADC_CH12_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_12_BitMap [98:96]- | | Refer Bitmap Description |

1197 .ADC_CH12_BIT_MAP_CONFIG_3 Register Description

ADC_CH13_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_13_BitMap [31:0]- | | Refer Bitmap Description |

1198 .ADC_CH13_BIT_MAP_CONFIG_0 Register Description

ADC_CH13_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_13_BitMap [63:32]- | | Refer Bitmap Description |

1199 .ADC_CH13_BIT_MAP_CONFIG_1 Register Description

ADC_CH13_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_13_BitMap [95:64]- | | Refer Bitmap Description |

1200 .ADC_CH13_BIT_MAP_CONFIG_2 Register Description

ADC_CH13_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_13_BitMap [98:96]- | | Refer Bitmap Description |

1201 .ADC_CH13_BIT_MAP_CONFIG_3 Register Description

ADC_CH14_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_14_BitMap [31:0]- | | Refer Bitmap Description |

1202 .ADC_CH14_BIT_MAP_CONFIG_0 Register Description

ADC_CH14_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_14_BitMap [63:32]- | | Refer Bitmap Description |

1203 .ADC_CH14_BIT_MAP_CONFIG_1 Register Description

ADC_CH14_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_14_BitMap [95:64]- | | Refer Bitmap Description |

1204 .ADC_CH14_BIT_MAP_CONFIG_2 Register Description

ADC_CH14_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_14_BitMap [98:96]- | | Refer Bitmap Description |

1205 .ADC_CH14_BIT_MAP_CONFIG_3 Register Description

ADC_CH15_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_15_BitMap [31:0]- | | Refer Bitmap Description |

1206 .ADC_CH15_BIT_MAP_CONFIG_0 Register Description

ADC_CH15_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_15_BitMap [63:32]- | | Refer Bitmap Description |

1207 .ADC_CH15_BIT_MAP_CONFIG_1 Register Description

ADC_CH15_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_15_BitMap [95:64]- | | Refer Bitmap Description |

1208 .ADC_CH15_BIT_MAP_CONFIG_2 Register Description

ADC_CH15_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_15_BitMap [98:96]- | | Refer Bitmap Description |

1209 .ADC_CH15_BIT_MAP_CONFIG_3 Register Description

ADC_CH16_BIT_MAP_CONFIG_0

| Bit | Access | Function | Default Value | Description |
|------|--------|---------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_16_BitMap [31:0]- | | Refer Bitmap Description |

1210 .ADC_CH16_BIT_MAP_CONFIG_0 Register Description

ADC_CH16_BIT_MAP_CONFIG_1

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_16_BitMap [63:32]- | | Refer Bitmap Description |

1211 .ADC_CH16_BIT_MAP_CONFIG_1 Register Description

ADC_CH16_BIT_MAP_CONFIG_2

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:0 | R/W | Channel_16_BitMap [95:64]- | | Refer Bitmap Description |

1212 .ADC_CH16_BIT_MAP_CONFIG_2 Register Description

ADC_CH16_BIT_MAP_CONFIG_3

| Bit | Access | Function | Default Value | Description |
|------|--------|----------------------------|---------------|--------------------------|
| 31:3 | | | | |
| 2:0 | R/W | Channel_16_BitMap [98:96]- | | Refer Bitmap Description |

1213 .ADC_CH16_BIT_MAP_CONFIG_3 Register Description

ADC_CH1_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch1_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-1 should be sampled. |

1214 .ADC_CH1_OFFSET Register Description

ADC_CH2_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch2_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-2 should be sampled. |

1215 .ADC_CH2_OFFSET Register Description

ADC_CH3_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch3_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-3 should be sampled. |

1216 .ADC_CH3_OFFSET Register Description

ADC_CH4_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch4_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-4 should be sampled. |

1217 .ADC_CH4_OFFSET Register Description

ADC_CH5_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch5_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-5 should be sampled. |

[1218 .ADC_CH5_OFFSET Register Description](#)

ADC_CH6_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch6_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-6 should be sampled. |

[1219 .ADC_CH6_OFFSET Register Description](#)

ADC_CH7_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch7_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-7 should be sampled. |

[1220 .ADC_CH7_OFFSET Register Description](#)

ADC_CH8_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch8_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-8 should be sampled. |

[1221 .ADC_CH8_OFFSET Register Description](#)

ADC_CH9_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch9_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-9 should be sampled. |

[1222 .ADC_CH9_OFFSET Register Description](#)

ADC_CH10_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------|---------------|-------------|
| 31:16 | | | | |

| Bit | Access | Function | Default Value | Description |
|------|--------|-------------|---------------|--|
| 15:0 | R/W | ch10_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-10 should be sampled. |

1223 .ADC_CH10_OFFSET Register Description

ADC_CH11_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch11_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-11 should be sampled. |

1224 .ADC_CH11_OFFSET Register Description

ADC_CH12_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch12_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-12 should be sampled. |

1225 .ADC_CH12_OFFSET Register Description

ADC_CH13_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch13_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-13 should be sampled. |

1226 .ADC_CH13_OFFSET Register Description

ADC_CH14_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch14_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-14 should be sampled. |

1227 .ADC_CH14_OFFSET Register Description

ADC_CH15_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------|---------------|-------------|
| 31:16 | | | | |

| Bit | Access | Function | Default Value | Description |
|------|--------|-------------|---------------|--|
| 15:0 | R/W | ch15_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-15 should be sampled. |

1228 .ADC_CH15_OFFSET Register Description

ADC_CH16_OFFSET

| Bit | Access | Function | Default Value | Description |
|-------|--------|-------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch16_offset | 0 | This Register specifies initial offset value with respect to AUX_ADC clock after which Channel-16 should be sampled. |

1229 .ADC_CH16_OFFSET Register Description

ADC_CH1_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch1_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-1. |

1230 .ADC_CH1_FREQ Register Description

ADC_CH2_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch2_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-2. |

1231 .ADC_CH2_FREQ Register Description

ADC_CH3_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------|---------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch3_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-3. |

1232 .ADC_CH3_FREQ Register Description

ADC_CH4_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------|---------------|-------------|
| 31:16 | | | | |

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|---|
| 15:0 | R/W | ch4_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-4. |

1233 .ADC_CH4_FREQ Register Description

ADC_CH5_FREQ

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch5_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-5. |

1234 .ADC_CH5_FREQ Register Description

ADC_CH6_FREQ

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch6_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-6. |

1235 .ADC_CH6_FREQ Register Description

ADC_CH7_FREQ

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch7_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-7. |

1236 .ADC_CH7_FREQ Register Description

ADC_CH8_FREQ

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|---|
| 31:16 | | | | |
| 15:0 | R/W | ch8_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-8. |

1237 .ADC_CH8_FREQ Register Description

ADC_CH9_FREQ

| Bit | Access | Function | Default Value | Description |
|------------|---------------|-----------------|----------------------|--------------------|
| 31:16 | | | | |

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------|---------------|---|
| 15:0 | R/W | ch9_freq_val | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-9. |

1238 .ADC_CH9_FREQ Register Description

ADC_CH10_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|---------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch10_freq_val | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-10. |

1239 .ADC_CH10_FREQ Register Description

ADC_CH11_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch11_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-11. |

1240 .ADC_CH11_FREQ Register Description

ADC_CH12_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch12_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-12. |

1241 .ADC_CH12_FREQ Register Description

ADC_CH13_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch13_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-13. |

1242 .ADC_CH13_FREQ Register Description

ADC_CH14_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch14_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-14. |

1243 .ADC_CH14_FREQ Register Description

ADC_CH15_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch15_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-15. |

1244 .ADC_CH15_FREQ Register Description

ADC_CH16_FREQ

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|--|
| 31:16 | | | | |
| 15:0 | R/W | ch16_freq_value | 0 | This register specifies Sampling frequency rate at which AUX ADC Date is sampled for Channel-16. |

1245 .ADC_CH16_FREQ Register Description

AUXADC_CONFIG_1

| Bit | Access | Function | Default value | Description |
|-------|--------|-------------------|---------------|--------------------------------------|
| 31:27 | | | | |
| 26 | R/W | auxadc_diff_mode0 | | AUX ADC Differential Mode |
| 25:22 | R/W | auxadc_inn_sel | 0 | Mux select for negetive input of adc |
| 21:17 | R/W | auxadc_inp_sel | 0 | Mux select for positive input of adc |
| 16:0 | | | | |

1246 .AUXADC_CONFIG_1 Register Description

ADC_SEQ_CTRL

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------------------|---------------|---|
| 31:16 | R/W | adc_seq_ctrl_dma_mode | 0x0 | To enable/disable per channel DAM mode (One-hot coding)Bit 31 :Enable for Channel 16Bit 30 :Enable for Channel 15Bit 29 :Enable for Channel 14Bit 28 :Enable for Channel 13Bit 27 :Enable for Channel 12Bit 26 :Enable for Channel 11Bit 25 :Enable for Channel 10Bit 24 :Enable for Channel 9Bit 23 :Enable for Channel 8Bit 22 :Enable for Channel 7Bit 21 :Enable for Channel 6Bit 20 :Enable for Channel 5Bit 19 :Enable for Channel 4Bit 18 :Enable for Channel 3Bit 17 :Enable for Channel 2Bit 16 :Enable for Channel 1 |
| 15:0 | R/W | adc_seq_ctrl_ping_pong | 0x0 | To enable/disable per channel ping-pong operation (One-hot coding)Bit 15 :Enable for Channel 16Bit 14 :Enable for Channel 15Bit 13 :Enable for Channel 14Bit 12 :Enable for Channel 13Bit 11 :Enable for Channel 12Bit 10 :Enable for Channel 11Bit 9 :Enable for Channel 10Bit 8 :Enable for Channel 9Bit 7 :Enable for Channel 8Bit 6 :Enable for Channel 7Bit 5 :Enable for Channel 6Bit 4 :Enable for Channel 5Bit 3 :Enable for Channel 4Bit 2 :Enable for Channel 3Bit 1 :Enable for Channel 2Bit 0 :Enable for Channel 1 |

1247 .ADC_SEQ_CTRL Register Description

VAD_BBP_ID

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------|---------------|---|
| 31:16 | R/W | discont_mode | 0x0 | Per channel discontinuous mode enable signal. When discontinuous mode is enabled, data is sampled only once from that channel and the enable bit is reset to 0. |
| 15:6 | | | | |
| 5 | R/W | aux_adc_bbp_en | 0x0 | Enable Indication for BBP |
| 4 | R/W | bbp_en | 0x0 | Enables Aux-ADC samples to BBP |
| 3:0 | R/W | bbp_id | 0x0 | Channel id for bbp samples |

1248 .VAD_BBP_ID Register Description

ADC_INT_MEM_1

| Bit | Access | Function | Default Value | Description |
|------|--------|--------------------|---------------|---|
| 31:0 | R/W | prog_wr_data[31:0] | 0x0 | These 32-bits specifies the start address of first/second buffer corresponding to the channel location ADC_INT_MEM_2[14:10] |

1249 .ADC_INT_MEM_1 Register Description

ADC_INT_MEM_2

| Bit | Access | Function | Default Value | Description |
|-------|--------|---------------------|---------------|---|
| 31:16 | | | | |
| 15 | R/W | prog_wr_data[42] | 0x0 | Valid bit for first/second buffers corresponding to ADC_INT_MEM_2[14:10] |
| 14:10 | R/W | prog_wr_addr | 0x0 | These bits correspond to the address of the internal memory basing on the channel number, whose information we want to program. For example this will be “0” or “1” for channel number 1 and so on. |
| 9:0 | R/W | prog_wr_data[41:32] | 0x0 | These 10-bits specify the buffer length of first/second buffer corresponding to the channel location ADC_INT_MEM_2[14:10] |

1250 .ADC_INT_MEM_2 Register Description

INTERNAL_DMA_CH_ENABLE

| Bit | Access | Function | Default Value | Description |
|--------|--------|---------------------|---------------|---|
| 31 | R/W | Internal_DMA_Enable | 0x0 | When Set, Internal DMA will be used for reading ADC samples from ADC FIFO and writing them to ULP SRAM Memories. |
| 30:16 | | | | |
| [15:0] | R/W | Per_channel Enable | 0x0 | Enable bit for Each channel.Bit 15 :Enable for Channel 16Bit 14 :Enable for Channel 15Bit 13 :Enable for Channel 14Bit 12 :Enable for Channel 13Bit 11 :Enable for Channel 12Bit 10 :Enable for Channel 11Bit 9 :Enable for Channel 10Bit 8 :Enable for Channel 9Bit 7 :Enable for Channel 8Bit 6 :Enable for Channel 7Bit 5 :Enable for Channel 6Bit 4 :Enable for Channel 5Bit 3 :Enable for Channel 4Bit 2 :Enable for Channel 3Bit 1 :Enable for Channel 2Bit 0 :Enable for Channel 1 |

1251 .INTERNAL_DMA_CH_ENABLE Register Description

TS_PTAT_ENABLE

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|--|
| 31:16 | | | | |
| 0 | R/W | TS_PTAT_EN | 0x0 | BJT based Temperature sensor enable1 : Enable0 : Disable |

1252 .TS_PTAT_ENABLE Register Description

AUXADC_CONFIG_2

| Bit | Access | Function | Default value | Description |
|-------|--------|----------------------|---------------|------------------------------|
| 31:12 | | | | |
| 11 | R/W | AUXADC_CONFIG_ENABLE | 0 | Aux ADC Configuration Enable |

| Bit | Access | Function | Default value | Description |
|------|--------|----------|---------------|-------------|
| 10:0 | | | | |

1253 .AUXADC_CONFIG_2 Register Description

AUXDAC_CONIG_1

| Bit | Access | Function | Default value | Description |
|-------|--------|---------------------|---------------|--|
| 31:15 | | | | |
| 14 | R/W | AUXDAC_DYN_EN | 0 | Enable for sending DAC words through FIFO |
| 13:4 | R/W | AUXDAC_DATA_S | 0 | Satatic AUX Dac Data |
| 3 | | | | |
| 2 | R/W | AUXDAC_OUT_MUX_SEL0 | | 0 – DAC Output is connected to AGPIO4 1 – DAC Output is connected to AGPIO15 Note: PAD should be configured to Analog mode |
| 1 | R/W | AUXDAC_OUT_MUX_EN | 0 | 0 – DAC output is not connected to PAD 1 – DAC output is connected to PAD |
| | R/W | AUXDAC_EN_S | 0 | Enable signal DAC |

1254 .AUXDAC_CONFIG_1 Register Description

17.4 Capacitive Touch Controller

17.4.1 General Description

The capacitive touch sensor (CTS) controller is used to detect the position of the touch from the user on the capacitive touch screen. This is achieved by sensing the change of the capacitor value of the sensor through an analog comparator, digital counter and the supporting circuitry. The touch is detected by the final count of the digital counter which will be proportional to the capacitor value.

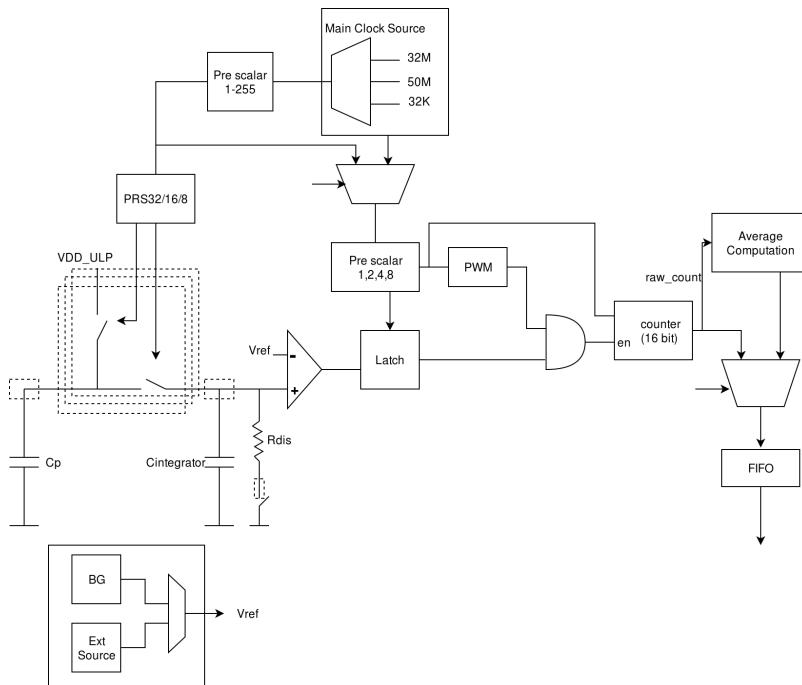
17.4.2 Features

- Supports up to 8 capacitive touch sensors
- Programmable input clock source from the multiple available clocks in the chip. Generate the divided clock with programmable division factor.
- Has an option to use external reference clock instead of internal programmable reference.
- Controls the rate of scanning for all sensors with configurable inter sensor scan ON time
- Support both samples streaming and cumulative average mode
- Uses PWM for generating a measurement window for counting the raw sample counts
- Has 8 to 16 bit PWM resolution
- Has asynchronous FIFO size 64x16, with an interrupt raised after FIFO occupancy crosses the configurable threshold value
- DMA capable
- 8,16 and 32-bit pseudo-random number for generating two non overlapping streams with configurable delay
- Programmable polynomial and seed values for pseudo-random number generator
- Programmable Vref to get proper sensing

- Provides Wake-up indication after capacitive touch sensing

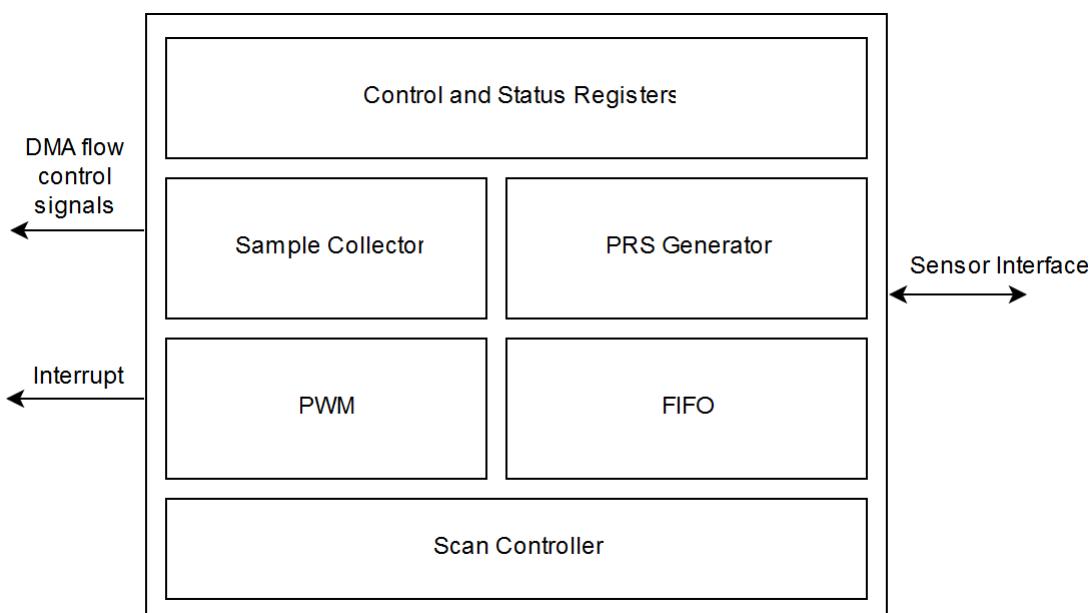
17.4.3 Functional Description

System level block diagram of Capacitive touch sensor is shown in below figure1.



Capacitive touch sensor system level block diagram

The capacitive touch sensor should be able to detect changes in the parasitic capacitance (C_p) and represent it in the form of changes in raw counts. The increase in parasitic capacitance is an indication of the presence of a human body near the touch panel connected to the PCB. The increased capacitance should be detected as an increase in the raw counts. If the count value increases by more than a certain % from threshold value, it will be considered as touch. The above block diagram represents such a system. Capacitive touch controller block diagram is shown in below figure2.



Capacitive touch controller block diagram

17.4.4 Programming Sequence

Comparator reference voltage selection

1) Program comparator reference voltage by using ref_volt_config bits in CTS_CONFIG_REG_1_7 Register

This value is used to select input reference voltage(Vref) to analog voltage comparator. This needs to be programmed correctly and depending on frequency of non-Overlap Φ_1 and Φ_2 streams, capacitor value Cp and the resistor value Rb to get proper latch output. The capacitance sensor detect the input capacitance and represent it in the form of count.

$$\text{Count} = \text{Rdis} * \text{fsw1} * \text{Cp} * (\text{VBATT} - \text{Vref}) / \text{Vref} \quad (\text{vref has to be trimmed to get nominal count of 0.5 - 0.8})$$

Rdis - discharging resistor

fsw1 - frequency of fsw1 clock coming from capsense controller

Cp - Input/sensor capacitance

Example: Rdis=30K Ω , fsw1=1MHz, Cp=10pF

| VBATT | Vref | Count |
|-------|------|-------|
| 3.6 | 1 | 0.78 |
| 3 | 0.9 | 0.7 |
| 2.4 | 0.7 | 0.73 |
| 1.8 | 0.5 | 0.78 |

Its real value will be count * PWM_on_period.

Resistor Selection:

| Operating Freq | Rdis |
|----------------|---------------|
| 10MHz | 3K Ω |
| 1MHz | 30K Ω |
| 100KHz | 300K Ω |

The above resistor values are optimized to make capsense work for voltage range of 1.8-3.6V, when input cap Cp=10pF

Vref Selection:

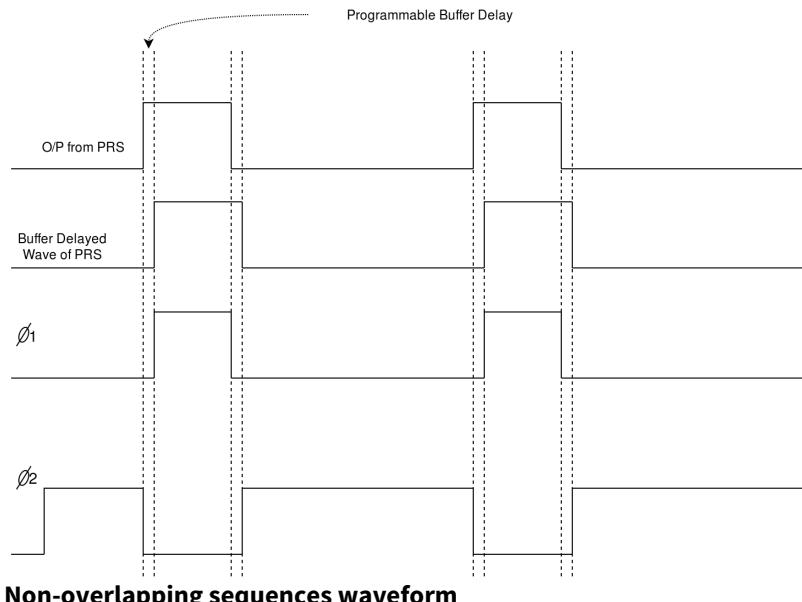
| vref_sel | vreg | units |
|----------|------|-------|
| 0 | 0.5 | V |
| 1 | 0.6 | V |
| 2 | 0.7 | V |
| 3 | 0.8 | V |
| 4 | 0.9 | V |
| 5 | 1.0 | V |
| 6 | 1.1 | V |
| 7 | 1.2 | V |

2) For knowing voltage level on ULP_IO_VDD33 :

ULP_VDD_33 supply should be available for the capsense to work. Keep a RC circuit on board (9116 module/SVP board) and connect it to one of the Aux-ADC channel and measure ULP_IO_VDD33 voltage. Above values to be programmed according to this voltage. This calibration to be done periodically by capacitive touch sensor firmware.

Non-Overlap Φ_1 and Φ_2 stream generation

Pseudo random sequence (PRS) generator comprises of a Linear Feedback shift register (LFSR), whose taps can be programmed according to the polynomial register bits set by the user. Required clock to PRS and Non overlap sequence generator is selected by using pre_scalar_1 and clk_sel1 in CTS_CONFIG_REG_0_0. PRS Polynomial and seed values are programmed by using CTS_CONFIG_REG_1_3 and CTS_CONFIG_REG_1_3 registers. Select the polynomial length and load the seed using polynomial_len and seed_load bits in CTS_CONFIG_REG_1_1 register. On each clock cycle, we get an output bit from the right most shift register and it is given to the “Non-overlap stream generator” which outputs two sequences Φ_1 and Φ_2 respectively. In Non Over lap sequence (NOS) generator, which consist of programmable 32 level buffer delay generator, each buffer consume around 0.2 ns time. These delays are programmed by using bits 7:3 of CTS_CONFIG_REG_1_1 register to generate non-overlap Φ_1 and Φ_2 stream.



Non-overlapping sequences waveform

Operating modes

There are two modes of operation.

One-hot mode

In this mode, the sensor will be sampling data from only one sensor and for a fixed number of samples. After all the data samples have been taken, the sensor will be inactive till the next trigger.

1. Select One hot mode by programming cnt_onehot_mode in CTS_CONFIG_REG_1_1 register
2. Select only one sensor among the 8 sensors using CTS_CONFIG_REG_1_6 and CTS_CONFIG_REG_1_7 registers.
3. Program the number of data samples to be taken by using CTS_CONFIG_REG_1_5 register.
4. For multiple samples, there is an option of taking all the samples or just their average by programming 12 bit in CTS_CONFIG_REG_1_1 register.
5. The sampled data can be written to a FIFO by programming 13 bit in CTS_CONFIG_REG_1_1 register.
6. The sensor can be triggered using external NPSS clock(1 ms/1 s clock) or internal clock by programming 19 and 18 bit in CTS_CONFIG_REG_1_1 register.

Continuous mode

In this mode, the sensor will be sampling data from multiple sensors. Each sensor will take in a fixed number of samples. This cycle will be running continuously till the trigger is active. Once the next trigger comes, the cycle will restart from its initial state.

1. Select continuous mode by programming cnt_onehot_mode in CTS_CONFIG_REG_1_1 register.
2. Select multiple sensors among the 8 sensors using CTS_CONFIG_REG_1_6 and CTS_CONFIG_REG_1_7 registers.
3. Program the number of data samples to be taken by using CTS_CONFIG_REG_1_5 register.
4. For multiple samples, there is an option of taking all the samples or just their average by programming 12 bit in CTS_CONFIG_REG_1_1 register.
5. The sampled data can be written to a FIFO by programming 13 bit in CTS_CONFIG_REG_1_1 register.
6. The sensor can be triggered using external NPSS clock(1 ms/1 s clock) or internal clock by programming 19 and 18 bit in CTS_CONFIG_REG_1_1 register.

Sensing and Capturing of samples from touch sensor

1. Enable CAPACITIVE_TOUCH power domain using NPSS register ULPSS_PWRCTRL_SET_REG
2. Enable CAPACITIVE_TOUCH clock using ULP_MISC_SOFT_SET_REG and ULP_TOUCH_CLK_GEN_REG registers present in ULPSS misc config register.
3. CAPACITIVE_TOUCH requires PCLK to read/write registers. Hence, PCLK must be enabled using ULP_MISC_SOFT_SET_REG as well as ULP_TA_CLK_GEN_REG registers. This is enabled by default.
4. Configuration of CTS internal clocks
 - a. Disable the CTS clock enable if already enabled i.e., clearing BIT(0) in ULP_TOUCH_CLK_GEN_REG.
 - b. Program [4:1] bits in ULP_TOUCH_CLK_GEN_REG register for selecting input source clock from RC, RO, SOC or xtal clocks.
 - c. Program clock division factor using [12:5]bits in ULP_TOUCH_CLK_GEN_REG register
 - d. Enable clock by programming BIT(0) in ULP_TOUCH_CLK_GEN_REG register.
5. Initialize ULP uDMA to read data from CTS FIFO
6. Enable NVIC interrupt for CAP Sensor
7. Data need to be sent on AGPIO depending upon the sensor which need to be touched. Program the required GPIOs used. For example
 - a. AGPIO8 for TOUCH0 sensor
 - b. AGPIO9 for TOUCH1
 - c. AGPIO10 for TOUCH2
 - d. AGPIO7 for TOUCH3
 - e. AGPIO6 for TOUCH4
 - f. AGPIO3 for TOUCH5
 - g. AGPIO0 for TOUCH6
 - h. AGPIO11 for TOUCH7
8. Program all required configuration in CAP_SENSOR registers depending the operating mode
 - a. To select proper clock, program clock muxsel1, clk muxsel2, pre_scalar_1 and pre_scalar_2 values using **CONFIG_REG_0_0** register.
 - b. The sensor can be triggered using external NPSS clock(1 ms/1 s clock) or internal clock by programming 19 and 18 bits in CTS_CONFIG_REG_1_1 register.
 - c. Configure fifo_aempty_thrl and fifo_afull_thrl in **CONFIG_REG_0_0** register
 - d. Configure one hot/continuous mode using BIT(11) in **CONFIG_REG_1_1** to sense the eight sensors.
 - e. Program BIT(16) and sample_mode in **CONFIG_REG_1_1** register to store samples in the FIFO. In case of multiples samples, there is an option of taking all the samples or just their average.
 - f. Configure buffer_delay in **CONFIG_REG_1_1**. Max value that can be programmed is 31
 - g. Configure polynomial_len in **CONFIG_REG_1_1** register
 - h. Configure Polynomial for PRS generator in **CONFIG_REG_1_4** register
 - i. Configure PRS seed value in **CONFIG_REG_1_3** register
 - j. Set Seed load in **CONFIG_REG_1_1** register

- k. To Bypass random number generator set BIT(15) in **CONFIG_REG_1_1** register
 - l. depending on requirement, PWM ON time and OFF time can be programmed in **CONFIG_REG_1_2** register
 - m. Configure inter sensor delay and number of repetitions of a sensor to be scanned by using bits [31:16] in **CONFIG_REG_1_5** register. This inter sensor delay will cause a sensor to wait for some time before it starts taking samples.
 - n. Configure sensor_cfg **CONFIG_REG_1_6** register;
 - o. Configure wake_up_threshold by using bits [31 : 16] in **CONFIG_REG_1_7** register for which the sensor detects it as a touch
 - p. If average is enabled, then wakeup mode BIT(15) need to be set in **CONFIG_REG_1_7** register. If this bit is set, then the system wake up for each value greater than the threshold value configured. i.e. comes out of sleep.
 - q. Configure reference voltage [14 : 8] depending on requirement and set BIT(6) for Vref in **CONFIG_REG_1_7** register.
 - r. disable the fifo_afull interrupt BIT(7) in **CONFIG_REG_1_7** register, if the wake up is enabled
 - s. Configure number of sensors [3 : 0] that are valid in **CONFIG_REG_1_7** register. For one-hot mode, value 1 is only valid.
 - t. Enable cap sensor by programming BIT(9) in **CONFIG_REG_1** register. This will start the scan controller.
9. Once cap sensor voltage greater than wake up threshold is sensed, the system wakes-up and read data from the FIFO using udma.
10. Enable the fifo_afull interrupt using BIT(7) in **CONFIG_REG_1_7** register after wakeup.

17.4.5 Register Summary

Base Address: 0x2404_2C00

| Register Name | Offset | Description |
|--------------------|--------|----------------------------|
| CTS_CONFIG_REG_0_0 | 0x000 | Configuration Register 0_0 |
| CTS_CONFIG_REG_1_1 | 0x100 | Configuration Register 1_1 |
| CTS_CONFIG_REG_1_2 | 0x104 | Configuration Register 1_2 |
| CTS_CONFIG_REG_1_3 | 0x108 | Configuration Register 1_3 |
| CTS_CONFIG_REG_1_4 | 0x10C | Configuration Register 1_4 |
| CTS_CONFIG_REG_1_5 | 0x110 | Configuration Register 1_5 |
| CTS_CONFIG_REG_1_6 | 0x114 | Configuration Register 1_6 |
| CTS_CONFIG_REG_1_7 | 0x118 | Configuration Register 1_7 |
| CTS_CONFIG_REG_1_8 | 0x11C | Configuration Register 1_8 |
| CTS_CONFIG_REG_1_9 | 0x120 | Configuration Register 1_9 |
| CTS_FIFO_ADDRESS | 0x004 | FIFO Address Register |

1255 . Register Summary Table

17.4.6 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, - = Reserved

CTS_CONFIG_REG_0_0 Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------|-----------|--|
| 31:29 | - | Reserved | | |
| 28 | R | fifo_empty | 0x0 | FIFO empty status bit |
| 27:22 | R/W | fifo_aempty_thrld | 0x00 | Threshold for fifo aempty |
| 21:16 | R/W | fifo_afull_thrld | 0x00 | Threshold for fifo afull |
| 15 | R/W | cts_static_clk_en | 0x0 | 1 - Clocks are not gated 0 - Clocks are gated |
| 14 | R/W | clk_sel2 | 0x0 | Mux select for clock_mux_2 |
| 13:10 | R/W | pre_scalar_2 | 0x0 | Division factor for clock divider |
| 9:2 | R/W | pre_scalar_1 | 0x0 | Division factor for clock divider |
| 1:0 | R/W | clk_sel1 | 0x0 | Mux select for clock_mux_1 |

1256 . CONFIG_REG_0_0 Register Description

CTS_CONFIG_REG_1_1 Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|---------------|-----------|--|
| 31:20 | - | Reserved | | |
| 19 | R/W | ext_trig_en | 0x0 | Select bit for NPSS clock / Enable 1 : NPSS clock 0 : Enable |
| 18 | R | Reserved | 0x0 | Reserved |
| 17:16 | R/W | bit_sel | 0x0 | Selects different set of 12 bits to be stored in FIFO |
| 15 | R/W | bypass | 0x0 | Bypass signal 1 – Bypass the Random number generator output to the Non-overlapping stream generator and to give “clk” as input to the Non-Overlapping stream generator. 0 – Use Random number generator output bit as input to Non-Overlapping stream generator. |
| 14 | R/W | reset_wr_fifo | 0x0 | In read operation, it acts as FIFO aempty status bit. In write operation, it resets the signal fifo_wr_int 0 - Reset 1 - Out of reset |

| Bit | Access | Function | POR Value | Description |
|-------|--------|-----------------|-----------|---|
| 13:12 | R/W | sample_mode | 0x0 | Select bits for FIFO write and FIFO average sample_mode[1] : 1 - No FIFO Write 0 - FIFO Write sample_mode[0] : 1 - Average 0 - No average |
| 11 | R/W | cnt_onehot_mode | 0x0 | Continuous or One hot mode 1 – Continuous 0 – One hot |
| 10 | R/W | soft_reset_2 | 0x0 | Reset the FIFO write and FIFO read occupancy pointers 1 - Reset 0 - Out of reset |
| 9 | R/W | enable | 0x0 | Enable signal 1 – enable the cap sensor module 0 – disable the cap sensor module |
| 8 | R/W | wake_up_ack | 0x0 | Ack for wake up interrupt. This is a level signal. To acknowledge wake up , set this bit to one and reset it . |
| 7:3 | R/W | buffer_delay | 0x0A | Delay of buffer. Delay programmed will be equal to delay in nano seconds. Max delay value is 32. Default delay should be programmed before using Capacitive touch sensor module. |
| 2 | R/W | seed_load | 0x0 | Seed of polynomial 1 – to load the seed 0 – loading of seed is not allowed |
| 1:0 | R/W | polynomial_len | 0x0 | Length of polynomial |

1257 .CONFIG_REG_1_1 Register Description

CTS_CONFIG_REG_1_2 Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------------|-----------|----------------|
| 31:16 | R/W | pwm_off_period | 0x0000 | PWM OFF period |
| 15:0 | R/W | pwm_on_period | 0x0000 | PWM ON period |

1258 .CONFIG_REG_1_2 Register Description

CTS_CONFIG_REG_1_3 Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | R/W | prs_seed | 0x0000_0000 | Pseudo random generator (PRS) seed value |

1259 .CONFIG_REG_1_3 Register Description

CTS_CONFIG_REG_1_4 Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-------------|---|
| 31:0 | R/W | prs_poly | 0x0000_0000 | Polynomial programming register for PRS generator |

1260 .CONFIG_REG_1_4 Register Description

CTS_CONFIG_REG_1_5 Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|--------------------|-----------|--------------------------------------|
| 31:16 | R/W | N_sample_count | 0x0000 | Number of repetitions of sensor scan |
| 15:0 | R/W | inter_sensor_delay | 0x00FF | Inter-sensor scan delay value |

1261 .CONFIG_REG_1_5 Register Description

CTS_CONFIG_REG_1_6 Register

| Bit | Access | Function | POR Value | Description |
|------|--------|------------|-------------|---|
| 31:0 | R/W | sensor_cfg | 0x0000_0000 | Register of scan controller containing the programmed bit map |

1262 .CONFIG_REG_1_6 Register Description

CTS_CONFIG_REG_1_7 Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|-------------------|-----------|---|
| 31:16 | R/W | wake_up_threshold | 0x0000 | Wakeup threshold |
| 15 | R/W | wakeup_mode | 0x0 | Select bit for high/low mode 1 : Wakeup if count is greater than threshold 0 : Wakeup if count is lesser than threshold Wakeup mode will only work for average mode. |
| 14:12 | R/W | ref_volt_config | 0x000 | This is given as an input voltage to analog model as comparator reference voltage. |
| 11:8 | - | Reserved | | |

| Bit | Access | Function | POR Value | Description |
|-----|--------|----------------------|-----------|---|
| 7 | R/W | mask_fifo_afull_intr | 0x1 | <p>Wake up interrupt and fifo_afull_intr are ORed and given as a single interrupt to the processor.</p> <p>To enable wake up interrupt alone, fifo_afull_intr has to be masked.</p> <p>1 - fifo_afull_intr is masked</p> <p>0 - fifo_afull_intr is unmasked</p> |
| 6 | R/W | vref_sel | 0x0 | Enable for Vref programmed |
| 5:4 | - | Reserved | | |
| 3:0 | R/W | valid_sensors | 0x0 | Value of number of sensors valid in the bit map |

1263 .CONFIG_REG_1_7 Register Description

CTS_CONFIG_REG_1_8 Register

| Bit | Access | Function | POR Value | Description |
|------|--------|-----------|-------------|----------------------|
| 31:0 | R | prs_state | 0x0000_0000 | Current state of PRS |

1264 .CONFIG_REG_1_8 Register Description

CTS_CONFIG_REG_1_9 Register

| Bit | Access | Function | POR Value | Description |
|-------|--------|----------|-----------|---|
| 31:10 | - | Reserved | | |
| 9:0 | R/W | trig_div | 0x000 | Allows one pulse for every 'trig_div' no. of pulses of 1 ms clock |

1265 .CONFIG_REG_1_9 Register Description

CTS_FIFO_ADDRESS Register

| Bit | Access | Function | POR Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | R/W | FIFO | 0x0000_0000 | Used for FIFO reads & write operations |

1266 .CTS_FIFO_ADDRESS Register Description

17.5 VAD

17.5.1 General Description

VAD (Voice-Activity-Detection) is an hardware accelerator to detect voice activity on the samples provided by the Processor. The samples can be collected through ADC from a Analog Source or I²S from a Digital Source. There are multiple algorithms used for voice activity detection which are configurable by the Processor.

17.5.2 Features

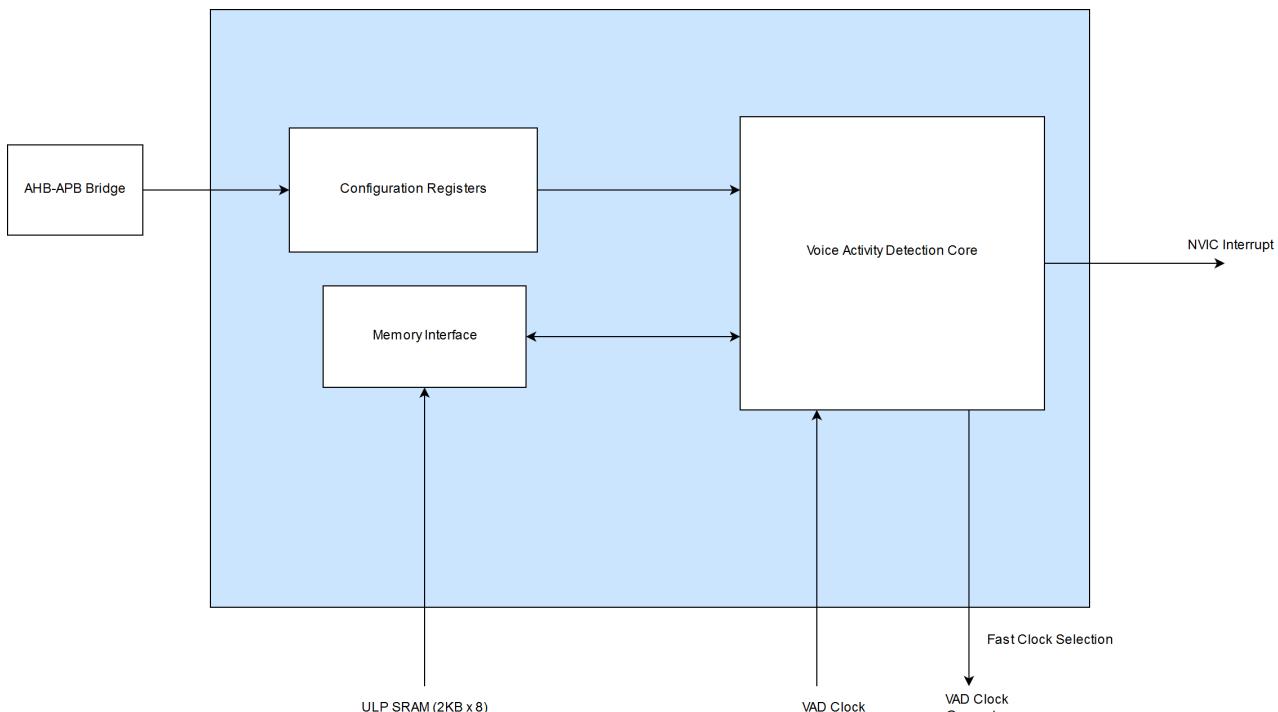
- Support Interrupt generation on Voice Activity Detection.
- Support Multiple Algorithms based detection

- Zero Crossing (ZCR).
- Auto Correlation Function (ACF).
- Weighted Auto Correlation Function (WACF).
- Average Magnitude Difference Function (AMDF).
- Support flexible storage of samples in Memory.
- Support multiple SRAM banks using ping-pong method for processing samples.

17.5.3 Functional Description

Block Diagram

The diagram below depicts the VAD Interface and functionality.



Block Diagram of VAD

17.5.4 Programming Sequence

The programming sequence below needs to be followed for performing the Voice Activity Detection. Refer the Register description below for the configuration registers.

1. Configure VAD Slow clock and Fast clock as described in MCU-ULP Clock architecture section.
2. Ensure that the samples on which VAD needs to be performed are present in SRAM before initiating the process.
 - a. The 10-bit Input samples need to be sign-extended to 16-bits/32-bits based on the sample bit width.
3. Configure the Ping and Pong Addresses to be used by VAD in VAD_CONFIG_REG9 Register.
4. Configure the detection parameters for different algorithms based on sample depth being used for VAD.
5. Configure the sample ordering in SRAM which is configured in VAD_CONFIG_REG1.
6. Configure the sample source and the combination of Algorithms need to be used for VAD which is configured in VAD_CONFIG_REG7.
7. Program the Samples present in SRAM to VAD as described in VAD_CONFIG_REG8.
8. Start the VAD operation as described in VAD_CONFIG_REG8.

9. Poll for VAD processing Status as described in VAD_CONFIG_REG8.

17.5.5 Register Summary

Base Address: 0x2404_3400

| Register Name | Offset | Description |
|---------------|--------|---------------------------------|
| VAD_CONF_REG1 | 0x00 | Configuration Registers for VAD |
| VAD_CONF_REG2 | 0x04 | |
| VAD_CONF_REG3 | 0x08 | |
| VAD_CONF_REG4 | 0x0C | |
| VAD_CONF_REG5 | 0x10 | |
| VAD_CONF_REG6 | 0x14 | |
| VAD_CONF_REG7 | 0x18 | |
| VAD_CONF_REG8 | 0x1C | |
| VAD_CONF_REG9 | 0x20 | |

1267 . List of VAD Registers

17.5.6 Register Description

VAD_CONF_REG1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|--|
| 31:25 | - | Reserved | 0 | It is recommended to write these bits to 0. |
| 24 | RW | full_width | 0 | Writing 1 to this indicates that the 32-bits are used to store 2 10-bit samples. Writing 1 to this indicates that the 32-bits are used to store 1 10-bit samples. |
| 23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | RW | normalize_frame | 0 | Normalize frame |
| 21:20 | RW | smpls_per_addr | 1 | Indicates the bit width of the samples 0 - Reserved 1 - 16-bit Samples 2 - 32-bit Samples |
| 19:10 | RW | threshold_mag | 420 | Indicates the magnitude threshold for the AMDF algorithm. |
| 9:0 | RW | smpls_per_frame | 512 | Indicates the number of samples in one processing frame. |

1268 . VAD_CONF_REG1

VAD_CONF_REG2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|---|
| 31:10 | - | Reserved | - | It is recommended to write these bits to 0. |
| 9:0 | RW | smples_zero_cross | 10 | Indicates the threshold for number of zero crossings to ensure detection using ZCR algorithm. |

1269 .VAD_CONF_REG2

VAD_CONF_REG3

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------------------|-------------|---|
| 31:22 | - | Reserved | - | It is recommended to write these bits to 0. |
| 21:20 | RW | prog_smples_for_energ1y_check | 1 | Indicates the number of samples for validating the energy of the samples before further processing. 0 - 4 Samples 1 - 8 Samples 2 - 16 Samples 3 - 32 Samples |
| 19:10 | RW | threshold_smpl_collected | 1 | Indicates the threshold for energy of the samples (configured in prog_smples_for_energy_check which is described above). |
| 9:0 | RW | threshold_frame_energy | 0 | Indicates the threshold for energy of the samples over the entire processing frame to start the algorithm execution. |

1270 .VAD_CONF_REG3

VAD_CONF_REG4

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|---|
| 31:24 | - | Reserved | - | It is recommended to write these bits to 0. |
| 23:12 | RW | threshold_wacf | 51 | Threshold for WACF algorithm. |
| 11:0 | RW | threshold_acf | 1024 | Threshold for ACF algorithm. |

1271 .VAD_CONF_REG4

VAD_CONF_REG5

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------|-------------|---|
| 31:23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | RW | amdf_by_sqaring | 0 | Writing 1 to this enables sqaring function for AMDF. Writing 0 to this disables sqaring function for AMDF. |
| 21:12 | RW | threshold_null_coun t | 21 | Indicates the threshold for number of Null's for AMDF algorithm to detect Voice activity. |
| 11:0 | RW | threshold_null | 2662 | Indicates the threshold for determining a Null for AMDF algorithm to detect Voice activity. |

1272 .VAD_CONF_REG5

VAD_CONF_REG6

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------|-------------|---|
| 31:22 | - | Reserved | - | It is recommended to write these bits to 0. |
| 21:12 | RW | threshold_peak_cou nt | 410 | Indicates the threshold for number of Peak's for AMDF algorithm to detect Voice activity. |
| 11:0 | RW | threshold_peak | 1843 | Indicates the threshold for determining a Peak for AMDF algorithm to detect Voice activity. |

1273 .VAD_CONF_REG6

VAD_CONF_REG7

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-------------------|-------------|--|
| 31:23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22:20 | RW | choose_vad_method | 4 | Indicates the combination of Algorithms to be used for Voice Activity Detection. 0x0 - ZCR 0x1 - ACF 0x2 - AMDF 0x3 - WACF 0x4 - ZCR and ACF and AMDF and WACF 0x5 - ZCR and ACF 0x6 - Reserved 0x7 - ZCR and WACF |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|--|
| 19:10 | RW | end_delay_val | 16 | End delay used in ACF, WACF and AMDF algorithms. |
| 9:0 | RW | start_delay_val | 2 | Start delay used in ACF, WACF and AMDF algorithms. |

1274 .VAD_CONF_REG7

VAD_CONF_REG8

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|---|
| 31:23 | - | Reserved | - | It is recommended to write these bits to 0. |
| 22 | R | vad_proc_done | 0 | Indicates that the VAD processing is done. |
| 21:11 | - | Reserved | - | It is recommended to write these bits to 0. |
| 10 | RW | en_vad_process | 0 | Writing 1 to this enables VAD processing. Writing 1 to this disabled VAD processing. |
| 9:0 | RW | INP_DATA | 0 | 10 bit Samples Data used for VAD. |

1275 .VAD_CONF_REG8

VAD_CONF_REG9

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|--|
| 31:29 | - | Reserved | - | It is recommended to write these bits to 0. |
| 28 | W | pong_int_clear | 0 | Writing 1 to this clears the VAD interrupt on Pong Buffer samples. |
| 27 | W | ping_int_clear | 0 | Writing 1 to this clears the VAD interrupt on Ping Buffer samples. |
| 26:14 | RW | pong_addr | 512 | Indicates the Start Address of Pong Memory containing the VAD samples. Value = (ULP_SRAM_ADDR >> 2). Please Refer Memory Architecture Section for ULP-SRAM Addressing. |
| 13:1 | RW | ping_addr | 0 | Indicates the Start Address of Ping Memory containing the VAD samples. Value = (ULP_SRAM_ADDR >> 2). Please Refer Memory Architecture Section for ULP-SRAM Addressing. |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

1276 .VAD_CONF_REG9

17.6 FIM

17.6.1 General Description

FIM (Filter-Interpolation-Multiplication) is used in a wide range of applications for performing complex operations. This block contains hardware accelerators with support for Filtering, Interpolation, Multiplication and Add/Subtract operations.

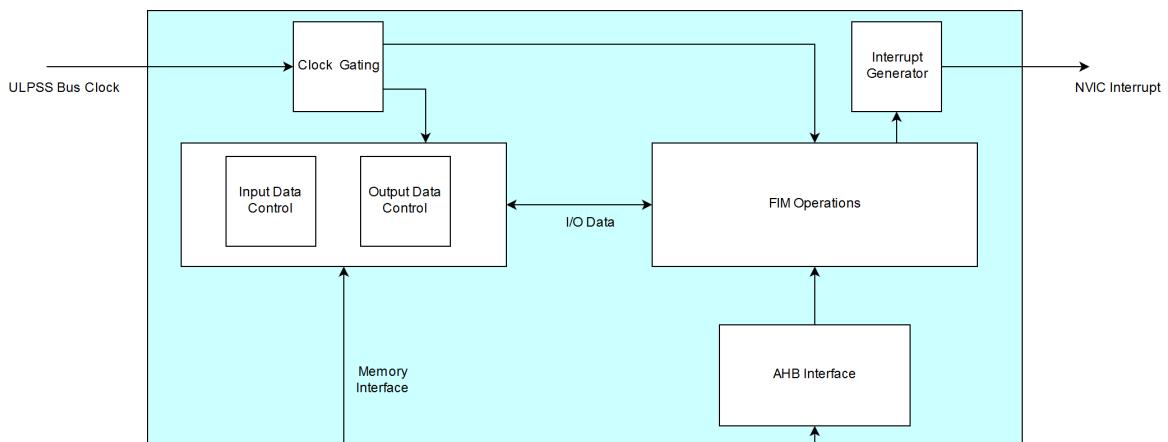
17.6.2 Features

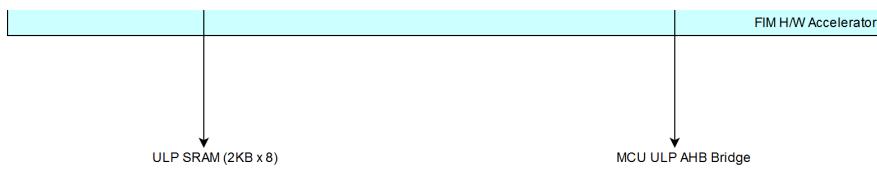
- The following accelerators are supported
 - FIR Filtering.
 - IIR Filtering.
 - Interpolation.
 - Scalar Addition.
 - Scalar Subtraction.
 - Scalar Multiplication.
 - Vector Addition.
 - Vector Subtraction.
 - Vector Multiplication.
 - NORM Square computation.
 - Matrix Multiplication.
- Supports maximum input length of 1023 for Addition, Subtraction and Multiplication.
- Supports maximum output length of 1023 for Filtering and Interpolation.
- Supports the above operations for Real and Complex Inputs.
- Supports fixed point operations through programmable shifting.
- Supports inputs with 32-bit precision for Real operations and 16-bit precision for Complex operations.
- Supports the saturation on the Output Data.
- Interface with ULP-SRAM (16KB) for Inputs and Outputs.
- Interrupt Generation for Processor wakeup from Standby states.
- Operates on MCU ULP Bus Clock.
- The clocks are disabled on reset and needs to be enabled for performing an operation.
- S-Bus Interface through MCU ULP AHB Bridge.

17.6.3 Functional Description

Block Diagram

The diagram below depicts the FIM Interface and functionality.





Block Diagram of FIM

17.6.4 Operations

Addition/Subtraction

This operation can be performed on Real or Complex values. For Real Operations 32-bits are considered as 1 element whereas for complex operations MSB 16-bits of the 32-bits are considered as Real and LSB 16-bits are considered as Imaginary components.

The Mathematical equation carried out for this is:

- For Scalar Mode: $\text{OPER_OUT} = (\text{INP1} + \text{SCALAR})$ where INP1 is a vector and the SCALAR value is added/subtracted from all the elements in the INP1 vector.
- For Vector Mode: $\text{OPER_OUT} = (\text{INP1} + \text{INP2})$ where INP1 and INP2 are vector and the addition/subtraction is done on corresponding elements.

Due to this operation the number of bits in the output data can be increased by 1-bit and the Saturation Value decided the number of LSB's to be removed. If the Saturation value is programmed to "0", then the output data is saturated to the maximum value supported by 32-bit/16-bit based on the real/complex type of operation performed.

So the final output is interpreted as $\text{OUT} = \text{OPER_OUT} >> \text{SAT_VAL}$. The Maximum value for the SAT_VAL parameter is 0x1F for Real operations and 0xF for Complex operations.

Multiplication

This operation can be performed on Real or Complex values. For Real Operations 32-bits are considered as 1 element whereas for complex operations MSB 16-bits of the 32-bits are considered as Real and LSB 16-bits are considered as Imaginary components.

The Mathematical equation carried out for this is:

- For Scalar Mode: $\text{OPER_OUT} = (\text{INP1} * \text{SCALAR})$ where INP1 is a vector and the SCALAR value is multiplied with all the elements in the INP1 vector.
- For Vector Mode: $\text{OPER_OUT} = (\text{INP1} * \text{INP2})$ where INP1 and INP2 are vector and the multiplication is done on corresponding elements.

Based on the type of data mentioned the operation is done in the following manner:

- For INP1-Real and INP2-Real: $\text{OPER_OUT} = \text{INP1} * \text{INP2}$
- For INP1-Real and INP2-Complex: $\text{OPER_OUT} = [\text{INP1} * \text{Real}(\text{INP2})] + j[\text{INP1} * \text{Imag}(\text{INP2})]$
- For INP1-Complex and INP2-Real: $\text{OPER_OUT} = [\text{Real}(\text{INP1}) * \text{INP2}] + j[\text{Imag}(\text{INP1}) * \text{INP2}]$
- For INP1-Complex and INP2-Complex: $\text{OPER_OUT} = [(\text{Real}(\text{INP1}) * \text{Real}(\text{INP2})) - (\text{Imag}(\text{INP1}) * \text{Imag}(\text{INP2}))] + j[(\text{Real}(\text{INP1}) * \text{Imag}(\text{INP2})) + (\text{Imag}(\text{INP1}) * \text{Real}(\text{INP2}))]$

Due to this operation the number of bits in the output data can be increased by twice the input bits and the Saturation Value decided the number of LSB's to be removed. If the Saturation value is programmed to "0", then the output data is saturated to the maximum value supported by 32-bit/16-bit based on the real/complex type of operation performed.

So the final output is interpreted as $\text{OUT} = \text{OPER_OUT} >> \text{SAT_VAL}$. The Maximum value for the SAT_VAL parameter is 0x1F for Real operations and 0xF for Complex operations.

Matrix-Multiplication

This operation can be performed only on Real values. The number of rows for Matrix1 and number of columns for Matrix2 are defined. The number of columns for Matrix1 and number of rows for Matrix2 need to be same.

The derivation of the final output is same as described in the Multiplication section above.

NORM Square

This operation can be performed only on Complex values and only INP1 is used for this operation.

The Mathematical equation carried out for this is: $\text{OPER_OUT} = [(\text{Real}(\text{INP1}) * \text{Real}(\text{INP1})) + ((\text{Imag}(\text{INP1}) * \text{Imag}(\text{INP1})))]$

The derivation of the final output is same as described in the Multiplication section above.

FIR Filtering

This operation can be performed on both Real and Complex values. The Inputs should be filled in such a way that the vector (out of Input Data and Filter coefficients) with the higher number of elements should be present in INP2 Memory locations.

The Mathematical equation carried out for this is: $\text{OPER_OUT}(n) = \sum \text{INP}(n) * \text{COEFF}(n-k)$ k is the filter length.

The length of the output data is derived as $\text{Output length} = \text{input_length} + \text{filter_length} - 1;$

For example: if a,b,c,d are inputs and x,y,z are filter coefficients, then the outputs are

- Out1 = x * a
- Out2 = (y * a) + (x * b)
- Out3 = (z * a) + (y * b) + (x * c)
- Out4 = (z * b) + (y * c) + (x * d)
- Out5 = (z * c) + (y * d)
- Out6 = (z * d)

So the final output is interpreted as $\text{OUT} = \text{OPER_OUT} >> \text{SAT_VAL}$. The Maximum value for the SAT_VAL parameter is 0x1F for Real operations and 0xF for Complex operations.

IIR Filtering

This operation can be performed on both Real and Complex values. Maximum of 2 feed-back filter coefficients are supported. The Inputs should be filled in such a way that the vector (out of Input Data and Filter coefficients) with the higher number of elements should be present in INP2 Memory locations.

INP1 is the data to be filtered and INP2 is the feed-forward filter coefficients. The feed-back filter coefficients are programmed through SCALAR_POLE_DATA1 and POLE_DATA2.

The length of the output data is derived as $\text{Output length} = \text{input_length} + \text{feed_forward_length} - 1;$

For example: if a0, a1, a2, a3 are inputs and x0, x1, x2, x3 are feed-forward coefficients and b0, b1 are feed-back coefficients, then the outputs are

- Out1 = x0 * a0
- Out2 = ((x1 * a0) + (x0 * a1) + (out0 * b1))
- Out3 = ((x2 * a0) + (x1 * a1) + (x0 * a2) + (out1 * b1) + out0 * b2)
- Out4 = ((x3 * a0) + (x2 * a1) + (x1 * a2) + (x0 * a3)) (out2 * b1) + out1 * b2)
- Out5 = ((x3 * a1) + (x2 * a2) + (x1 * a3)) (out3 * b1) + out2 * b2)
- Out6 = ((x3 * a2) + (x2 * a3)) (out4 * b1) + out3 * b2)
- Out7 = ((x3 * a3)) (out5 * b1) + out4 * b2)

So the final output is interpreted as ***OUT = OPER_OUT >> SAT_VAL***. The Maximum value for the SAT_VAL parameter is 0x1F for Real operations and 0xF for Complex operations.

Interpolation

Interpolation Factor is configured through INTRP_FAC parameter. The Inputs should be filled in such a way that the vector (out of Input Data and Filter coefficients) with the higher number of elements should be present in INP2 Memory locations.

If Interpolation Factor is L then (L-1) zeros are appended between consecutive input samples and then the FIR filtering operation is performed .If L=1,then it is an FIR implementation

17.6.5 Programming Sequence

The programming sequence below needs to be followed for performing any of the FIM operation. Refer the Register description below for the configuration registers.

1. Enable FIM clock as described in ULPSS Clock architecture section.
2. Fill the SRAM with the input values required.
3. Configure the INPUT and OUTPUT addresses as described in FIM_INP1_ADDR, FIM_INP2_ADDR and FIM_OUT_ADDR Register.
4. Configure the type of operation to be done as described in FIM_MODE_INTERRUPT Register.
5. Enable operational mode Latching as described in FIM_MODE_INTERRUPT Register.
6. Program the Scalar Data for Scalar operation or Feedback coefficients for IIR filtering as described in FIM_SCALAR_POLE_DATA1 and FIM_POLE_DATA2.
7. Configure the parameters for Output Data configuration as described in FIM_SAT_SHIFT Register.
8. Configure the parameters as described in FIM_CONFIG_REG1 and FIM_CONFIG_REG2.
9. Enable FIM interrupt in NVIC.
10. Start the FIM Operation as described FIM_CONFIG_REG2.
11. Wait for Interrupt indicating the completion of FIM operation.
12. Clear the FIM interrupt as described in FIM_MODE_INTERRUPT Register.
13. The Output Data is present in the SRAM at the location programmed in FIM_OUT_ADDR Register.

17.6.6 Register Summary

Base Address: 0x2407_0000

| Register Name | Offset | Description |
|-----------------------|--------|---|
| FIM_MODE_INTERRUPT | 0x00 | Configuration for FIM Operation Mode and Interrupt Control. |
| FIM_INP1_ADDR | 0x04 | Memory Offset Address for 1 st Input for FIM Operations. |
| FIM_INP2_ADDR | 0x08 | Memory Offset Address for 2 nd Input for FIM Operations. |
| FIM_OUT_ADDR | 0x0C | Memory Offset Address for Output from FIM Operations. |
| FIM_SCALAR_POLE_DATA1 | 0x10 | Scalar Input Data or Feedback coefficient for IIR filter operation. |
| FIM_POLE_DATA2 | 0x14 | Feedback coefficient for IIR filter operation. |
| FIM_SAT_SHIFT | 0x18 | Configuration for precision of Output Data for FIM Operations. |
| FIM_CONFIG_REG1 | 0x1C | Configuration Register for FIM Operations. |
| FIM_CONFIG_REG2 | 0x20 | Configuration Register for FIM Operations. |

[1277 . List of FIM Registers](#)

17.6.7 Register Description

FIM_MODE_INTERRUPT

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------|-------------|---|
| 31:11 | - | Reserved | - | It is recommended to write these bits to 0. |
| 10 | W | INTR_CLEAR | 0 | Writing 1 to this clears the FIM Interrupt to Processor. Writing 0 to this has no effect. |
| 9 | - | Reserved | - | It is recommended to write these bits to 0. |
| 8:1 | RW | OPER_MODE | 0 | Indicates the Mode of Operation to be performed 0x01 - FIR Filtering 0x02 - IIR Filtering 0x44 - Scalar Addition 0x45 - Scalar Subtraction 0x46 - Scalar Multiplication 0x47 - Vector Addition 0x49 - Vector Subtraction 0x4A - Vector Multiplication 0x4C - Matrix Multiplication 0x63 - Interpolation 0xAB - Norm Square |
| 0 | W | LATCH_MODE | 0 | Writing 1 to this latches the FIM Operation as configured above. Writing 0 to this has no effect. |

1278 .FIM Mode/Interrupt Register

FIM_INP1_ADDR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------|-------------|--|
| 31:13 | - | Reserved | - | It is recommended to write these bits to 0. |
| 12:0 | RW | INP1_ADDR | 0 | Indicates the Start Address of 1 st Input Data for FIM Operations Value = (ULP_SRAM_ADDR >> 2). Please Refer Memory Architecture Section for ULP-SRAM Addressing. |

1279 .FIM Input1 Address Register

FIM_INP2_ADDR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------|-------------|--|
| 31:13 | - | Reserved | - | It is recommended to write these bits to 0. |
| 12:0 | RW | INP2_ADDR | 0 | Indicates the Start Address of 2 nd Input Data for FIM Operations Value = (ULP_SRAM_ADDR >> 2). Please Refer Memory Architecture Section for ULP-SRAM Addressing. |

1280 .FIM Input-2 Address Register

FIM_OUT_ADDR

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|---|
| 31:13 | - | Reserved | - | It is recommended to write these bits to 0. |
| 12:0 | RW | OUT_ADDR | 0 | Indicates the Start Address of Output Data for FIM Operations Value = (ULP_SRAM_ADDR >> 2). Please Refer Memory Architecture Section for ULP-SRAM Addressing. |

1281 .FIM Output Address Register

FIM_SCALAR_POLE_DATA1

| Bit | Access | Function | Reset Value | Description |
|------|--------|-------------------|-------------|--|
| 31:0 | RW | SCALAR_POLE_DATA1 | 0 | Indicates the Input Scalar Data for Scalar Operations. Indicates the feedback coefficient for IIR Operations. |

1282 .FIM Scalar Data/Feed back Register1

FIM_POLE_DATA2

| Bit | Access | Function | Reset Value | Description |
|------|--------|------------|-------------|--|
| 31:0 | RW | POLE_DATA2 | 0 | Indicates the feedback coefficient for IIR Operations. |

1283 .FIM Feed back Register2

FIM_SAT_SHIFT

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|---|
| 31:16 | - | Reserved | - | It is recommended to write these bits to 0. |

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 15:10 | RW | SHIFT_VAL | 0 | Indicates the number of bits to be right-shifted for Output Data. |
| 9:5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4:0 | RW | SAT_VAL | 0 | Indicates the number of MSB's to be saturated for Output Data. |

1284 .FIM SAT TRUNC ROUND Register

FIM_CONFIG_REG1

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| 31:26 | - | Reserved | - | It is recommended to write these bits to 0. |
| 25:16 | RW | INP2_LEN | 0 | Indicates the length of 2 nd input for FIM Operations other than filtering (FIR, IIR) and Interpolation. Indicates the length of the Input Data or Filter coefficients whichever is the highest. |
| 15:6 | RW | INP1_LEN | 0 | Indicates the length of 1 st input for FIM Operations other than filtering (FIR, IIR) and Interpolation. Indicates the length of the Input Data or Filter coefficients whichever is the highest. |
| 5:0 | RW | MAT_LEN | 0 | Indicates the number of columns in 1 st input for Matrix Multiplication. This is same as number of rows in 2 nd input for Matrix Multiplication. Indicates the length of the Input Data or Filter coefficients whichever is the lowest. |

1285 .FIM Configuration Register1

FIM_CONFIG_REG2

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|---|
| 31:26 | - | Reserved | - | It is recommended to write these bits to 0. |
| 27:22 | RW | INTRP_FAC | 0 | Indicates the Interpolation Factor. |
| 21:16 | RW | ROW_M1 | 0 | Indicates the number of rows in 1 st input for Matrix Multiplication. |
| 15:10 | RW | COL_M2 | 0 | Indicates the number of columns in 2 nd input for Matrix Multiplication. |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------|-------------|---|
| 9:8 | RW | CPLX_FLAG | 0 | <p>Indicates the Type of Inputs provided for FIM Operations</p> <p>0x0: Input1 is Real and Input2 is Real</p> <p>0x1: Input1 is Real and Input2 is Complex</p> <p>0x2: Input1 is Complex and Input2 is Real</p> <p>0x3: Input1 is Complex and Input2 is Complex</p> <p>Note: This is not valid for Matrix Multiplication</p> |
| 7:1 | - | Reserved | - | It is recommended to write these bits to 0. |
| 0 | W | START_OPER | 0 | <p>Writing 1 to this starts the FIM Operation as configured above.</p> <p>Writing 0 to this has no effect.</p> |

1286 .FIM Configuration Register2

17.7 Enhanced-GPIO (EGPIO)

17.7.1 General Description

The functionality on the ULP GPIOs is similar to the SoC GPIOs. In addition to this, the register details are also similar to MCU HP Enhanced GPIO Peripheral except for the difference in the Register Base Address.

The peripheral functionality is described in [Enhanced GPIO \(EGPIO\)](#) section in MCU APB Peripherals Section.

17.8 ULP Timers

17.8.1 General Description

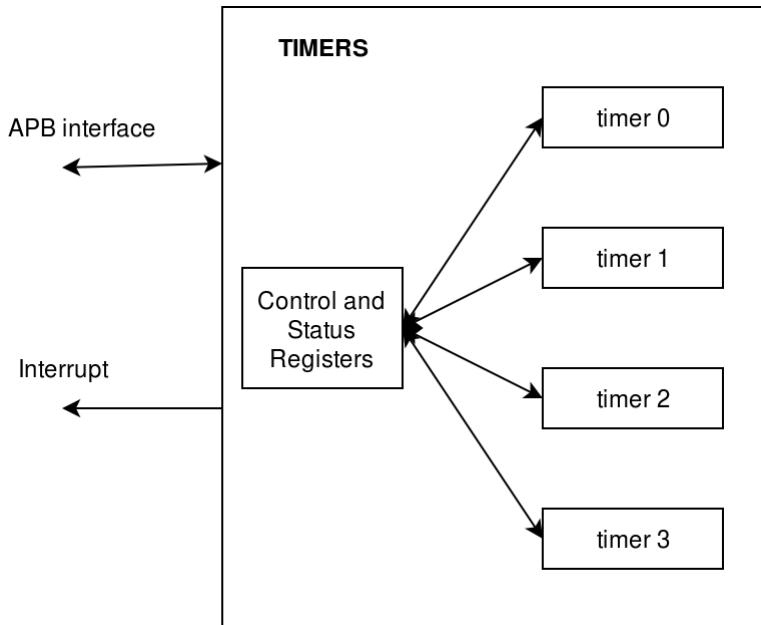
The MCU ULP Timer block supports four 32-bit timers, which can be used to generate various timing events for the software. Each of the four timers can be independently programmed to work in periodic or one-shot mode and can be configured either as a microsecond timer or as a counter.

17.8.2 Features

- Supports 4 independent timers
- Supports per timer enable and disable.
- Option to configure each timer as a 32-bit counter or 32-bit microsecond timer.
- Supports 1µs mode and 256µs modes per timer.
- Support for programming 1µs and 256µs time unit values. Accounts for integral and fractional value of the time units programmed.
- Microsecond timer supports two modes:
 - 1 Microsecond mode : The time unit is 1µs. Number of microseconds required to be counted has to be programmed.
 - 256microsecond mode : The time unit is 256µs. Number of 256µs units required to be counted has to be programmed. This is useful when the timer is being used for counting large time values and microsecond based tracking not required.

- Supports one shot and periodic modes per timer.
- Option to interrupt the processor on timeout. Supports per timer interrupt enable and disable.
- Can run synchronous or asynchronous to SoC clock.

17.8.3 Block Diagram



Timers block Diagram

17.8.4 Functional Description

Basic Timer Operation

To operate the n^{th} timer the count till which the counter should count is loaded in the MCUULP_TMRn_MATCH register and the start bit MCUULP_TMRn_CNTRL[TMR_START] is set to start the timer. When the timer reaches the timeout value, a time out indication can be read in the MCUULP_TMR_INTR_STAT register. If the interrupt is enabled in the MCUULP_TMRn_CNTRL register the timeout condition will generate an interrupt to the processor. The timer can also be stopped any time in between its operation before the time out condition by setting the MCUULP_TMRn_CNTRL[TMR_STOP] bit. After stopping the timer new parameters can be programmed into the registers.

One-shot & Periodic Operation

In one shot operation, the timer counts till the timeout and then generates a single interrupt after which it returns to idle state. Whereas in periodic operation the timer when reaches the timeout value generates an interrupt and starts counting again from the originally set value.

Timers are by default in one shot mode; Periodic operation can be enabled by setting the MCUULP_TMRn_CNTRL[TMR_TYPE] bit.

Microsecond Timer Operation

Microsecond timer supports two modes:

- 1 μ s mode and 256 μ s mode.

- In 1 μ s mode, the time unit is 1 μ s. Number of microseconds required to be counted has to be programmed. The maximum value that can be counted is $(2^{32}-1)\mu$ s, i.e 1.2 hours.
- In 256 μ s mode, the time unit is 256 μ s. Number of 256 μ s units required to be counted has to be programmed. This is useful when the timer is being used for counting large time values and microsecond based tracking not required. The maximum value that can be counted is $(2^{32}-1)*256\mu$ s, i.e. nearly equal to 13 days.

One microsecond mode is enabled by setting the MCUULP_TMRn_CNTRL[TMR_MODE] to '1'. In this mode of operation, the value programmed in MCUULP_TMRn_MATCH register is considered in microseconds.

To operate in this mode, the MCUULP_TMR_US_PERIOD_INT register should be programmed with integer part of number of clock cycles per microsecond and MCUULP_TMR_US_PERIOD_FRAC is programmed with the fractional part of number of clock cycles per microsecond depending on the system clock being used(clock period of the system clock used in microseconds). Only the lower significant 8-bits of MCUULP_TMR_US_PERIOD_FRAC are considered. In fractional representation, the nth bit (n=0..7) has the value of $2^{(n-8)}$

256 μ s mode is enabled by setting the MCUULP_TMRn_CNTRL[TMR_MODE] to '2'. In this mode of operation, the value programmed in MCUULP_TMRn_MATCH register is to be in terms of TUs, where 1TU = 256us.

To operate in this mode, the MCUULP_TMR_MS_PERIOD_INT register should be programmed with integer part of number of clock cycles per TU and MCUULP_TMR_MS_PERIOD_FRAC is programmed with the fractional part of number of clock cycles per TU.

17.8.5 Register Summary

Base Address: 0x2404_2000

| Register Name | Offset | Description |
|---------------------------|-----------|--|
| MCUULP_TMR0_MATCH | 0x00 | Timer 0 Match Register |
| MCUULP_TMR0_CNTRL | 0x04 | Timer 0 Control Register |
| MCUULP_TMR1_MATCH | 0x08 | Timer 1 Match Register |
| MCUULP_TMR1_CNTRL | 0x0C | Timer 1 Control Register |
| MCUULP_TMR2_MATCH | 0x10 | Timer 2 Match Register |
| MCUULP_TMR2_CNTRL | 0x14 | Timer 2 Control Register |
| MCUULP_TMR3_MATCH | 0x18 | Timer 3 Match Register |
| MCUULP_TMR3_CNTRL | 0x1C | Timer 3 Control Register |
| Reserved | 0x20-0x7C | |
| MCUULP_TMR_INTR_STAT | 0x80 | Timer Interrupt Status Register |
| MCUULP_TMR_US_PERIOD_INT | 0x84 | Micro-second Timer Period Integer Part Register |
| MCUULP_TMR_US_PERIOD_FRAC | 0x88 | Micro-second Timer Period Fractional Part Register |
| MCUULP_TMR_MS_PERIOD_INT | 0x8C | Milli-second Timer Period Integer Part Register |
| MCUULP_TMR_MS_PERIOD_FRAC | 0x90 | Milli-second Timer Period Fractional Part Register |
| MCUULP_TMR_ACTIVE_STATUS | 0x9C | Timer Active Status Register |

1287 . Timers Register Summary Table

17.8.6 Register Description

Legend:

R = Read-only, W = Write-only, R/W = Read/Write, R/WC = Read/Clear on Write

MCUULP_TMRn_MATCH

n = 0 to 3

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| [31:0] | R/W | TMR_MATCH | 0xffff_ffff | <p>These bits are used to program the timer time out value in microseconds or in number of system clocks.</p> <p>Upon reading, these bits indicate time remaining before timeout.(read as 32'hFFFF_FFFF initially)</p> <p>If counter_up is set these bits directly gives out the up-running counter/timer value.</p> |

1288 .MCUULP_TMRn_MATCH Register Description

MCUULP_TMRn_CNTRL

n = 0 to 3

| Bit | Access | Function | Reset Value | Description |
|------------|---------------|-----------------|--------------------|--|
| [31:8] | R/W | Reserved | 0x0 | Reserved for future use. |
| 7 | R/W | counter_up | 0x0 | For reading/tracking counter in up-counting this bit has to be set. By setting this bit down-counting reading will be lost. |
| 6 | W | TMR_STOP | 0x0 | This bit is used to stop the active timer. '1'- stops the timer, if timer is active (This bit is self clearing bit) |
| 5 | R/W | TMR_MODE | 0x0 | This bit is used to select the mode of working of timer '1'- Periodic timer '0'- One shot time In periodic mode, timer gets reset when timeout occurs and starts again automatically. |
| 4:3 | R/W | TMR_TYPE | 0x0 | These bits are used to select the type of the timer '2'- 256 µs mode '1'-1µs mode '0'- Countdown timer |
| 2 | R/W | TMR_INTR_ENABLE | 0x0 | This bit is used to enable the timeout interrupt. '1'- Interrupt enabled '0'- Interrupt disabled |
| 1 | W | TMR_INTR_CLR | 0x0 | This bit is used to clear the interrupt '1'- Clear interrupt (This bit is self clearing bit) |
| 0 | W | TMR_START | 0x0 | This bit is used to start the timer. Timer gets reset upon setting this bit. '1'- Timer start (This bit is self clearing bit) |

1289 .MCUULP_TMR_CNTRL Register Description

MCUULP_TMR_INTR_STAT

| Bit | Access | Function | Reset Value | Description |
|--------|--------|------------------|-------------|--|
| [31:4] | R | Reserved | 0x0 | Reserved |
| 3 | R/WC | TMR3_INTR_STATUS | 0x0 | This bit indicates the status of the interrupt generated by timer3. '1'- Interrupt present '0'- No interrupt present |
| 2 | R/WC | TMR2_INTR_STATUS | 0x0 | This bit indicates the status of the interrupt generated by timer2. '1'- Interrupt present '0'- No interrupt present |
| 1 | R/WC | TMR1_INTR_STATUS | 0x0 | This bit indicates the status of the interrupt generated by timer1. '1'- Interrupt present '0'- No interrupt present |
| 0 | R/WC | TMR0_INTR_STATUS | 0x0 | This bit indicates the status of the interrupt generated by timer0. '1'- Interrupt present '0'- No interrupt present |

1290 . MCUULP_TMR_STAT Register Description

MCUULP_TMR_US_PERIOD_INT

| Bit | Access | Function | Reset Value | Description |
|---------|--------|-------------------|-------------|---|
| [31:16] | R | Reserved | 0x0 | Reserved |
| [15:0] | R/W | TMR_US_PERIOD_INT | 0xffff | These bits are used to program the integer part of number of clock cycles per microsecond of the system clock being used. |

1291 . MCUULP_TMR_US_PERIOD_INT Register Description

MCUULP_TMR_US_PERIOD_FRAC

| Bit | Access | Function | Reset Value | Description |
|--------|--------|--------------------|-------------|--|
| [31:8] | R | Reserved | 0x0 | Reserved |
| [7:0] | R/W | TMR_US_PERIOD_FRAC | 0xff | These bits are used to program the fractional part of clock cycles per microsecond of the system clock being used. |

1292 . MCUULP_TMR_US_PERIOD_FRAC Register Description

MCUULP_TMR_MS_PERIOD_INT

| Bit | Access | Function | Reset Value | Description |
|---------|--------|------------------------|-------------|---|
| [31:16] | R | Reserved | 0x0 | Reserved |
| [15:0] | R/W | TMR_MS_PERIOD_IN NT | 0xffff | These bits are used to program the integer part of number of clock cycles per 256 microsecond of the system clock being used. |

1293 . MCUULP_TMR_MS_PERIOD_INT Register Description

MCUULP_TMR_MS_PERIOD_FRAC

| Bit | Access | Function | Reset Value | Description |
|--------|--------|------------------------|-------------|--|
| [31:8] | R | Reserved | 0x0 | Reserved |
| [7:0] | R/W | TMR_MS_PERIOD_FR AC | 0xff | These bits are used to program the fractional part of clock cycles per 256 microsecond of the system clock being used. |

1294 . MCUULP_TMR_MS_PERIOD_FRAC Register Description

MCUULP_TMR_ACTIVE_STATUS

| Bit | Access | Function | Reset Value | Description |
|--------|--------|--------------|-------------|---|
| [31:4] | R | Reserved | 0x0 | Reserved |
| [3:0] | R | Timer_active | 0x0 | Timer active status for each timer. LSB bit specifies the status for 0 th timer and so on. For each bit, 1 – Corresponding timer is active 0 – Corresponding timer is inactive |

1295 . MCUULP_TMR_ACTIVE_STATUS Register Description

17.8.7 Programming Sequence

1. Enable Timer module power domain in ULP_Peripheral_Power_Control_SET register as described in [Power Architecture](#).
2. Configure Timer module clock using MCUULP_CLK_EN_REG1 and MCUULP_TIMER_CLK_CONFIG as described in [MCU ULP Clock Architecture](#).
3. To function as Microsecond timer MCUULP_TMRn_CNTRL[TMR_TYPE] should be set accordingly and to function as Counter timer the MCUULP_TMRn_CNTRL[TMR_TYPE] bit should be cleared.
4. Program the maximum count value of the counter in MCUULP_TMRn_MATCH register if counter mode is selected.
5. If the one microsecond mode is selected, program the MCUULP_TMR_US_PERIOD_INT and MCUULP_TMR_US_PERIOD_FRAC registers with integral and fractional part of time period (in number of clocks) of the system clock being used and program the time out value of the timer in microseconds in MCUULP_TMRn_MATCH register.
6. If the 256 micro-second mode is selected, program the MCUULP_TMR_MS_PERIOD_INT and MCUULP_TMR_MS_PERIOD_FRAC registers with integral and fractional part of time period required for 1 TU

(i.e.256 microseconds) and program the time out value of the timer in terms of TU's in the MCUULP_TMRn_MATCH register.

7. To enable the interrupt on timeout, set the MCUULP_TMRn_CNTRL[TMR_INTR_ENABLE] bit.
8. Timers are by default in one shot mode; Periodic operation can be enabled by setting the MCUULP_TMRn_CNTRL[TMR_TYPE] bit.
9. To start the timer set the MCUULP_TMRn_CNTRL[TMR_START] bit.
10. Wait for the timer interrupt, if the timeout interrupt is enabled.
11. Set MCUULP_TMRn_CNTRL[TMR_INTR_CLR], to clear the interrupt generated by timeout.
12. MCUULP_TMR_INTR_STAT[TMRn_INTR_STATUS] can also be monitored to check the timeout status.

17.9 ULP I2C

17.9.1 General Description

There are four I²C Master/Slave controllers - two in the MCU HP peripherals (I2C1, I2C2), one in the NWP/security subsystem and one in the MCU ULP subsystem (ULP_I2C). The I2C interface allows the processor to serve as a master or slave on the I2C bus.

Refer to [I2C Master and slave](#) for the features supported, functional description and programming information. The base address of ULP_I2C is 0x2404_0000.

17.10 ULP UART

17.10.1 General Description

There are three UART controllers - two in the MCU HP peripherals (USART1, USART2) and one in the MCU ULP subsystem (ULP_UART).

Refer to [UART](#) for the features supported, functional description and programming information. The base address of ULP_UART is 0x2404_1800.

17.11 ULP I2S

17.11.1 General Description

There are three I²S controllers - one in the MCU HP peripherals (I2S_2CH), one in the MCU ULP subsystem (ULP_I2S) and one in the security/NWP subsystem. Each I²S controller supports PCM mode of operation also.

I2S is a protocol used for digital stereo audio. It is used in systems that process digital audio signals, such as:

- A/D and D/A converters
- digital signal processors
- error correction for compact disc and digital recording
- digital filters
- digital input/output interfaces

Refer to [I2S/PCM Master and Slave](#) for the features supported, functional description and programming information. The base address of ULP_I2S is 0x2404_0400.

18 UULP VBAT Peripherals

18.1 Bandgap Top

General Description

- This Bandgap Reference provides a voltage reference of 1.2V which is independent of PVT variations and PTAT currents of 20nA and 50nA.
- There is a V2I which provides constant currents of 5nA, 10nA, 20nA only in high power mode.
- It also has reference scaling circuits which provides reference voltages of 0.8-1.1V, 0.75-1.05V, 0.8-1.05 and 0.55-0.8V for ULP_DCDC, 0.85-1.2V for pmu.
- It has a low power mode/sampling mode in which its current consumption is ~15nA.

18.1.1 Features

1. Provides reference of 1.2V for other analog blocks. It can be trimmed using BG_R and BG_R_PTAT to optimize temperature coefficient. Available in both LP mode and HP mode.
2. It provides 20nA, 50nA and 100nA currents to other analog blocks. These currents can be switched on/off using bod_clks_ptat_en and an_perif_ptat_en for ULP and analog_peripherals respectively. Available in both LP mode and HP mode.
3. It provides programmable references of 0.8-1.1V, 0.75-1.05V to SCDC, can be programmed using ref_sel_dcdc. Available in both LP mode and HP mode.
4. It provides programmable reference of 0.8-1.05V to HP and LP Ldos in SCDC, can be programmed using ref_sel_lp_dcdc. Available in both LP mode and HP mode.
5. It provides programmable reference of 0.55-0.8V to 0.6V LDO, can be programmed using LDO_0P6_CTRL. Available in both LP mode and HP mode.
6. It provides programmable reference of 0.85-1.2V to pmu, it will be used as reference in PFM mode of buck, can be programmed using ref_sel_PMU. Available only in HP mode.
7. It provides 5nA, 10nA and 20nA constant currents to pmu, high freq RO and Temp sensor respectively. These currents are available only in HP mode.
8. LP mode (low power mode or sampled mode of bandgap) will be enabled using sampling_en. And will be controlled using bg_en and sh_en clks. These clks are generated using 32KRO/32KRC clocks in ipmu_digital_top.



Note

The voltages can only be programmed using SPI writes, so changing reference voltages is not possible in LP mode.

18.1.2 Functional Description

- UULP_VBATT_AVDD is the power supply for bandgap. All the control signals will be on ULP_VDD_11 and are level shifted to UULP_VBATT_AVDD.
- This is enabled by default. bg_en and sh_en are always low in HP mode.
- $V_{bg} = V_{be} + (\Delta V_{be}/R_{ptat}) * R$, $I_{ptat} = \Delta V_{be}/R_{ptat}$
- BG_R_PTAT trims PTAT current and thereby trims Vbg also. Increasing BG_R_PTAT increases PTAT current and Vbg. Increasing BG_R_PTAT creates positive gradient in Vbg across temperature.
- BG_R trims Vbg. Increasing BG_R reduces Vbg and also create negative gradient in Vbg across temperature.
- If Vbg reduces(increases) with increase in temperature, increasing(decreasing) BG_R_PTAT or decreasing(increasing) BG_R helps.
- The bandgap can enter low power mode only when all high frequency clocks and pmu are off.

Low Power mode

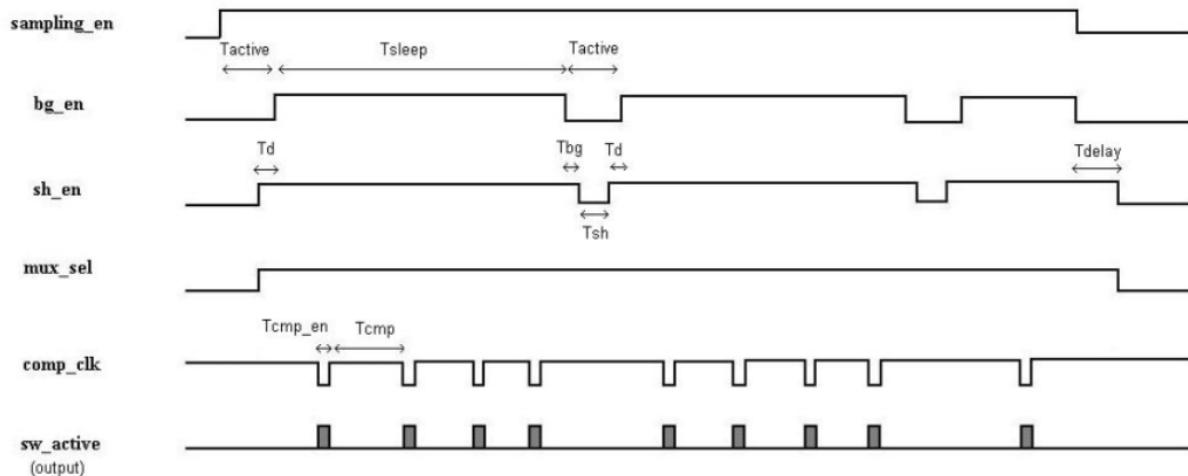
The bandgap can enter low power mode only when all high frequency clocks and pmu are off.

Low power mode has two states ACTIVE and SLEEP. During ACTIVE state bg_en is low(so bandgap will be on) and during SLEEP state bg_en and sh_en are high.

ACTIVE time and SLEEP time can be programmed using bgs_active_timer_sel and bgs_sleep_timer_sel respectively.

If the bandgap output decays faster in SLEEP state, reduce SLEEP time.

If the ACTIVE time is not sufficient (in this case, out will be much lower than 1.2V) , increase ACTIVE time using bgs_active_timer_sel.



18 . Low Power mode

18.1.3 Register Summary

Base Address: 0x2405A400

| Register Name | Offset | Description |
|---------------------------|--------|-------------|
| BG_SLEEP_TIMER_REG | 0x94 | |
| SCDC_CTRL_REG_0 | 0x98 | |
| BG_SCDC_PROG_REG_1 | 0x9C | |
| BG_SCDC_PROG_REG_2 | 0xA0 | |
| BG_LDO_REG | 0xA4 | |
| BG_SCDC_READ_BACK | 0xA8 | |

1296 . Register Summary

18.1.4 Register Description

BG_SLEEP_TIMER_REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|--|
| 31:22 | - | Reserved | - | It is recommended to write these bits to 0. |
| 21:20 | RW | bgs_sleep_timer_sel | 2 | <p>sleep timer count is</p> <p>2'd0: active_timer_count * 2^7</p> <p>2'd1: active_timer_count * 2^8</p> <p>2'd2: active_timer_count * 2^9</p> <p>2'd3: active_timer_count * 2^10</p> |

| | | | | |
|------|----|-----------------------|---|---|
| 19 | RW | bgs_active_timer_sel | 0 | active timer count = 1'b0: 8, 1'b1: 16; if 8, sh_en delay = 5, sh_width = 2 if 16, sh_en_delay = 10, sh_width = 4 time for which sh_en is low in active state = sh_en_width time after which sh_en goes low after bg_en goes low = sh_en_delay |
| 18 | RW | mask_sw_active | 0 | 1: disable comp clock in between sleep duration |
| 17:4 | - | Reserved | - | It is recommended to write these bits to 0. |
| 3 | RW | bg_ctrl_auto | 1 | 1: bg_en and bg_sh_en are automatically controlled 0: bg_en and sh_en take values from SPI. |
| 2 | RW | bypass_pwrgating_cmbi | 0 | Powergating is disabled for combi logic . It will always be ON. |
| 1 | RW | bg_sampling_spi_sel | 0 | enable bandgap sampling through spi / pin 1 = spi ; 0 = pin(sleep_en) |
| 0 | RW | bgs_clk_en | 0 | bandgap sampling enable through spi |

1297 . Band-Gap Sleep Timer Register Description

SCDC_CTRL_REG_0

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------|-------------|---|
| 31:22 | RW | Reserved | - | It is recommended to write these bits to 0. |
| 21 | RW | ext_cap_en | 1'b0 | To change current trim bits to high or low through spi, based on high power or low power mode. When 0, curr prog value is 0. |
| 20:17 | RW | fixed_curr_prog_high | 4'd15 | Current <i>prog value to take when ext cap en is high and sel_high freq_ext_b is 0</i> |
| 16:13 | RW | fixed_curr_prog_low | 4'd0 | Current <i>prog value to take when ext cap en is high and sel_high freq_ext_b is 1</i> |
| 12 | RW | bypass_trim_ro | 1'b0 | To program the trim value manually, irrespective of the fsm |
| 11:7 | RW | fixed_trim_ro | 5'd0 | Manual trim word |
| 6 | RW | fixed_mode | 1'b0 | fixed mode |
| 5:4 | RW | max_mode | 2'd2 | maximum mode it can go to |
| 3:0 | RW | count_reset | 4'hF | Count reset value, count threshold will be doubler this value |

1298 . SCDC Control Register Description

BG_SCDC_PROG_REG_1

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|-------------|
| | | | | |

| | | | | |
|-------|----|------------------|---|--|
| 31:22 | RW | Reserved | - | It is recommended to write these bits to 0. |
| 21:19 | RW | bg_r_ptat | 2 | Bandgap voltage programming |
| 18:16 | RW | bg_r | 0 | Bandgap voltage programming |
| 15 | RW | bg_en | 0 | bg_en from spi |
| 14 | RW | bg_sh_en | 0 | bg_sh_en from spi |
| 13 | - | Reserved | - | It is recommended to write these bits to 0. |
| 12:10 | RW | ref_sel_dcdc | 1 | DCDC output programming vref_1p1/vref_1p05 3'd0 - 1.1/1.05 3'd1 - 1.0/0.95 3'd2 - 0.9/0.85 3'd3 - 0.8/0.75 3'd4 - 1.05/1.0 3'd5 - 0.95/0.9 |
| 9:7 | RW | ref_sel_lp_dcdc | 1 | DCDC output programming in low power mode 3'd0 - 1.05 3'd1 - 1.0 3'd2 - 0.95 3'd3 - 0.9 3'd4 - 0.85 3'd5 - 0.8 |
| 6:5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4 | RW | bod_clks_ptat_en | 1 | 1 - To enable ptat currents to clocks and bod(cmp_npss) |
| 3 | RW | an_perif_ptat_en | 1 | 1 - To enable ptat currents to analog peripherals |
| 2:0 | RW | ref_sel_PMU | 0 | 3'd0 - 1.2V 3'd1 - 1.15V 3'd2 - 1.1V 3'd3 - 1.05V 3'd4 - 1.0V 3'd5 - 0.95V 3'd6 - 0.9V 3'd7 - 0.85V |

1299 . Band-Gap SCDC Control-1 Register Description

BG_SCDC_PROG_REG_2

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------|-------------|--|
| 31:22 | - | Reserved | - | It is recommended to write these bits to 0. |
| 21 | RW | scdc_dc_sel | 0 | To switch to SCDCDC mode from LDO mode. 1 - SCDC mode 0 - LDO mode |
| 20 | RW | testmode_0_en | 0 | Enable for output on to BG_TESTMODE0 |
| 19:18 | RW | testmode_0_sel | 0 | 2'd0: bg_sw_active 2'd1: scdc_dc_sown 2'd2: scdc_dc_lp_mode (sel_high_freq_ext_b) 2'd3: scdc_dc_sel (To select ldo - scdc_dc) |
| 17 | RW | testmode_1_en | 0 | To enable test mux for BG_TESTMODE1 |
| 16:15 | RW | testmode_1_sel | 0 | 2'd0: bg_sh_en 2'd1: scdc_dc_up 2'd2: scdc_dc_en (Enable for scdc_dc block) 2'd3: scdc_dc_lp_en (enable for 10uA LDO) |
| 14 | RW | testmode_2_en | 0 | To enable testmux for BG_TESTMODE2 |

| | | | | |
|-------|----|-------------------|----|---|
| 13:11 | RW | testmode_2_sel | 0 | 3'd0: bg_en 3'd1: bg_comp_clk 3'd2: en_ldo_5m_b 3'd3: comp_clk 3'd4: scdcdc_conv_1b1 3'd5: scdcdc_conv_1b2 3'd6: scdcdc_conv_1b3 3'd7: 0 |
| 10:6 | RW | trim_clamp_lp | 1 | trim value lower clamp value when sel high freq_b is 1 |
| 5:1 | RW | trim_clamp_hp | 16 | trim value lower clamp value when sel high freq_b is 0 |
| 0 | RW | scdcdc_soft_reset | 0 | soft reset signal for scdcdc fsm |

1300 . Band-Gap SCDC Control-2 Register Description

BG_LDO_REG1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|--|
| 31:22 | - | Reserved | - | It is recommended to write these bits to 0. |
| 21 | RW | LDO_OP6_BYPASS | 1'b0 | bypass signal for DCDC1p1_lp_500uA |
| 20:18 | RW | LDO_OP6_CTRL | 3'd2 | vref for DCDC1p1_lp_500uA 3'd0 - 0.8V 3'd1 - 0.75V 3'd2 - 0.7V 3'd3 - 0.65V 3'd4 - 0.6V 3'd5 - 0.55V |
| 17 | - | Reserved | - | Reserved |
| 16 | RW | LDO_OP6_LP_MODE | 1'b0 | enable low power mode, otherwise in high power mode |
| 15 | RW | LDO_OP6_ENABLE | 1'b1 | enable digital LDO |
| 14:5 | - | Reserved | - | It is recommended to write these bits to 0. |
| 4 | RW | test_amux_en | 1'b0 | Enable analog mux to test reference voltages |
| 3:1 | RW | test_amux_sel | 3'd0 | Select for analog mux 3'd0: vref_1p1 3'd1: vref_1p05 3'd2: vref_0p6 3'd3: vref_ulp 3'd4: vref_pmu |
| 0 | - | Reserved | - | It is recommended to write these bits to 0. |

1301 . Band-Gap LDO Control Register Description

BG_SCDC_READ_BACK

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:22 | - | Reserved | - | - |

| | | | | |
|-------|---|------------------|----|--|
| 21:18 | R | scdcdc_curr_prog | 0 | Scdcdc curr prog read back |
| 17:13 | R | trim_4mhz_ro | 16 | Trim value for scdcdc ring oscillator |
| 12:1 | - | Reserved | - | - |
| 0 | R | sync_reset_read | 0 | Read back for sync reset with ro clock |

1302 . Band-Gap Read-Back Register Description

18.2 BOD

18.2.1 General Description

The Nano Power BOD consists of a comparator, reference buffer, scaler and a resistor bank.

The comparator compares inputs p and n to produce an output, cmp_out.

$p > n$, cmp_out = 1

$p < n$, cmp_out = 0

18.2.2 Features

The following cases of comparison are possible

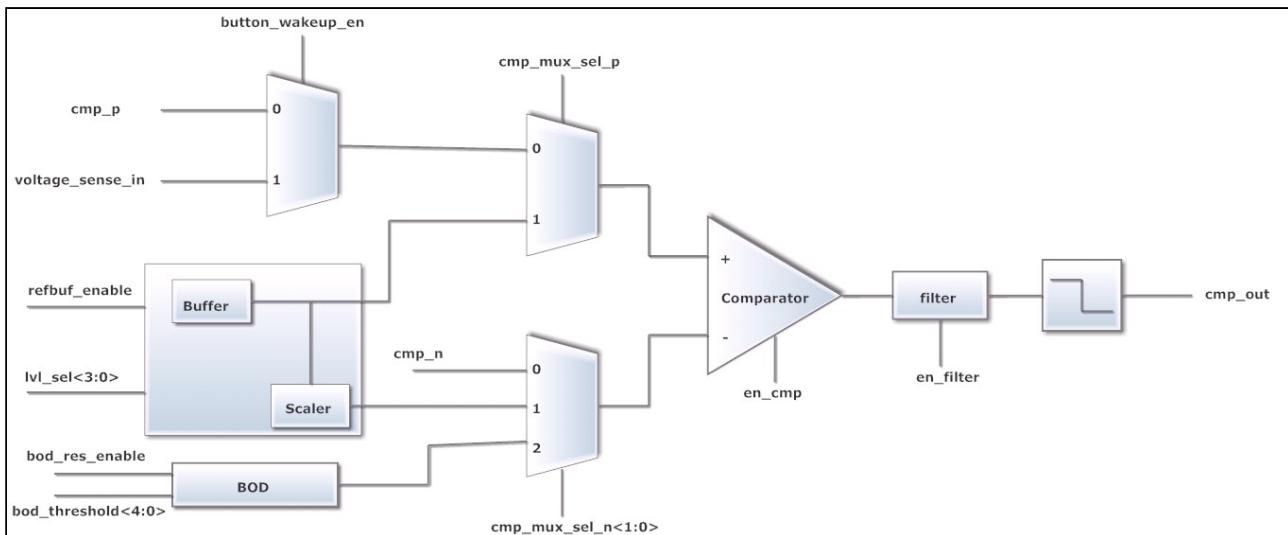
1. Compare external pin inputs
2. Compare external pin input to internal voltages.
3. Compare internal voltages.

The comparator has to be enabled to enable reference buffer and resistor bank.

The reference buffer buffers the bandgap reference voltage and also has a voltage divider which gives programmable 100mV to 1.1V output.

The resistor bank is used to detect battery voltage from 1.75V to 3.65V with a 50mV step. Since each bod_threshold value refers to a particular battery voltage, the battery voltage is found by comparing resistor bank output with reference buffer output using comparator for different bod_threshold values.

18.2.3 Block Diagram



18.2.4 GPIO Pins

| Pin | GPIO |
|-------|------------------|
| cmp_p | UULP_VBAT_GPIO_3 |

| Pin | GPIO |
|------------------|------------------|
| cmp_n | UULP_VBAT_GPIO_4 |
| voltage_sense_in | UULP_VBAT_GPIO_2 |

 UULP_VBATT_AD_GPIO_x has to be in mode 7

18.2.5 Functional Description

- ULP_VDD_11 and UULP_VBATT_AVDD supplies should be available for the comparators to work.
- Enable reference buffer and resistor bank if required.
- Select the inputs of comparator using input mux selects and then enable the comparator. By default comparator is disabled.

18.2.6 Voltage Scaler

The reference buffer uses 1.2V from ULP_BG as its reference. And the scaler takes this buffered 1.2V as its input. Scaler is configured using lvl_sel<3:0>. The output of scaler for different scale factors are in the table below

| Bandgap_scale_factor | Scaler output | Units |
|----------------------|---------------|-------|
| 0 | 0.1 | V |
| 1 | 0.2 | V |
| 2 | 0.3 | V |
| 3 | 0.4 | V |
| 4 | 0.5 | V |
| 5 | 0.6 | V |
| 6 | 0.7 | V |
| 7 | 0.8 | V |
| 8 | 0.9 | V |
| 9 | 1 | V |
| 10 | 1.1 | V |

18.2.7 Resistor Bank (BOD)

BOD is configured using bod_threshold<5:0>. If the battery voltage goes lower than the referred voltage, the comparator output becomes high. The referred voltages for 0 to 38 are shown in the table below. Words 39 to 63 are mapped to voltage corresponding to word 38.

$$\text{Resbank output} = \text{VBATT} * \text{Resbank_output_fraction} = \text{VBATT} * (200 / (290 + \text{bod_threshold} * 8.33))$$

| bod threshold | Resbank_output_fraction | Referred voltage | Unit |
|---------------|-------------------------|------------------|------|
| 0 | 0.69 | 1.75 | V |

| bod threshold | Resbank_output_fraction | Referred voltage | Unit |
|----------------------|--------------------------------|-------------------------|-------------|
| 1 | 0.67 | 1.8 | V |
| 2 | 0.652 | 1.85 | V |
| 3 | 0.635 | 1.9 | V |
| 4 | 0.618 | 1.95 | V |
| 5 | 0.603 | 2 | V |
| 6 | 0.588 | 2.05 | V |
| 7 | 0.574 | 2.1 | V |
| 8 | 0.561 | 2.15 | V |
| 9 | 0.548 | 2.2 | V |
| 10 | 0.536 | 2.25 | V |
| 11 | 0.524 | 2.3 | V |
| 12 | 0.513 | 2.35 | V |
| 13 | 0.502 | 2.4 | V |
| 14 | 0.492 | 2.45 | V |
| 15 | 0.482 | 2.5 | V |
| 16 | 0.472 | 2.55 | V |
| 17 | 0.463 | 2.6 | V |
| 18 | 0.455 | 2.65 | V |
| 19 | 0.446 | 2.7 | V |
| 20 | 0.438 | 2.75 | V |
| 21 | 0.43 | 2.8 | V |
| 22 | 0.423 | 2.85 | V |
| 23 | 0.415 | 2.9 | V |
| 24 | 0.408 | 2.95 | V |
| 25 | 0.401 | 3 | V |
| 26 | 0.395 | 3.05 | V |
| 27 | 0.388 | 3.1 | V |
| 28 | 0.382 | 3.15 | V |
| 29 | 0.376 | 3.2 | V |
| 30 | 0.37 | 3.25 | V |
| 31 | 0.365 | 3.3 | V |
| 32 | 0.359 | 3.35 | V |

| bod threshold | Resbank_output_fraction | Referred voltage | Unit |
|---------------|-------------------------|------------------|------|
| 33 | 0.354 | 3.4 | V |
| 34 | 0.349 | 3.45 | V |
| 35 | 0.344 | 3.5 | V |
| 36 | 0.339 | 3.55 | V |
| 37 | 0.334 | 3.6 | V |
| 38 | 0.33 | 3.65 | V |

18.2.8 Input Modes

| Mode | Mode Name | Input 1 | Input 2 | Enable Bit | Interrupt Generated | manual_cmp_mux_sel (if in manual mode) |
|------|---------------|--------------------------------|--------------|-------------------|---------------------|---|
| 1 | | cmp_p | cmp_n | cmp1_en | cmp_intr_1 | 3'd0 |
| 2 | | cmp_p | v1p2_scaled | cmp2_en | cmp_intr_2 | 3'd1 |
| 3 | | cmp_p | VBATT_scaled | cmp3_en | cmp_intr_3 | 3'd2 |
| 4 | | v1p2_buffered | cmp_n | cmp4_en | cmp_intr_4 | 3'd3 |
| 5 | BOD | v1p2_buffered | VBATT_scaled | cmp5_en | cmp_intr_5 | 3'd4 |
| 6 | Button Wakeup | Button Wakeup voltage_sense_in | VBATT_scaled | button_wakeu_p_en | cmp_intr_6 | 3'd5 |

18.2.9 Comparison Modes

It has two modes, Auto comparison mode and Manual comparison mode.

Auto comparison mode

In this mode all the 6 comparisons are made in 6 consecutive clock cycles in a slot in regular intervals if all the comparisons are enabled i.e., cmp_[1-5]_en and button_wakeup_en. (or whatever comparisons are enabled)

Manual Comparison mode

In manual mode only one particular comparison keeps happening. Select which comparison is to be made using manual_cmp_mux_sel. Also enable the respective cmp_en signal.

18.2.10 Button wakeup

Up to 3 buttons can be detected. When the voltage output from button press falls into any of the following regions (defined using Resbank_Output_Fraction in Resistor Bank section), button detection occurs.

Region for button 1 = {button1_min_value: button1_min_value + button_max_value};

Region for button 2 = {button2_min_value: button2_min_value + button_max_value};

Region for button 3 = {button3_min_value: button3_min_value + button_max_value};

Conditions to avoid false detection:

- Regions should be non-overlapping.
- Region should not contain values in the extremes near 0 and the highest values.
- The min and max values must be programmed based on the resistor values used around buttons.

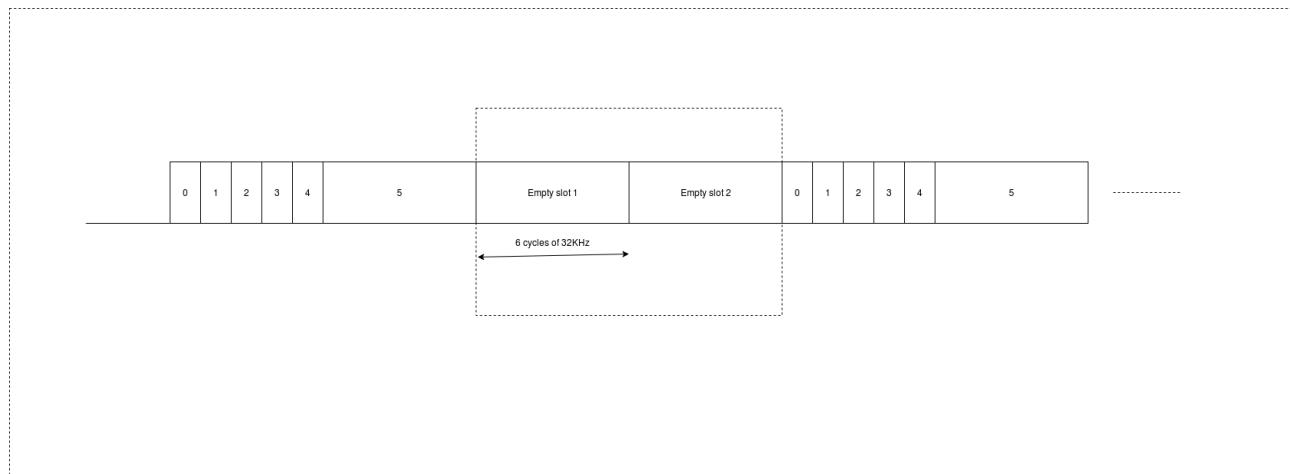
In manual mode, Button wakeup is checked for continuously using a FSM. When a region is detected, cmp_6_intr is raised and sent to FSM and further action is taken. In order to check which button is detected read spi register 1E3 for button1_wakeup, button2_wakeup and button3_wakeup. The corresponding bit will be high.

18.2.11 Slotting

By programming cmp_slot_value , number of empty slots in between 2 slots can be programmed. Each empty slot consists of 6 cycles of 32KHz (ulp_fsm_clk).

In the below figure, 0 to 5 are the 6 modes of comparator and cmp_slot_value is programmed as 2.

- mode 5 is button wakeup, its length is 6 cycles when button wakeup mode is enabled and 1 when disabled.



18.2.12 Register Summary

| Register Name | Offset | Reset Value | Description |
|------------------------------|--------|-------------|-------------|
| BOD_COMP_MODE_REG | 1E0 | | |
| BOD_COMP_SEL_REG | 1E1 | | |
| BOD_BUTTON_REG | 1E2 | | |
| BOD_TEST,PG&VBATT_STATUS_REG | 1E3 | | |

1303 . Register Summary

18.2.13 Register Description

NANO POWER BOD

BOD_COMP_MODE_REG

BOD : Address : 1E0

| Access | Bits | Register | Default value | Description |
|--------|-------|--------------------|---------------|--|
| R/W | 21 | auto_cmp_mode_en | 1'b0 | 1: Enable auto slotting of comparator inputs auto comparison mode enable 0: Check for manual mode |
| R/W | 20 | manual_cmp_mode_en | 1'b0 | Manual comparison mode en ; 1: Manually select the inputs for comparator comparision 0: Manual mode disabled |
| R/W | 19:17 | manual_cmp_mux_sel | 3'd0 | Selecting and fixing the inputs of comparator when in manual mode. |
| R/W | 16:1 | cmp_slot_value | 16'd0 | Slot value after which comparator outputs are sampled in auto mode. |
| R/W | 0 | cmp_op_filter_en | 1'b0 | Enable signal for filter at comparator output. |

BOD_COMP_SEL_REG

BOD : Address : 1E1

| Access | Bits | Register | Default value | Description |
|--------|---------------|------------------|---------------|---|
| | Combinational | cmp_en_reg_wr | 1'b0 | Whenever any of the cmp_ens change or this register of spi is written cmp_en_reg_wr = 1; |
| R/W | 21 | button_wakuep_en | 1'b0 | Enable button wakeup |
| R/W | 20 | cmp_1_en | 1'b0 | Enables comparision 1 |
| R/W | 19 | cmp_2_en | 1'b0 | Enables comparision 2 |
| R/W | 18 | cmp_3_en | 1'b0 | Enables comparision 3 |

| NANO POWER BOD | | | | |
|----------------------------|-------------|----------------------|----------------------|--|
| R/W | 17 | cmp_4_en | 1'b0 | Enable signal for comparision 4 |
| R/W | 16 | cmp_5_en | 1'b0 | Enable signal for bod detection |
| R/W | 15 | cmp_1_polarity | 1'b0 | 1: polarity of comparator is reversed 0: Same op of comparator is taken |
| R/W | 14 | cmp_2_polarity | 1'b0 | 1: polarity of comparator is reversed 0: Same op of comparator is taken |
| R/W | 13 | cmp_3_polarity | 1'b0 | 1: polarity of comparator is reversed 0: Same op of comparator is taken |
| R/W | 12 | cmp_4_polarity | 1'b0 | 1: polarity of comparator is reversed 0: Same op of comparator is taken |
| R/W | 11 | cmp_5_polarity | 1'b0 | 1: polarity of comparator is reversed 0: Same op of comparator is taken |
| R/W | 10:7 | bandgap_scale_factor | 4'd0 | Programmability for scaling bandgap v1p2 |
| R/W | 6:1 | batt_scale_factor | 6'd0 | Programmability for scaling vbatt |
| R | 0 | Reserved | 1'd0 | Reserved |
| BOD_BUTTON_REG | | | | |
| BOD : Address : 1E2 | | | | |
| Access | Bits | Register | Default value | Description |
| R | 21 | sync_reset_read | 1'b0 | read back synced reset with 32KHz fsm clock |
| R/W | 20 : 17 | button_max_value | 4'd0 | MAximum range for each button wakeup detect |

| NANO POWER BOD | | | | |
|---|-------------|--|----------------------|--|
| R/W | 16 : 12 | button1_min_value | 5'd0 | Minimum threshold value to detect button 1 wakeup |
| R/W | 11 : 7 | button2_min_value | 5'd0 | Minimum value for button 2 detect |
| R/W | 6 : 2 | button3_min_value | 5'd0 | Minimum value for button 3 detect (calib word lies in the range of min3_value to min3_value + button_max_value) |
| R/W | 1:0 | comparator hysteresis | 2'd0 | comparator hysteresis |
| BOD_TEST,PG&VBATT_STATUS_REG | | | | |
| BOD : Address : 1E3 | | | | |
| Access | Bits | Register | Default value | Description |
| R/W | 21 | en_bod_test_mux | 1'd0 | To enable test mux to connect to GPIO pad |
| R/W | 20:19 | bod_test_sel | 2'd0 | Select bits for test mux |
| R | 18 | button1_wakeup (read) | 1'b0 | 1 => button 1 detected for wakeup 0 => button 1 not detected for wakeup |
| R | 17 | button2_wakeup (read) | 1'b0 | button 2 wakeup status |
| R | 16 | button3_wakeup (read) | 1'b0 | button 3 wakeup status |
| R/W | 15 | bod_pwrgate_en_n <ul style="list-style-type: none">1'b1p_button_calib | | pwrgate enable signal for button calib and vbatt status checking block |
| R/W | 14 | blackout_en | 1'b0 | 0: Force disable the blackout block 1: Enables black out in active state and when brown out is detected. |
| R/W | 13:9 | brown_out_interrupt_threshold | 5'd2 | Threshold for brown out detection beyond which interrupt is not given to NPSS |
| R/W | 8 | periodic_trigger_en | 1'b0 | Periodic checking for battery status enable |

| NANO POWER BOD | | | | |
|-----------------------|-----|--------------------|------|--|
| only W | 7 | check_vbatt_status | 1'b0 | pulse signal, combinational logic. to check battery status |
| R | 6 | vbatt_status_valid | 1'b0 | Valid signal for reading vbatt status |
| R | 5:0 | vbatt_status | 1'b0 | Status of battery, 31 -> full , 0 -> low |

| BOD : Address : 1E4 | | | | |
|----------------------------|-------------|------------------|----------------------|-----------------------------------|
| Access | Bits | Register | Default value | Description |
| R/W | 21:16 | read_button_word | 6'd0 | Word captured when button pressed |
| - | 15:0 | reserved | 0 | Reserved |

1304 . Register Description

18.3 Calendar

18.3.1 General Description

Calender block acts a RTC with time in seconds, minutes, hours, days, months, years and centuries. The real time can also be read through APB with accuracy less than a second by reading the milli second count value and further less also by reading the number of counts of APB clock in 1 milli second of RTC clock. Accuracy is high.

18.3.2 Features

- Calendar block can provide a seconds trigger and also a msec trigger.
- Calender block takes care of no.of days in each month and also leap years. It can count up to 4 centuries.
- Real time is readable through APB and also programmable through APB.
- Option to choose either RC clock RO clock as calendar clock.

18.3.3 Functional Description

Calendar block counts time based on the 24 bit time_period measured. Calender block has separate counters for milliseconds (7 bits: 0 - 124), 1/8th seconds (3 bits: 0 - 7), seconds (6 bits: 0 - 59), minutes (6 bits: 0 - 59), hours (5 bits: 0 - 23), days(5 bits: 1 - 31), months (4 bits: 1 - 12), years (7 bits: 0 - 99) and centuries(2 bits: 0 - 3). All these counters can be programmed to a required value through APB by programming prog_time_trig bit to 1.

Every 4th year, year[1:0] == 00 is considered a leap year. The clock input to this block can be selected through APB by rtc_clk_sel bit. rtc_clk_sel = 0 : RC clock and rtc_clk_sel = 1 : RO clock.

18.3.4 Register Summary

Base Address: 0x2404_8200

| Register Name | Offset | Description |
|----------------------|--------|-------------|
| MCU CAL ALARM PROG 1 | 0x1C | |

| | | |
|----------------------------|------|--|
| MCU CAL ALARM PROG 2 | 0x20 | |
| MCU CAL POWERGATE REG | 0x24 | |
| MCU CAL PROG TIME 1 | 0x28 | |
| MCU CAL PROG TIME 2 | 0x2C | |
| MCU CAL READ TIME MSB | 0x30 | |
| MCU CAL READ TIME LSB | 0x34 | |
| MCU CAL READ COUNT TIMER | 0x38 | |
| MCU CAL SLEEP CLK COUNTERS | 0x3C | |
| MCU CAL KEY EANBLE | 0x40 | |

1305 . Example Table Header

18.3.5 Register Description

MCU CAL ALARM PROG 1

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------|-------------|-----------------------------------|
| 31:27 | | | | |
| 26:22 | R/W | prog_alarm_hour | 0 | hours value of alarm time |
| 21:16 | R/W | prog_alarm_min | 0 | mins value of alarm time |
| 15:10 | R/W | prog_alarm_sec | 0 | seconds value of alarm time |
| 9:0 | R/W | prog_alarm_msec | 0 | milli seconds value of alarm time |

1306 .MCU CAL ALARM PROG 1 Register Description

MCU CAL ALARM PROG 2

| Bit | Access | Function | Default Value | Description |
|-------|--------|--------------------|---------------|---|
| 31 | R/W | alarm_en | 0 | 1-alarm function enable for calendar alarm interrupt is generated when real time matches alarm time |
| 30:25 | | | | |
| 24:23 | R/W | prog_alarm_century | 0 | century count in alarm time |
| 22:16 | R/W | prog_alarm_year | 0 | year count in alarm time 0 - 99 |
| 15:12 | -- | -- | 0 | -- |
| 11:8 | R/W | prog_alarm_month | 0 | month count in alarm time |
| 7:5 | -- | Reserved | 0 | Reserved |
| 4:0 | R/W | prog_alarm_day | 0 | day count in alarm time 1-31 |

1307 .MCU CAL ALARM PROG 2 Register Description

MCU CAL POWERGATE REG

| Bit | Access | Function | Default Value | Description |
|------|--------|-----------------------|---------------|---|
| 31:2 | | | | |
| 1 | R/W | enable_calender_combi | 1'b0 | Enable calender combinational logic block |
| 0 | R/W | pg_en_calender | 1'b1 | Start calender block |

1308 . MCU CAL POWERGATE REG Register Description



Note

Only Accessible if RTC_KEY is Enabled

MCU CAL PROG TIME 1

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------|---------------|--|
| 31:27 | | | | |
| 26:22 | R/W | prog_hour | 5'd0 | hours value to be programmed to real time in calendar when pro_time_trig is 1 0 - 23 |
| 21:16 | R/W | prog_min | 6'd0 | minutes value to be programmed to real time in calendar when pro_time_trig is 1 0- 59 |
| 15:10 | R/W | prog_sec | 6'd0 | seconds value to be programmed to real time in calendar when pro_time_trig is 1 0 - 59 |
| 9:0 | R/W | prog_msec | 10'd0 | Milli seconds value to be programmed to real time in calendar when pro_time_trig is 1 0-999 |

1309 . MCU CAL PROG TIME 1 Register Description



Note

Only Accessible if RTC_KEY is Enabled

MCU CAL PROG TIME 2

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------|---------------|--|
| 31 | W | prog_time_trig | 0 | 1 - load the programmed to the real time in calendar block |
| 30:25 | | | | |

| Bit | Access | Function | Default Value | Description |
|-------|--------|---------------|---------------|---|
| 24:23 | R/W | prog_century | 0 | century value to be programmed to real time in calendar when pro_time_trig is 1 0 - 3 |
| 22:16 | R/W | prog_year | 0 | year value to be programmed to real time in calendar when pro_time_trig is 1 0 - 99 |
| 15:12 | | | | |
| 11:8 | R/W | prog_month | 0 | month value to be programmed to real time in calendar when pro_time_trig is 1 1-12 |
| 7:5 | R/W | prog_week_day | 0 | program which week day it is |
| 4:0 | R/W | prog_day | 5'd0 | day count value to be programmed to real time in calendar when pro_time_trig is 1 1-31 |

1310 . MCU CAL PROG TIME 2 Register Description



Note

Only Accessible if RTC_KEY is Enabled

MCU CAL READ TIME MSB

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|---|
| 31:16 | | | | |
| 15:0 | R | Real_time[47:32] | -- | Read value of current time in calendar block real time = {century_count , year_count, months, week days,days,hours, mins, secs, milliseconds} real_time[47:46] = century count real_time[45:39] = year_count real_time[38:35] = months_count real_time[34:32] = week day |

1311 . MCU CAL READ TIME MSB Register Description

MCU CAL READ TIME LSB

| Bit | Access | Function | Reset Value | Description |
|------|--------|-----------------|-------------|--|
| 31:0 | R | Real_time[31:0] | -- | Read value of current time in calendar block real time = {century_count , year_count, months, week days,days,hours, mins, secs, milliseconds} real_time[31:27] = days_count real_time[26:22] = hours_count real_time[21:16] = mins_count real_time[15:10] = secs count real_time[9:0] = milliseconds count |

1312 . MCU CAL READ TIME LSB Register Description

MCU CAL SLEEP CLK COUNTERS

| Bit | Access | Function | Reset Value | Description |
|-------|--------|-----------------------------|-------------|---|
| 31:28 | | | | |
| 27:16 | R | pclk_count_wrt_sleep --_clk | -- | no. of APB clks in 1 sleep clock duration |
| 15:12 | | | | |
| 11:0 | R | sleep_clk_duration | -- | No of sleep clks with respect to APB clock so far from the posedge of sleep clk |

1313 . MCU CAL SLEEP CLK COUNTERS Register Description

MCU CAL READ COUNT TIMER

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|---|
| 31:27 | | | | |
| 26:0 | R | read_count_timer | -- | Read timer which increments by time period value to reach to count milliseconds |

1314 . MCU CAL READ COUNT TIMER Register Description

MCU CAL KEY EANBLE

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|--|
| 31:0 | W | rtc_key | 0x55555555 | <p>Program the below key to enable access to program Watch dog registers 0x8D845F4C</p> <p>Program the below key to disable access to program Watch dog registers 0x55555555</p> |

1315 . MCU CAL KEY EANBLE Register Description

18.4 GPIO Timestamp

18.4.1 General Description

The Block is used for capturing the Timestamp of GPIO signal going high from SLEEP to Active state.

18.4.2 Features

- Option to choose 1 GPIO out 5 UULP GPIO's for time stamping application.
- Uses a 32MHz RC clock for time stamping.

18.4.3 Programming Sequence

Configuring

- GPIO Timesampling is available only during Sleep state.
- Choose one GPIO on which Time stamping is required.
 - program 'timestamping_on_gpio0' register in address **MCU_GPIO_TIMESTAMP_CONFIG** to get GPIO timestamp of UULP_GPIO_0.
 - program 'timestamping_on_gpio1' register in address **MCU_GPIO_TIMESTAMP_CONFIG** to get GPIO timestamp of UULP_GPIO_1.
 - program 'timestamping_on_gpio2' register in address **MCU_GPIO_TIMESTAMP_CONFIG** to get GPIO timestamp of UULP_GPIO_2.
 - program 'timestamping_on_gpio3' register in address **MCU_GPIO_TIMESTAMP_CONFIG** to get GPIO timestamp of UULP_GPIO_3.
 - program 'timestamping_on_gpio4' register in address **MCU_GPIO_TIMESTAMP_CONFIG** to get GPIO timestamp of UULP_GPIO_4.
- Program 'enable_gpio_timestamping' register in address **MCU_GPIO_TIMESTAMP_CONFIG** for enabling time stamping application.

Reading

- Poll register 'timestamping_done' in address **MCU_GPIO_TIMESTAMP_CONFIG**. When the signal goes high will indicated the timestamp values is ready for reading.
- Read register **MCU_GPIO_TIMESTAMP_READ** to get the GPIO TimeStamp.

- Value of 'gpio_event_count_partial' will indicating number for 32MHz clock present in 1 Sleep clock.
- Value of 'gpio_event_count_full' will indicating the duration from GPIO going high to first Sleep clock posedge from GPIO going high with respect to 32MHz clock.

Re-initiating

- Once timestamp value is read. Write to address **MCU_GPIO_TIMESTAMP_READ** with value 0 to clear timestamp values so that is can get timestamp of next GPIO event.

18.4.4 Register Summary

Base Address: 0x2404_8100

| Register Name | Offset | Description |
|----------------------------------|--------|-------------|
| MCU_GPIO_TIMESTAMP_CONFIG | 0x28 | |
| MCU_GPIO_TIMESTAMP_READ | 0x2C | |

18.4.5 Register Description

MCU_GPIO_TIMESTAMP_CONFIG

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|--------------------------|-------------|---|
| 31 | R | timestamping_done | 0 | This signal indicated Timestamp of GPIO is ready for reading. |
| 30:6 | | | | |
| 5 | R/W | timestamping_on_gpio4 | 0 | Enable GPIO time stamping on GPIO4 |
| 4 | R/W | timestamping_on_gpio3 | 0 | Enable GPIO time stamping on GPIO3 |
| 3 | R/W | timestamping_on_gpio2 | 0 | Enable GPIO time stamping on GPIO2 |
| 2 | R/W | timestamping_on_gpio1 | 0 | Enable GPIO time stamping on GPIO1 |
| 1 | R/W | timestamping_on_gpio0 | 0 | Enable GPIO time stamping on GPIO0 |
| 0 | R/W | enable_gpio_timestamping | 0 | Enable GPIO time stamping Feature. This will enable measurement of GPIO high duration from SLEEP to wakeup |



Note

The GPIO used for doing timespaming application are UULP GPIO's

MCU_GPIO_TIMESTAMP_READ

| Bit | Access | Function Name | Reset Value | Description |
|-------|--------|--------------------------|-------------|--|
| 31:27 | | | | |
| 26:16 | R | gpio_event_count_partial | | Counter value indicating number for 32MHz clock present in 1 Sleep clock (MCU FSM Clock) |
| 15:11 | | | | |
| 10:0 | R | gpio_event_count_full | 0 | Counter value indicating the duration from GPIO going high to first Sleep clock(MCU FSM Clock) posedge from GPIO going high with respect to 32MHz clock. |

18.5 POC

18.5.1 General Description

The POC (poc + blackout monitor) cell generates a POC (Power On Control) signal that is distributed to all I/O cells to prevent the I/O cells from powering up in undesired configuration and is also used inside the IC to safe state the IC till a valid supply is available for proper operation of the IC. This power management is functional in both power up and power down sequences.

During power up, until the core supply (VDD) reaches 0.7V and till the IO supply (DVDD) reaches 1.8V, the POC signal stays high. Once the core supply exceeds 0.7V and IO supply exceeds 1.8V POC becomes low and normal operation of the IC starts.

Once the POC becomes low, it stays low irrespective of VDD voltage. But if DVDD becomes lower than 1.8V, POC becomes high.

18.5.2 Features

1. This block has trim to adjust threshold voltage. By default RESBANK_TRIM is 2'd0, and threshold voltage is expected to be 1.8V.
2. In low power modes, this can be disabled using blackout_en signal. BOD will be functional at that time and periodically check the supply, once BOD occurs blackout will be automatically switched on.
3. If there is performance issue with VOUTSCDC0P6 or low power ldo mode for VOUTSCDC1P1, make trim_0p5nA1 and trim_0p5nA2 high one by one. It increases bias current for these ldos.

18.5.3 Functional Description

Finding Blackout Threshold

POC1_MANUF

Blackout Threshold Trim

By default trim is 2'd0. If the threshold is different from required value of 1.8V, then change the trim as shown below

| Expected threshold (V) | Observed Threshold (V) | New trim | New threshold (V) |
|------------------------|------------------------|----------|-------------------|
| 1.8 | 1.8 | 2'd0 | ~ 1.8 |

| Expected threshold (V) | Observed Threshold (V) | New trim | New threshold (V) |
|------------------------|------------------------|----------|-------------------|
| 1.8 | 1.7 | 2'd1 | ~ 1.8 |
| 1.8 | 1.6 | 2'd2 | ~ 1.8 |
| 1.8 | 1.95 | 2'd3 | ~ 1.8 |

18.5.4 Register Summary

IPMU registers

| Register Name | offset |
|------------------------------|---------|
| SPARE_REG_1 | 10'h140 |
| BOD_TEST,PG&VBATT_STATUS_REG | 10'h1E3 |

1316 . Register Summary

18.5.5 Register Description

Register Description

18.6 Secure Storage

18.6.1 General Description

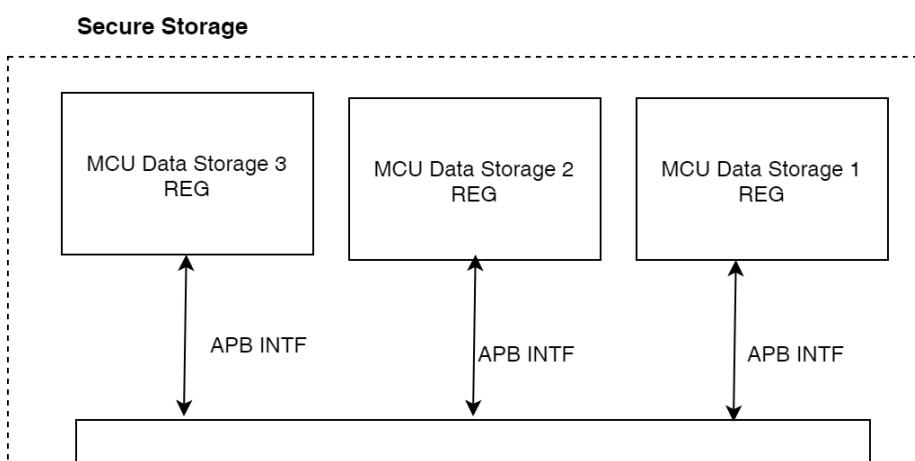
The Block is used for storing configuration values with data protection feature.

18.6.2 Features

- MCU has 3 set's for storage block
 - First chunk is 64 bits.
 - Second chunk is 64 bits &
 - Third Chunks is 128 bits
- Each chunk is a power domain.
- Secure mode is available for first and second Chunk.
- Storage space can be used for storing Configuration values

18.6.3 Functional Description

Block Diagram





Block Diagram of Secure Storage

18.6.4 Programming Sequence

Writing to MCU Storage Domain

1. Enable Write Access to TASS Storage domain by programming to MCU_STORAGE_WRITE_KEY register. Please see register description.
2. Write to register MCU_STORAGE_REG0 with required value.
3. Write to register MCU_STORAGE_REG1 with required value.
4. Write to register MCU_STORAGE_REG2 with required value.
5. Write to register MCU_STORAGE_REG3 with required value.
6. Write to register MCU_STORAGE_REG4 with required value.
7. Write to register MCU_STORAGE_REG5 with required value.
8. Write to register MCU_STORAGE_REG6 with required value.
9. Write to register MCU_STORAGE_REG7 with required value.
10. Disable Write Access to TASS Storage domain by programming to MCU_STORAGE_WRITE_KEY register. Please see register description.

Reading from MCU Storage Domain

1. Read from register MCU_STORAGE_REG0 to fetch required value.
2. Read from register MCU_STORAGE_REG1 to fetch required value.
3. Read from register MCU_STORAGE_REG2 to fetch required value.
4. Read from register MCU_STORAGE_REG3 to fetch required value.
5. Read from register MCU_STORAGE_REG4 to fetch required value.
6. Read from register MCU_STORAGE_REG5 to fetch required value.
7. Read from register MCU_STORAGE_REG6 to fetch required value.
8. Read from register MCU_STORAGE_REG to fetch required value.

18.6.5 Secure Mode

There are two options available:

1. Reset Protection
2. Power Domain Protection from accidental turn-off of power domain controls of Storage domain.

The feature will be enabled by Boot load code.

Reset Protection

When Bit[2] is Set in register **NWPAON_POR_CTRL_BITS**. Storage domain's will be immune to Reset from Pin, WDT Reset and Host Reset Request.



Note

Once the Bit is set it can not be cleared.

Power Domain Protection

When **Write_protect/Bit[4]** is set in register **NWPAON_POR_CTRL_BITS**. Storage domain's are protected from accidental turn-off Power-Supply to these blocks and Once data is written to the protected registers is can not be over-written again



Note

Once the Bit is set it can not be cleared.

18.6.6 Register Summary

Base Address: 0x2404_8500

| Register Name | Offset | Description |
|-----------------------|--------|-------------|
| MCU_STORAGE_REG0 | 0x80 | |
| MCU_STORAGE_REG1 | 0x84 | |
| MCU_STORAGE_REG2 | 0x88 | |
| MCU_STORAGE_REG3 | 0x88 | |
| MCU_STORAGE_REG4 | 0x90 | |
| MCU_STORAGE_REG5 | 0x94 | |
| MCU_STORAGE_REG6 | 0x98 | |
| MCU_STORAGE_WRITE_KEY | 0x9C | |

1317 . Base Address: 0x2404_8500

Base Address: 0x2404_8700

| Register Name | Offset | Description |
|------------------|--------|-------------|
| MCU_STORAGE_REG0 | 0x00 | |

1318 . Base Address: 0x2404_8700

18.6.7 Register Description

MCU_STORAGE_REG0

| Bit | Access | Function Name | Reset Value | Description |
|-------|--------|-------------------------|-------------|--|
| 31:0] | R/W | MCU_STORAGE_WOR0 D_0 | | This register Can be used to storing 32bits of Data. If Write_protect is set and a value is written to register, then data will not be overwritten |

1319 . MCU_STORAGE_REG0 Register Description

MCU_STORAGE_REG1

| Bit | Access | Reset Value | Reset Value | Description |
|------|--------|-----------------------|-------------|--|
| 31:0 | R/W | MCU_STORAGE_WOR_0_D_1 | | This register Can be used to storing 32bits of Data. If Write_protect is set and a value is written to register, then data will not be overwritten |

1320 . MCU_STORAGE_REG1 Register Description

MCU_STORAGE_REG2

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|----------------------|-------------|--|
| 31:0 | R/W | MCU_STORAGE_WORD_0_2 | | This register Can be used to storing 32bits of Data. If Write_protect is set and a value is written to register, then data will not be overwritten |

1321 . MCU_STORAGE_REG2 Register Description

MCU_STORAGE_REG3

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|-----------------------|-------------|--|
| 31:0 | R/W | MCU_STORAGE_WOR_0_D_3 | | This register Can be used to storing 32bits of Data. If Write_protect is set and a value is written to register, then data will not be overwritten |

1322 . MCU_STORAGE_REG3 Register Description

MCU_STORAGE_REG4

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|---------------------|-------------|---|
| 31:0 | R/W | MCU_STORAGE_WORD_40 | | This register Can be used to storing Data |

1323 . MCU_STORAGE_REG4 Register Description

MCU_STORAGE_REG5

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|---------------------|-------------|---|
| 31:0 | R/W | MCU_STORAGE_WORD_50 | | This register Can be used to storing Data |

1324 . MCU_STORAGE_REG5 Register Description

MCU_STORAGE_REG6

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|---------------------|-------------|---|
| 31:0 | R/W | MCU_STORAGE_WORD_60 | | This register Can be used to storing Data |

1325 . MCU_STORAGE_REG6 Register Description

MCU_STORAGE_REG7

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|---------------------|-------------|---|
| 31:0 | R/W | MCU_STORAGE_WORD_70 | | This register Can be used to storing Data |

1326 . MCU_STORAGE_REG7 Register Description

MCU_STORAGE_REG8

| Bit | Access | Function Name | Reset Value | Description |
|------|--------|--------------------|-------------|---|
| 31:0 | R/W | MCU_STORAGE_WORD_8 | 0 | This register Can be used to storing Data |

1327 . MCU_STORAGE_REG8 Register Description

MCU_STORAGE_WRITE_KEY

| Bit | Access | Function Name | Description | Description |
|------|--------|------------------|-------------|---|
| 31:0 | R/W | MCU_STORAGE_KEY0 | | Program the below key to enable access to program MCU storage register 0x91437B2B Program the below key to disable access to program MCU storage register 0xFFFFFFFF |

1328 . MCU_STORAGE_WRITE_KEY Register Description

18.7 Sleep Clock Calibrator

18.7.1 General Description

In this block, the time periods of 32KHz RC clock, 32KHz RO clock and 32KHz XTAL clock can be calibrated. Apart from this there is block to generate periodic triggers to recalibrate time periods for temperature changes and periodic changes. Also there is another block which gives seconds trigger approximately with every 2^{10} clocks of FSM clock.

18.7.2 Features

- 32KHz RC clock time period calibration is done using time period of known XTAL 40MHz clock programmed through APB.
- 32KHz RO calibration uses time period of RC 32KHz clock as reference.

- The frequency of RO calibration is programmable and calibration can happen at a maximum rate of 8 times per second.
- RC calibration can be triggered every 5 secs or every 10 secs or every 15 secs or every 30secs or temperature based, when temperature change is more than the given maximum temperature change acceptable.
- Temperature sensor is triggered to measure the temperature change every 1 sec, 2sec, 4 sec or 5 seconds.
- RTC time period can be made either of clocks time periods based on the clock being used as fsm clk.

18.7.3 Functional Description

RC Timeperiod Calibration

Calendar block provides rc_calibration trigger in regular intervals for time period calibration based on rc_trigger_time_sel.

2'b00 : Every 30 seconds (default)

2'b01 : Every 15 seconds

2'b10 : Every 10 seconds

2'b00 : Every 5 seconds

After every trigger, 40mhz XTAL clock is enabled and waits until the clock gets settled. The settling time is programmable through APB, default value is 7'd64. Time periods used and measured in this block are assumed to have a **granularity of 10ps for the LSB**. Timeperiod of reference clock 13 bits must be programmed in the register before the trigger.

There are 2 counters. One running on RC 32KHz clock and the other on Reference 40mhz clock. Counter 1 counts for $2^{no_of_rc_clocks} = N_{rc}$ (can be 1, 2, 4 or 8). In the mean time counter 2 (28 bit) accumulates the time period of ref clock every posedge, say the counter encounters N_{ref} clock cycles, then the counter 2 value will be $N_{ref} * T_{ref}$.

Total time = $N_{ref} * T_{ref} = N_{rc} * T_{rc_inst}$ where T_{rc_inst} is the instantaneous time period of RC 32KHz clock.

Therefore, $T_{rc_inst} = (N_{ref} * T_{ref}) / N_{rc} == Counter_2_value << no_of_rc_clocks$

The instantaneous time period measured is averaged with the previously measured value.

$T_{RC} = T_{RC_PREV} * (1 - alpha_{rc}) + alpha_{rc} * T_{RC_INST}$

Where $alpha_{rc}$ is a fraction can be 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 programmed through APB.

RO Timeperiod Calibration

RO timeperiod calibration block is triggered very frequently as RO clock is not stable based on the ro_trigger_time_sel .

2'b00 : Every second (default)

2'b01 : Every 1/2 second

2'b10 : Every 1/4 second

2'b11 : Every 1/8 second

RC 32KHz clock acts as reference clock for this calibration. The procedure and theory is similar to that of RC calibration. But as both the clocks are of relatively equal frequencies, higher no.of clock cycles are counted for to notice the difference.

Here the counter 1 counts for $2^{no_of_ro_clocks}$ (can be a 1, 2, 4,....., 2^16, default is 2^10). Counter 2 is 40 bits to account for so much time. With each posedge od 32KHz RC clock counter 1 increments by timeperiod of RC clock. With each posedge of RO clock counter 1 increments by 1 to total N_{ro} .

$T_{ro_inst} = (N_{rc} * T_{rc}) / N_{ro}$

$$T_{ro} = T_{ro_PREV} * (1 - \alpha) + \alpha * T_{ro_inst}$$

Where alpha_ro is a fraction, which can be different from alpha_rc programmed through APB.

Temperature Change Detector

Calendar block triggers temperature sensor at regular intervals of time based on the temp_trigger_time_sel value

2'b00 : Every second (default)

2'b01 : Every 2 seconds

2'b10 : Every 4 seconds

2'b11 : Every 5 seconds

This block keeps a note of temperature at which last calibration of RC time period happened (Temp_prev). After every trigger of temperature sensor It checks for the temperature measurement done signal and reads the temperature value after done.

If this temperature value is beyond the maximum temperature change acceptable, It triggers the RC calibration and copies this value to T_prev. Maximum temperature change acceptable can be changed through APB default (5'd5).

Everytime RC calibration block is triggered, this is also triggered to note the Temp_prev value.

18.7.4 Register Summary

Base Address: 0x2404_8200

| Register Name | Offset | Description |
|---------------------------------------|--------|-------------|
| MCU CAL RO TIMEPERIOD READ | 0x00 | |
| MCU CAL TIMER CLOCK PERIOD | 0x04 | |
| MCU CAL TEMP PROG REG | 0x08 | |
| MCU CAL START REG | 0x0C | |
| MCU CAL REF CLK SETTLE REG | 0x1C | |
| MCU CAL RC TIMEPERIOD READ | 0x14 | |
| MCU CAL REF CLK TIMEPERIOD REG | 0x18 | |

1329 . Base Address: 0x2404_8200

18.7.5 Register Description

MCU CAL RO TIMEPERIOD READ

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|--------------------------|
| 31:25 | | | | |
| 24:0 | R | timeperiod_ro | 0 | Calibrated RO timeperiod |

1330 . MCU CAL RO TIMEPERIOD READ Register Description

MCU CAL TIMER CLOCK PERIOD

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------------------------------|-------------|--|
| 31 | R | spi_rtc_timer_clk_period_applied | 0 | Indicated SOC programmed rtc_timer clock period is applied at KHz clock domain. 1 - Programmed period applied 0 - Programmed period is not applied yet |
| 30:25 | | | | |
| 24:0 | W/R | rtc_timer_clk_period | 0 | RTC timer clock period programmed by SOC MS 8 bit are for Integer part & LS 17bit are for Fractional part Ex: 32Khz clock = 31.25us ==> 31.25*2^17 = 4096000 = 0x3E8000 |

1331 . MCU CAL TIMER CLOCK PERIOD Register Description

MCU CAL TEMP PROG REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------|-------------|---|
| 31:25 | | | | |
| 24 | R/W | rtc_timer_period_mux_sel | 0 | 1- calibrated value is taken 0- SPI value is taken |
| 23 | R/W | periodic_temp_calib_en | 0 | Enable periodic checking of temperature |
| 22:21 | R/W | temp_trigger_time_sel | 0 | 2'd0: Every second 2'd1: every 2 seconds 2'd2: every 4 seconds 2'd3: every 5 seconds |
| 20:16 | R/W | max_temp_change | 5 | maximum temperature change after which rc calibration must be trigger |
| 15:1 | | | | |
| 0 | R/W | bypass_calib_pg | 0 | To bypass power gating and keep all the blocks always on |

1332 . MCU CAL TEMP PROG REG Register Description

MCU CAL START REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|----------|-------------|-------------|
| 31:30 | | | | |

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------------|-------------|--|
| 29:27 | R/W | vbatt_trigger_time_sel | 1 | trigger to ipmu block for checking vbatt status periodically 3'd6 : Every 2 minutes 3'd5: Every minute 3'd4: Every 30 secs 3'd3 : every 15 secs, 3'd2: every 10 secs, 3'd1: every 5 secs 3'd0: every second |
| 26 | R/W | low_power_trigger_sel | 0 | 1 - seperate counter runs based 2^{15} clocks of 32KHz clock = 1sec 0 - calendar runs and triggers are generated based on calendar |
| 25 | R/W | rc_xtal_mux_sel | 0 | 0 - RC clock calibration happens 1 - XTAL 32khz clock timeperiod calibration occurs with reference clock as 40mhz xtal This should not be changed in the middle of process. Must be changed only once. |
| 24 | W | start_calib_rc | 0 | to initiate RC calibration |
| 23 | W | start_calib_ro | 0 | to initiate RO calibration |
| 22 | R/W | periodic_rc_calib_en | 0 | periodically calibrate RC timeperiod based rc trigger time sel |
| 21 | R/W | periodic_ro_calib_en | 0 | periodically calibrate RO timeperiod based ro trigger time sel |
| 20:18 | R/W | rc_trigger_time_sel | 0 | 3'd0 : Every 5secs 3'd1 : every 10 secs, 3'd2: every 15 secs, 3'd3: every 30 secs 3'd4: every minute 3'd5: Every 2 minutes |
| 17:16 | R/W | ro_trigger_time_sel | 0 | 2'd3 : 8 times in a second 2'd2 : 4 times in a second 2'd1 : 2 times in a second 2'd0 : 1 time in a second |
| 15:13 | R/W | rc_settle_time | 5 | no of clocks of RO for the RC clk to settle when enabled |
| 12:10 | R/W | no_of_rc_clks | 3 | $2^{\text{no_of_rc_clocks}}$ = no of rc clocks used in calibration |
| 9:6 | R/W | no_of_ro_clks | 10 | $2^{\text{no_of_ro_clocks}}$ no of clocks of ro clock counts for no of rc clocks in that time to measure timeperiod |
| 5:3 | R/W | alpha_rc | 2 | $\text{alpha} = 1/2^{\text{alpha_rc}}$, averaging factor of RC timeperiod $T = \text{alpha}(t_{\text{inst}}) + (1-\text{alpha})t_{\text{prev}}$ |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|----------|-------------|---|
| 2:0 | R/W | alpha_ro | 2 | alpha = 1/2^alpha_ro , averaging factor of RO timeperiod T = alpha(t_inst) + (1- alpha)t_prev |

1333 . MCU CAL START REG Register Description

MCU CAL REF CLK SETTLE REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------------|-------------|---|
| 31:18 | | | | |
| 17 | R | valid_ro_timeperiod | 0 | Valid signal for reading RO timeperiod |
| 16 | R | valid_rc_timeperiod | 0 | Valid signal for reading RC timeperiod calibrated |
| 15:7 | | | | |
| 6:0 | R/W | xtal_settle | 64 | no of 32khz clocks for xtal 40mhz clk to settle |

1334 . MCU CAL REF CLK SETTLE REG Register Description

MCU CAL RC TIMEPERIOD READ

| Bit | Access | Function | Reset Value | Description |
|-------|--------|---------------|-------------|--------------------------|
| 31:25 | | | | |
| 24:0 | R | timeperiod_rc | 0 | Calibrated RC timeperiod |

1335 . MCU CAL RC TIMEPERIOD READ Register Description

MCU CAL REF CLK TIMEPERIOD REG

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------|-------------|---|
| 31:24 | | | | |
| 23:0 | R/W | timeperiod_ref_clk | 0x33_3333 | timeperiod of reference clk with each bit corresponding to granularity of $2^{27} = 1\text{us}$ ex: 40Mhz Period is 25ns. ($25\text{ns}/1\text{us} = 25\text{e-3}\text{us} = 25\text{e-3} \cdot 2^{27} = 24'd3355443 = 0x33_3333$) ex: 36Mhz Period is 27.778ns. ($27.778\text{ns}/1\text{us} = 27.8\text{e-3}\text{us} = 27.8\text{e-3} \cdot 2^{27} = 24'd3728270 = 0x38_E38E$) |

1336 . MCU CAL REF CLK TIMEPERIOD REG Register Description

18.7.6 Programming Sequence

RC timeperiod Calibration

1. Write the timeperiod of ref clock and settling time for reference clock being used (npss_ref_clk from IPMU) into **MCU CAL REF CLK TIMEPERIOD REG**.
Default values are for xtal_settle(40mhz clock with settling time) in register **MCU CAL REF CLK SETTLE REG** is of 64 clocks of 32khz clock = $64 * 31.25\mu s = 2ms$.
2. In **MCU CAL START REG** program alpha_rc, no_of_rc_clocks based on the requirement. **rc_xtal_mux_sel = 0** and also wirte to **start_calib_rc = 1**;
3. If periodic calibration of RC is required, Enable bit 21, of **MCU CAL START REG**, **periodic_RC_calib_en** , and select **rc_trigger_time_sel** bits based on the rate at with calibration needs to happen.
4. If temperature based calibration has to happen along with periodic calibration, also enable, 23:16 bits of **MCU CAL TEMP PROG REG** based on the description. max_temp_change is the value increase in the temperature after which RC has to be recalibrated.
5. Look for 24th bit in **MCU CAL RC TIMEPERIOD READ** if 1, RC timeperiod is valid, Already the timeperiod calibration has occurred once.Inorder to read the timeperiod of RC, read 23:0 bits of same register.

XTAL 32K timeperiod Calibration

1. Write the timeperiod of ref clock and settling time for reference clock being used (npss_ref_clk from IPMU) into **MCU CAL REF CLK REG**.
Default values are for 40mhz clock with settling time of 64 clocks of 32khz clock = $64 * 31.25\mu s = 2ms$.
2. In **MCU CAL START REG** program alpha_rc, no_of_rc_clocks based on the requirement. **rc_xtal_mux_sel = 1** and also wirte to **start_calib_rc = 1**;
3. If periodic calibration of RC is required, Enable bit 21, of **MCU CAL START REG**, **periodic_RC_calib_en** , and select **rc_trigger_time_sel** bits based on the rate at with calibration needs to happen.
4. If temperature based calibration has to happen along with periodic calibration, also enable, 23:16 bits of **MCU CAL TEMP PROG REG** based on the description. max_temp_change is the value increase in the temperature after which RC has to be recalibrated.
5. Look for 24th bit in **MCU CAL RC TIMEPERIOD READ** if 1, RC timeperiod is valid, Already the timeperiod calibration has occurred once.Inorder to read the timeperiod of RC, read 23:0 bits of same register.

R0 timeperiod Calibration

1. A valid RC timeperiod non zero value must be available to start R0 calibration. Also if the reference clock is 32k xtal clock The 32KHz xtal clock must be enabled manually in IPMU. For rc ref clock - **rc_xtal_mux_sel = 0**, for xtal 32khz ref clock **rc_xtal_mux_sel = 1**.
2. In **MCU CAL START REG** program alpha_ro, no_of_ro_clocks, rc_settle_time based on the requirement. wirte to **start_calib_ro = 1**;
3. If periodic calibration of RO is required, Enable in register **MCU CAL START REG**, **periodic_RO_calib_en** , and select **ro_trigger_time_sel** bits based on the rate at with calibration needs to happen.
4. Look for 24th bit in **MCU CAL RO TIMEPERIOD READ** if 1, RO timeperiod is valid, Already the timeperiod calibration has occurred once. Inorder to read the timeperiod of RO, read 23:0 bits of same register.

18.8 WatchDog Timer (WDT)

18.8.1 General Description

The WatchDog Timer is used generate an interrupt on timeout and a reset in case of system failure which can be caused by an external event like ESD pulse or due to a software failure. Also the Interrupt can be used as a Wakeup source for transitioning from SLEEP/STANDBY to ACTIVE states.

18.8.2 Features

- Independent window watchdog timer.
- Interrupt is generated before the system reset is applied which can be used as a wakeup source.
- Generates reset upon Lockup indication from Processor.
- Configurable low frequency clock. The generation of this clock is described in UULP Vbat Peripheral Clock Architecture section.
 - Low-Frequency RC clock (RC_32KHZ_CLK).
 - Low-Frequency RO clock (RO_32KHZ_CLK).
 - External 32KHz XTAL clock (XTAL_32KHZ_CLK)
- Configurable timeout period.
- Able to operate when CPU is in SLEEP state (as defined in Power Architecture Section) during power-save applications
- APB Interface for accesses from CPU.
- Individually controllable power domain for low-power applications.

18.8.3 Functional Description

WatchDog Timer will generate an Interrupt and Reset at different time instants as configured.

There are two modes defined for the WatchDog Timer.

1. Open Mode: This is a mode during the WatchDog Operation where the Timer restart is allowed from CPU
2. Closed Mode: This is a mode during the WatchDog Operation where the Timer restart is not allowed from CPU.

The Processor needs to restart the Timer upon WDT Interrupt if it the timer is not intended to hit the reset threshold.

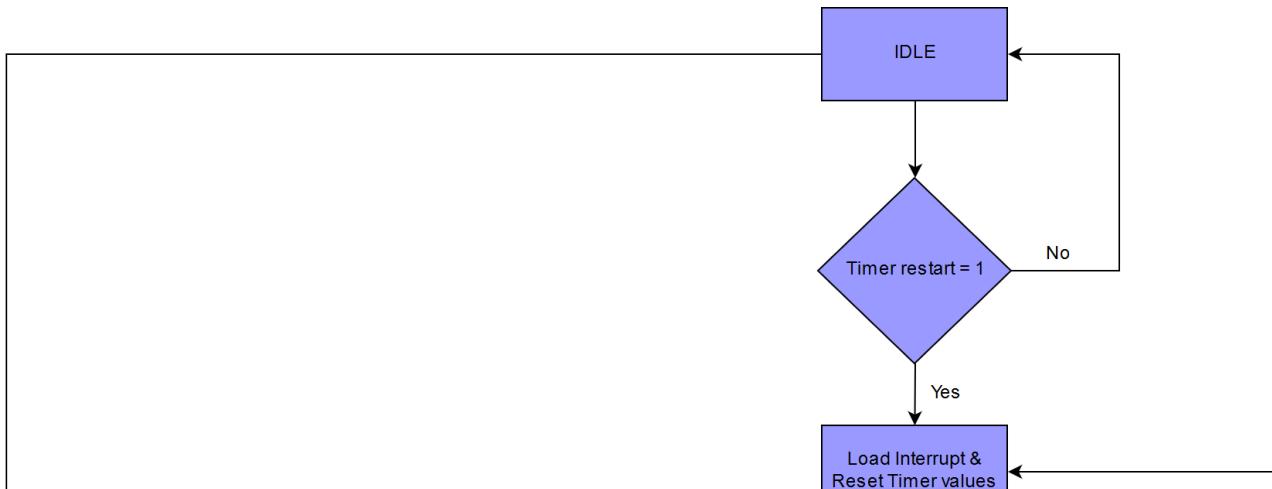
The Timer will be in Closed Mode as defined above till the Interrupt timer is reached. Once the Interrupt timer is reached, it will be in Open Mode till the reset is generated. Also upon Interrupt generation, the timer restarts for the Reset Duration.

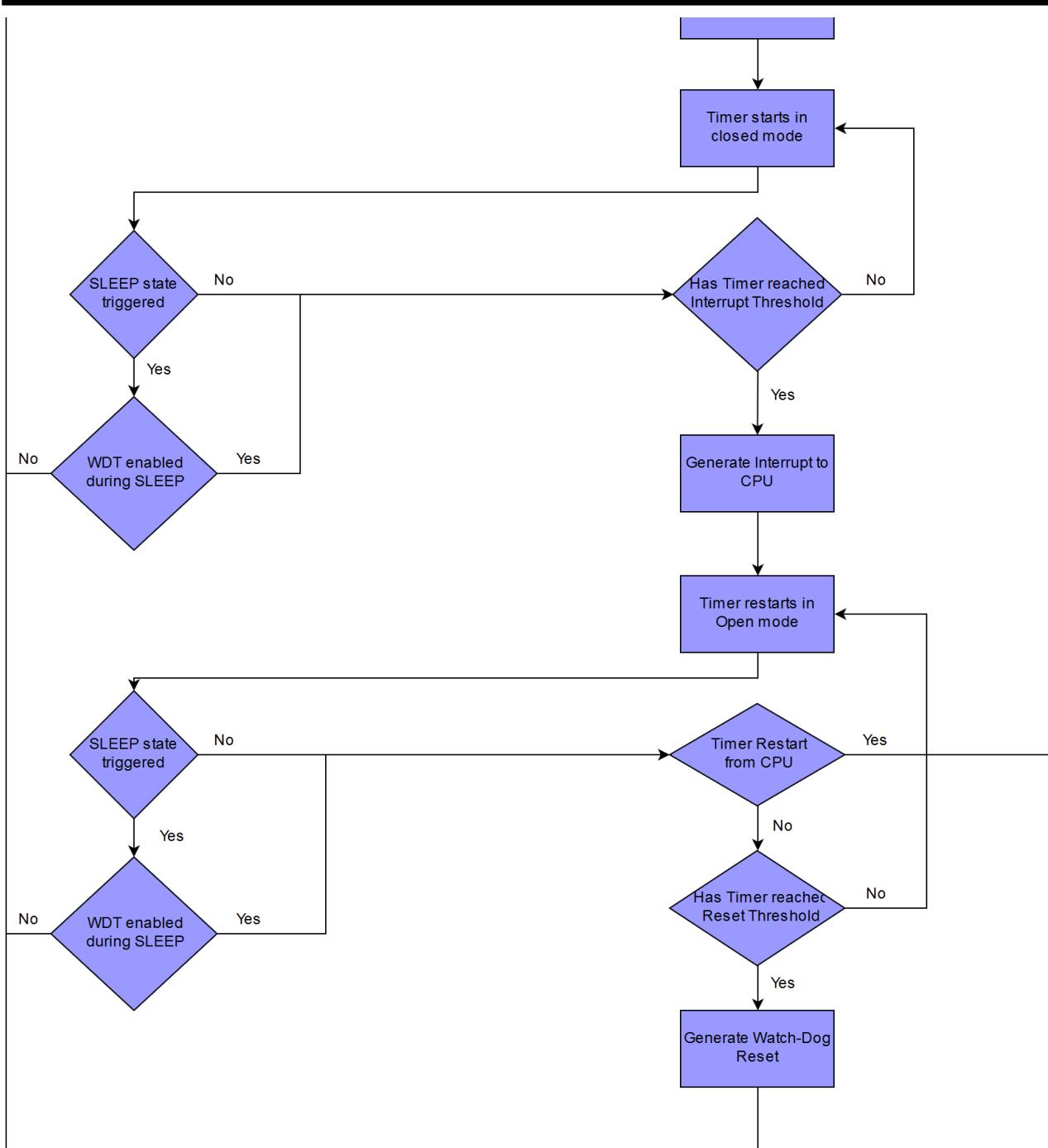
The Timer will be inactive upon reaching SLEEP state (as defined in Power Architecture section) and resets itself upon wakeup to ACTIVE state. However, the Timer can be configured to be running during sleep to avoid system failure during SLEEP state. Also the WDT Interrupt can be used for switching from SLEEP to ACTIVE state.

Also the system reset can be generated upon LockUp (as described in Processor section) indication from the Processor.

State Machine

The figure below depicts the functional flow for the WatchDog Timer





WatchDog Timer Functional Flow Diagram

Programming Sequence

The steps below shows the sequence of programming to be done for the WatchDog functionality

1. Power-Up the WatchDog Domain.
 - a. Refer Power Architecture section for programming details
2. Enable the WatchDog Timer
 - a. This can be configured through "wwd_timer_en" in [WWD_TIMER_ENABLE Register](#).

3. Load the Timer Values for Interrupt and Reset Generation
 - a. system reset duration can be configured through "wwd_system_reset_timer" in [WW SYSTEM RESET TIMER Register](#).
 - b. Interrupt duration can be configured through "wwd_interrupt_timer" in [WW INTERRUPT TIMER Register](#).
4. Start the Timer. The same configuration needs to be done for restarting the Timer in case of interrupt.
 - a. This can be configured through "wwd_timer_rstart" in [WW TIMER ENABLE Register](#).
5. The Timer can be configured to be running during SLEEP state
 - a. Refer [FSM_CTRL_POWER_DOMAINS](#) in Power Architecture section.
6. The WatchDog can be configured to generate system reset upon Lockup indication from Processor.
 - a. This can be configured through "processor_stuck_reset_en" in [WW PROC STUCK EN Register](#).
7. The WDT key needs to be configured to readback the parameters programmed to WDT
 - a. This can be configured through "wwd_key_enable" in [WW KEY ENABLE Register](#)

18.8.4 Register Summary

Base Address: 0x2404_8300

| Register Name | Offset | Description |
|---------------------------------------|--------|--|
| WW INTERRUPT_TIMER | 0x00 | Interrupt Configuration Register |
| WW SYSTEM RESET_TIMER | 0x04 | Reset Configuration Register |
| WW PROC STUCK EN | 0x0C | Reset on Processor Indication Register |
| WW TIMER ENABLE | 0x10 | WDT Enable Register |
| WW KEY ENABLE | 0x18 | WDT Key Register |

[1337 . Register Summary](#)

18.8.5 Register Description

WW INTERRUPT_TIMER

| Bit | Access | Function | Reset Value | Description |
|------|--------|---------------------|-------------|---|
| 31:5 | - | Reserved | - | It is recommended to write these bit to 0 |
| 4:0 | RW | wwd_interrupt_timer | 0 | Indicates the time duration for generation of System Reset This is specified in terms of number of clock(LOW-FREQ clock used for WDT) pulses Number of clock pulses = $2^{(wwd_interrupt_timer)}$ |

[1338 . Watch-Dog Interrupt Timer Register Description](#)

WW SYSTEM RESET_TIMER

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------|-------------|---|
| 31:5 | - | Reserved | - | It is recommended to write these bit to 0 |

| Bit | Access | Function | Reset Value | Description |
|-----|--------|------------------------|-------------|---|
| 4:0 | RW | wwd_system_reset_timer | 0 | <p>Indicates the time duration for generation of System Reset</p> <p>This is specified in terms of number of clock(LOW-FREQ clock used for WDT) pulses</p> <p>Number of clock pulses = $2^{(wwd_system_reset_timer)}$</p> |

1339 . Watch-Dog System Reset Timer Register Description

WW_D PROC_STUCK_EN

| Bit | Access | Function | Reset Value | Description |
|-------|--------|--------------------------|-------------|--|
| 31:17 | - | Reserved | - | It is recommended to write these bit to 0 |
| 16 | W | processor_stuck_reset_en | 0 | <p>Writing 1 to this enables WDT to generate Reset on CPU Lock-Up mode</p> <p>Writing 0 to this disables WDT to generate Reset on CPU Lock-Up mode</p> |
| 15:0 | - | Reserved | - | It is recommended to write these bit to 0 |

1340 . Watch-Dog PROC Struck Register Description

WW_D_TIMER_ENABLE

| Bit | Access | Function | Reset Value | Description |
|-------|--------|------------------|-------------|---|
| 31:24 | - | Reserved | - | It is recommended to write these bit to 0 |
| 23:16 | W | wwd_timer_en | 0 | 0xAA – Enables the WatchDog Timer 0xF0 – Disables the WatchDog Timer |
| 15:1 | - | Reserved | - | It is recommended to write these bit to 0 |
| 0 | W | wwd_timer_rstart | 0 | Writing 1 to this restarts the WatchDog Timer Writing 0 this has no effect |

1341 . Watch-Dog Mode Enable Register Description

WWD_KEY_ENABLE

| Bit | Access | Function | Reset Value | Description |
|------|--------|----------------|-------------|--|
| 31:0 | W | wwd_key_enable | 0x877F38E9 | <p>Specifies the key to read back the WDT Registers described above.</p> <p>Writing 0x877F38E9 to this enables Read Access</p> <p>Writing 0xAAAAAAA to this disables Read Access</p> |

1342 . Watch-Dog Key Enable Register Description

19 Analog Peripherals

| | | | | | |
|-------------|----------------|-----------------------|------|----------|---------------------|
| ULP_GPIO_0 | ADCP0 | TOUCH7 | | comp1_p0 | OPAMP1P2 |
| ULP_GPIO_1 | ADCP10 / ADCN0 | | | comp1_n0 | |
| ULP_GPIO_2 | ADCP1 | | | comp2_p0 | OPAMP1P3 |
| ULP_GPIO_3 | ADCP11 / ADCN1 | TOUCH6 | | comp2_no | |
| ULP_GPIO_4 | ADCP2 | C_int_res_in | DAC0 | comp1_n1 | OPAMP1OUT0 |
| ULP_GPIO_5 | ADCP12 / ADCN2 | res_out | | comp1_p1 | OPAMP2P1 |
| ULP_GPIO_6 | ADCP3 | TOUCH5 | | | OPAMP1P4 |
| ULP_GPIO_7 | ADCP15 / ADCN5 | TOUCH4 | | | OPAMP1P1 / OPAMP1N1 |
| ULP_GPIO_8 | ADCP4 | TOUCH1 / C_int_res_in | | | OPAMP1P5 |
| ULP_GPIO_9 | ADCP14 / ADCN4 | TOUCH2 | | | OPAMP2OUT0 |
| ULP_GPIO_10 | ADCP5 | TOUCH3 / res_out | | | OPAMP3P0 / OPAMP3N0 |
| ULP_GPIO_11 | ADCP13 / ADCN3 | TOUCH8 | | | OPAMP2P0 / OPAMP2N0 |
| ULP_GPIO_12 | | | | comp2_p1 | OPAMP1P0 / OPAMP1N0 |
| ULP_GPIO_13 | | | | comp2_n1 | |
| ULP_GPIO_14 | | | | | OPAMP3P1 |
| ULP_GPIO_15 | | | DAC1 | | OPAMP1OUT1 |

To program ADCP/ADCN on ULP_GPIOs refer to section ADC channel select mode in [Analog to Digital Converter](#)

To program TOUCH,C_int_res_in and res_out on ULP_GPIOs refer to section GPIO Selection in [Touch Capacitance sensor](#)

To program DAC on ULP_GPIOs refer to section AUXDAC OUTPUT MUX in [Digital to Analog Converter](#)

To program COMPA_P, COMPB_P, COMPA_N, COMPB_N on ULP_GPIOs refer to section Input Selection in [Analog Comparators](#)

To program OPAMP-P and OPAMP-N refer to Input Selection in [OPAMPS](#)

19.1 Analog Comparators

19.1.1 General Description

Analog comparators peripheral consists of two analog comparators, a reference buffer, a scaler and a resistor bank.

The comparator compares analog inputs p and n to produce a digital output, cmp_out according to:

$$p > n, \text{cmp_out} = 1$$

$$p < n, \text{cmp_out} = 0$$

19.1.2 Features

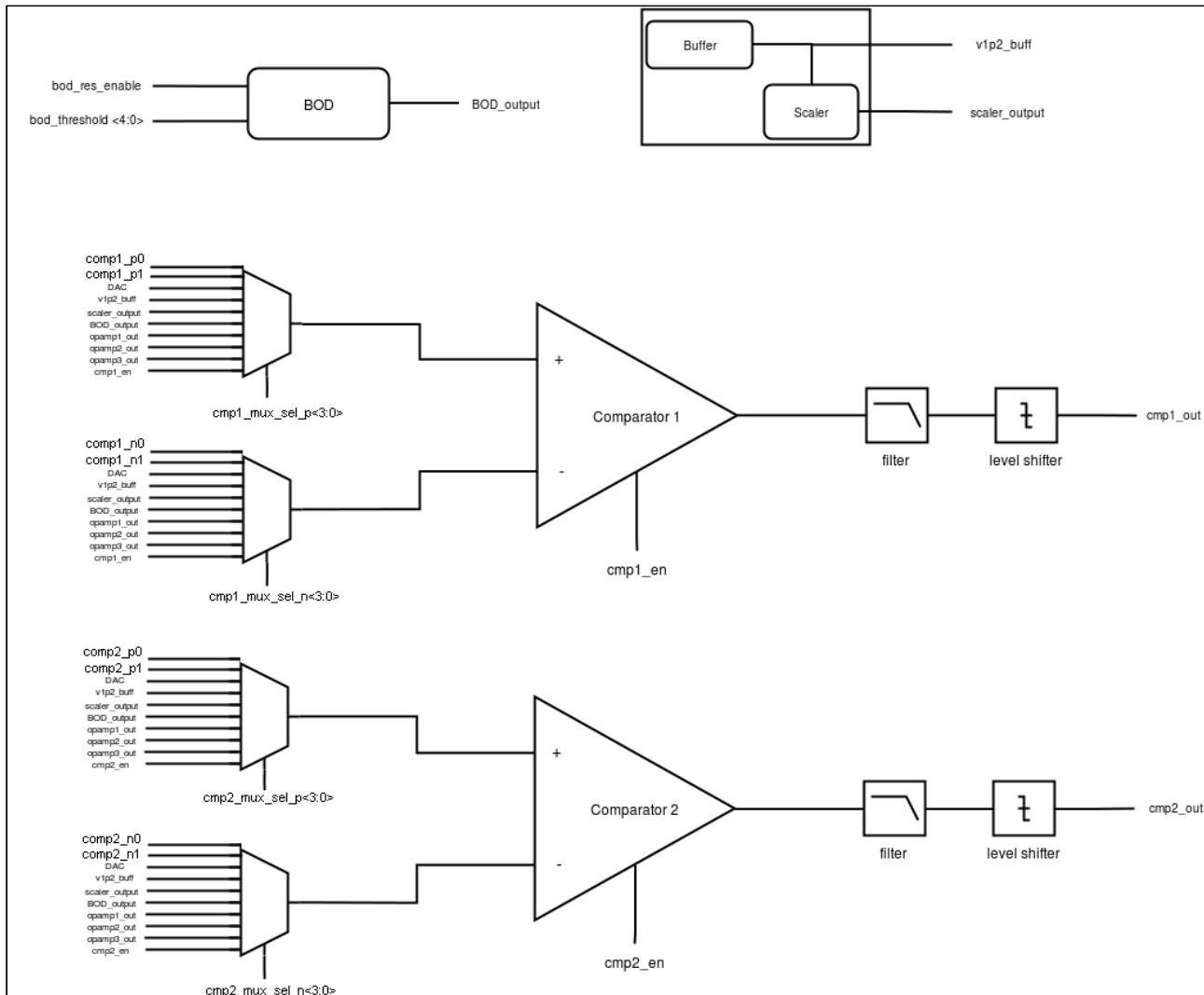
Both comparators can take inputs from GPIOs. AnalogPeripherals-I/OPinMultiplexing

There are 9 different inputs for each pin of comparator, and 2 of the 9 are external pin inputs (GPIOs).
The following cases of comparison are possible

1. Compare external pin inputs
2. Compare external pin input to internal voltages.
3. Compare internal voltages.

The inputs of 2 comparators can be programmed independently. The reference buffer, scaler and resistor bank are shared between the two comparators and can be enabled only when atleast one of the comparators is enabled.

19.1.3 Block Diagram



19.1.4 Functional Description

- ULP_VDD_33, AUX_VBATT_AVDD and MUXED_ULP_SS_VDD supplies should be available for the comparators to work.
- Enable reference buffer and resistor bank if required.
- Select the inputs of comparator using input mux selects and then enable the comparator. By default comparators are disabled.

19.1.5 Input Selection

Each comparator has Vinp mux to select "p" and Vinn mux to select "n" input of comparator.

cmp_p mux

| cmp_mux_se_l_p<3:0> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----------------------------------|----------|----------|----------|----------------------|----------------------|-------------------|------------|------------|------------|
| comparator 1 | comp1_p0 | comp1_p1 | DAC | reference buffer out | reference scaler out | resistor bank out | opamp1_out | opamp2_out | opamp3_out |
| comparator 2 | comp2_p0 | comp2_p1 | DAC | reference buffer out | reference scaler out | resistor bank out | opamp1_out | opamp2_out | opamp3_out |

cmp_n mux

| cmp_mux_sel_n<3:0> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------------------------------|----------|----------|----------|----------------------|----------------------|-------------------|------------|------------|------------|
| comparator 1 | comp1_n0 | comp1_n1 | DAC | reference buffer out | reference scaler out | resistor bank out | opamp1_out | opamp2_out | opamp3_out |
| comparator 2 | comp2_n0 | comp2_n1 | DAC | reference buffer out | reference scaler out | resistor bank out | opamp1_out | opamp2_out | opamp3_out |

19.1.6 Voltage Scaler

The reference buffer uses 1.2V from ULP_BG as its reference. And the scaler takes this buffered 1.2V as its input.

Scaler is configured using REFBUF_VOLT_SEL<3:0>. The output of scaler for different scale factors are in the table below

| Bandgap_scale_factor | Scaler output | Units |
|-----------------------------|----------------------|--------------|
| 0 | 0.1 | V |
| 1 | 0.2 | V |
| 2 | 0.3 | V |
| 3 | 0.4 | V |
| 4 | 0.5 | V |
| 5 | 0.6 | V |
| 6 | 0.7 | V |
| 7 | 0.8 | V |
| 8 | 0.9 | V |
| 9 | 1 | V |
| 10 | 1.1 | V |

19.1.7 Resistor Bank (BOD)

Resbank output = VBATT*(200/(300+bod_threshold*8.33))

| bod threshold | Referred voltage | Unit |
|---------------|------------------|------|
| 0 | 1.8 | V |
| 1 | 1.85 | V |
| 2 | 1.9 | V |
| 3 | 1.95 | V |
| 4 | 2 | V |
| 5 | 2.05 | V |
| 6 | 2.1 | V |
| 7 | 2.15 | V |
| 8 | 2.2 | V |
| 9 | 2.25 | V |
| 10 | 2.3 | V |
| 11 | 2.35 | V |
| 12 | 2.4 | V |
| 13 | 2.45 | V |
| 14 | 2.5 | V |
| 15 | 2.55 | V |
| 16 | 2.6 | V |
| 17 | 2.65 | V |
| 18 | 2.7 | V |
| 19 | 2.75 | V |
| 20 | 2.8 | V |
| 21 | 2.85 | V |
| 22 | 2.9 | V |
| 23 | 2.95 | V |
| 24 | 3 | V |
| 25 | 3.05 | V |
| 26 | 3.1 | V |
| 27 | 3.15 | V |
| 28 | 3.2 | V |

| bod threshold | Referred voltage | Unit |
|---------------|------------------|------|
| 29 | 3.25 | V |
| 30 | 3.3 | V |
| 31 | 3.35 | V |

19.1.8 Register Summary

Base Address: 0x24043800

| Register Name | Offset | Reset Value | Description |
|---------------------|--------|--------------|---|
| Comparator 1 | 0x204 | 0x 0000 0000 | Programs comparators 1&2 |
| BOD | 0x200 | 0x 0000 3E00 | Programs resistor bank, reference buffer and scaler |

[1343 . Register Summary](#)

19.1.9 Register Description

Comparator 1

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------|---------------|--|
| 31:24 | | | | |
| 23:20 | R/W | cmp2_mux_sel_n | 0 | Select for negative input of comparator 2 |
| 19:16 | R/W | cmp2_mux_sel_p | 0 | Select for positive input of comparator 2 |
| 15:14 | R/W | cmp2_hyst | 0 | Programmability to control hysteresis of comparator2 |
| 13 | R/W | cmp2_en_filter | 0 | 1 - To enable filter for comparator 2 |
| 12 | R/W | cmp2_en | 0 | 1 - To enable comparator 2 |
| 11:08 | R/W | cmp1_mux_sel_n | 0 | Select for negative input of comparator 1 |
| 7:04 | R/W | cmp1_mux_sel_p | 0 | Select for positive input of comparator 1 |
| 3:02 | R/W | cmp1_hyst | 0 | Programmability to control hysteresis of comparator1 |
| 1 | R/W | cmp1_en_filter | 0 | 1 - To enable filter for comparator 1 |
| 0 | R/W | cmp1_en | 0 | 1 - To enable comparator 1 |

BOD

| Bit | Access | Function | Default Value | Description |
|-------|--------|------------|---------------|--|
| 31:14 | | | | |
| 13:9 | R/W | BOD_THRSH | 0 | Programmability for resistor bank |
| 8 | R/W | BOD_RES_EN | 0 | Configuration for Resistor Bank 0 – Disable 1 – Enable |

| Bit | Access | Function | Default Value | Description |
|-----|--------|-----------------|---------------|---|
| 7:4 | R/W | REFBUF_VOLT_SEL | 0 | Please refer to Voltage Scalar section |
| 3 | R/W | REFBUF_EN | 0 | Reference Buffer Configuration 0 – Disable 1 – Enable |
| 2:0 | | | | |

1344 . Register Description

19.2 Analog to Digital Converter

19.2.1 General Description

The Analog to Digital Convertor Peripheral (AUXADC) converts analog input to 12 bit digital output.

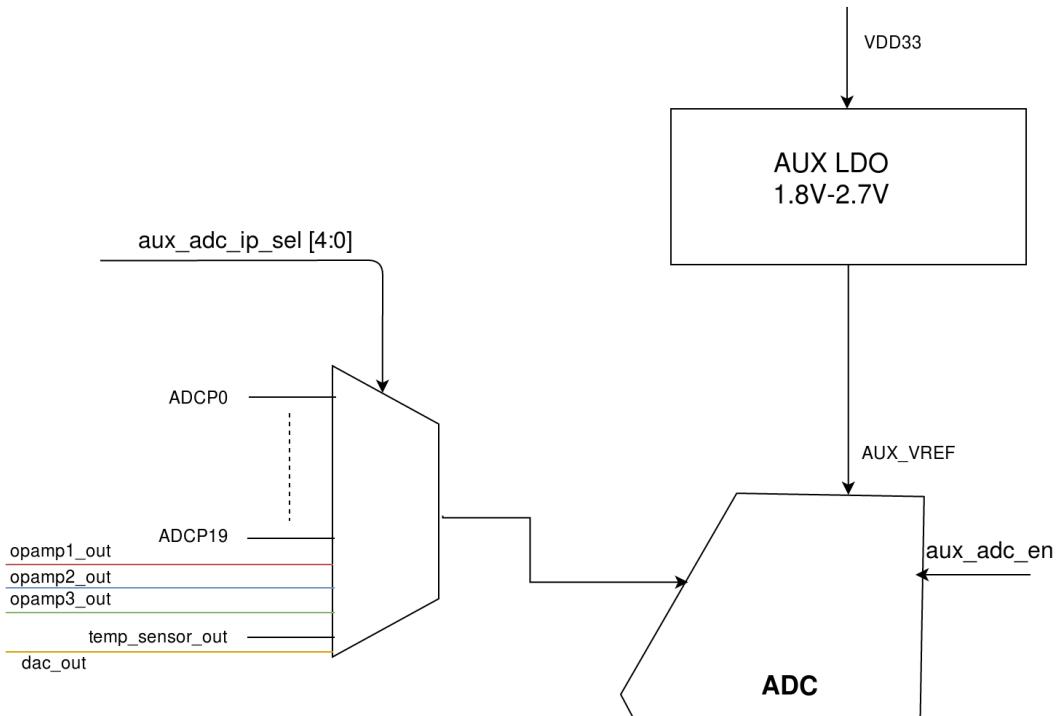
19.2.2 Features

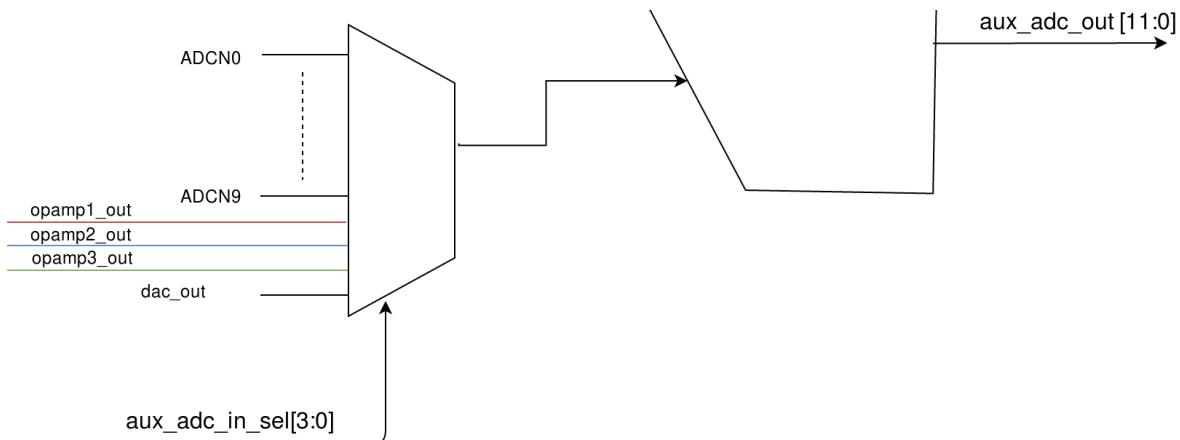
The AUXADC can take analog inputs in single ended or differential mode. The output is 12 bit digital which can be given out with or without noise averaging. The Aux VRef can be connected directly to Vbat (Aux LDO bypass mode) or to the Aux LDO output

The AUXADC has five modes of operation:

- 1)Single ended input with noise averaging,
- 2)Single ended input without noise averaging,
- 3)Differential input with noise averaging,
- 4)Differential input without noise averaging and
- 5)Shutdown mode.

19.2.3 Functional Description





Block Diagram of Analog to Digital Converter

1) Single ended input with noise averaging: Here make sure that following conditions met

- reset_n must be high
- enadc_in must be high
- auxadc_pg_en_b must be 1
- auxadc_bypass_iso_gen must be 0
- auxadc_isolatio_enable must be 0
- clk_in must be there
- differential_in should be 0
- bypass_noisearg must be 0.

Here the analog signal is expected with voltage swing of vdd peak to peak for single ended inputs, supporting common mode voltage of vdd/2. There is no reference supply voltage for the AUXADC .It operates from a supply ranging from 1.8V to 3.6V. The sampling frequency here can be less than or equal to 10MHz for 3.3V operation and 5MHz for 1.8V.

The outputs will come with noise averaged at 2nd posedge of clock with combinational delay in noise averaging. This data is registered at 3rd posedge of output clock.

2) Single ended input without noise averaging: Here make sure that following conditions met

- reset_n must be high
- enadc_in must be high
- auxadc_pg_en_b must be 1
- auxadc_bypass_iso_gen must be 0
- auxadc_isolatio_enable must be 0
- clk_in must be there
- differential_in should be 0
- bypass_noisearg must be 1.

Here the analog signal is expected with voltage swing of vdd peak to peak for single ended inputs, supporting common mode voltage of vdd/2. There is no reference supply voltage for the AUXADC .It operates from a supply ranging from 1.8V to 3.6V. The sampling frequency here can be less than or equal to 10MHz for 3.3V and 5MHz for 1.8V.

Here the outputs will come without noise averaging asynchronously during hold phase of the current sample. This data is registered at 2nd posedge of output clock..

3) Differential input with noise averaging: Here make sure that following conditions met

- reset_n must be high
- enadc_in must be high
- auxadc_pg_en_b must be 1
- auxadc_bypass_iso_gen must be 0
- auxadc_isolatio_enable must be 0
- clk_in must be there
- differential_in should be 1
- bypass_noiseavg must be 0.

Here the analog signal is expected with voltage swing of vdd/2 peak to peak for single ended inputs, supporting common mode voltage of vdd/4. There is no reference supply voltage for the AUXADC .It operates from a supply ranging from 1.8V to 3.6V. The sampling frequency here can be less than or equal to 10MHz for 3.3V and 5MHz for 1.8V.

Here the outputs will come noise averaged at 2nd posedge of clock with combinational delay in noise averaging. This data is registered at 3rd posedge of output clock.

4) Differential input without noise averaging: Here make sure that following conditions met

- reset_n must be high
- enadc_in must be high
- auxadc_pg_en_b must be 1
- auxadc_bypass_iso_gen must be 0
- auxadc_isolatio_enable must be 0
- clk_in must be there
- differential_in should be 1
- bypass_noiseavg must be 1.

Here the analog signal is expected with voltage swing of vdd/2 peak to peak for single ended inputs, supporting common mode voltages of vdd/4. There is no reference supply voltage for the AUXADC .It operates from a supply ranging from 1.8V to 3.6V. The sampling frequency here can be less than or equal to 10MHz for 3.3V and 5MHz for 1.8V.

Here the outputs will come without noise averaging asynchronously during hold phase of the current sample. This data is registered at 2nd posedge of output clock..

5) Shutdown mode: Here make sure that following conditions met

- reset_n must be high
- enadc_in must be low
- auxadc_pg_en_b must be 0
- auxadc_bypass_iso_gen must be 0
- auxadc_isolatio_enable must be 0

19.2.4 ADC channel select mode:

Here we need to select channel using the following register

| aux_adc_ip_sel<4:0> | inp | aux_adc_in_sel<3:0> | inn |
|----------------------------------|------------|----------------------------------|------------|
| 00000 | ULP_GPIO_0 | 0000 | ULP_GPIO_1 |
| 00001 | ULP_GPIO_2 | 0001 | ULP_GPIO_3 |
| 00010 | ULP_GPIO_4 | 0010 | ULP_GPIO_5 |

| aux_adc_ip_sel<4:0> | inp | aux_adc_in_sel<3:0> | inn |
|----------------------------------|-----------------|----------------------------------|-------------|
| 00011 | ULP_GPIO_6 | 0011 | ULP_GPIO_11 |
| 00100 | ULP_GPIO_8 | 0100 | ULP_GPIO_9 |
| 00101 | ULP_GPIO_10 | 0101 | ULP_GPIO_7 |
| 00110 | GPIO_25 | 0110 | GPIO_26 |
| 00111 | GPIO_27 | 0110 | GPIO_28 |
| 01000 | GPIO_29 | 1000 | GPIO_30 |
| 01001 | GPIO_23 | 1001 | GPIO_24 |
| 01010 | ULP_GPIO_1 | 1010 | opamp1_out |
| 01011 | ULP_GPIO_3 | 1011 | opamp2_out |
| 01100 | ULP_GPIO_5 | 1100 | opamp3_out |
| 01101 | ULP_GPIO_11 | 1101 | dac_out_1 |
| 01110 | ULP_GPIO_9 | | |
| 01111 | ULP_GPIO_7 | | |
| 10000 | GPIO_26 | | |
| 10001 | GPIO_28 | | |
| 10010 | GPIO_30 | | |
| 10011 | GPIO_24 | | |
| 10100 | opamp1_out | | |
| 10101 | opamp2_out | | |
| 10110 | opamp3_out | | |
| 10111 | temp_sensor_out | | |
| 11000 | dac_out_1 | | |

Here channel selection can be done by setting [21:17] bits as shown in the following register

| Register | SPI/APB | read/ write | Address | Data | address | Wait time (in us) | set/clear bit informat ion | comment s |
|----------------------------|---------|----------------|----------------------|-------------|---------|-------------------------|-------------------------------------|---|
| ADC_SINGL_E_CH_CTRL | APB | write | 0x24043800 + (113*4) | 0x0000_0000 | | | clear all bits | an_perif_adc_ip_sel = ch_index[21:17] Give adc positive input at AGPIO[0] (default auxadc_ip_sel is 0 which gives AGPIO[0] data to auxadc_ip) an_perif_adc_ip_sel = 20 => auxadc_ip from opamp1 an_perif_adc_ip_sel = 21 => auxadc_ip from opamp2 an_perif_adc_ip_sel = 22 => auxadc_ip from opamp3 an_perif_adc_ip_sel = 23 => auxadc_ip from temp_sensor an_perif_adc_ip_sel = 24 => auxadc_ip from auxdacout |

19.2.5 ADC calibration mode:

ADC comparator and caparray needs to be calibrated for the desired performance. Procedure for the calibration is as follows

Pre requisite to enable AUXADC is

- 1)reset_n must be high
- 2)enadc_in must be high
- 3)auxadc_pg_en_b must be 1
- 4)auxadc_bypass_iso_gen must be 0
- 5)auxadc_isolatio_enable must be 0
- 6)clk_in must be there
- 7)differential_in may be 1 or 0

Note:

This has to be done only for first time bootup

- 1)Calibration enable
- 2)Calib word read back
- 3)Calib write manually

Note:

Here calibration will happen for 32 calib_clk cycles(10 for calib_cmp, 10 for calib_cap_p, 10 for calib_cap_n, 2 clocks in between calibration switching) i.e., 32*4*clk_adc cycles(each calib clk is divided by 4 with clk_in frequency with default values) and also we need to wait for 16 clock cycles for reset_n to sync with calib_clk(32*4+16=144 clk_in cycles).

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informati on | comments |
|------------------------|-------------|----------------|---------|----------|-----------------|----------------------------|-------------------------------------|--|
| POWERGATE REG WRITE | SPI | write | 0x142 | 0x00C800 | 5080C800 | 100 | set bit11, clear bit10,9,8 | Note: It is a spi register auxadc_pg_en_b= 1, auxadc_bypass_is o_gen=0 auxadc_isolatio_en able=0 auxdac_pg_enb=0 |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informati on | comments |
|--------------------------|-------------|----------------|------------------|-------------|-----------------|----------------------------|-------------------------------------|--|
| ULP_DYN_CLK_CTRL_DISABLE | APB | write | 0x24041400 +0xA0 | 0X0006_3800 | | | set bit11,12,13,17,18 | Note: It is APB write Aux mem en=1 Aux clk en=1 aux pclk en=1 udma_clk_enable=1; ir_clk_enable=1 |
| MISC_CONFIG_REG | APB | write | 0x24041400 +0x00 | 0x3007_FFE0 | | | set bit 5 to 18,28,29 | |
| ULP_TA_CLK_GEN_REG | APB | write | 0x24041400 +0x14 | 0x0000_0001 | | | set bit0 clear all other bits | ta ref clk selection |
| ULP_AUX_CLK_GEN | APB | write | 0x24041400 +0x34 | 0x0000_0001 | | | set bit0 clear all other bits | ulp aux clk enable |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informati on | comments |
|--------------------|-------------|----------------|---------------------|-------------|-----------------|----------------------------|-------------------------------------|---|
| ADC_SINGLE_CH_CTRL | APB | write | 0x24043800 +(113*4) | 0x0000_0000 | | | clear all bits | an_perif_adc_ip_sel=ch_index[21:17] Give adc positive input at AGPIO[0] (default auxadc_ip_sel is 0 which gives AGPIO[0] data to auxadc_ip) an_perif_adc_ip_sel=20 => auxadc_ip from opamp1 an_perif_adc_ip_sel=21 => auxadc_ip from opamp2 an_perif_adc_ip_sel=22 => auxadc_ip from opamp3 an_perif_adc_ip_sel=23 => auxadc_ip from temp_sensor an_perif_adc_ip_sel=24 => auxadc_ip from auxdacout |
| ADC_SINGLE_CH_CTRL | APB | write | 0x24043800 +(114*4) | 0x0000_0000 | | | clear all bits | an_perif_adc_in_sel=ch_index[25:22] |
| ADC_SEQ_CTL | APB | write | 0x24043800 +(115*4) | 0x0001_0001 | | | set bit0,16 | To enable disable per channel ping-pong operation (One-hot coding) for channel 1 |
| ADC_INT_ME_M_1 | APB | write | 0x24043800 +(117*4) | 0x24062040 | | | | address 0 -ping memory address for channel 1 |
| ADC_INT_ME_M_2 | APB | write | 0x24043800 +(118*4) | 0x00008020 | | | | address 0- number of values 32+memory write enable 1 |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informati on | comments |
|--------------------|-------------|----------------|---------------------|-------------|-----------------|----------------------------|-------------------------------------|--|
| ADC_INT_ME_M_2 | APPB | write | 0x24043800 +(118*4) | 0x00000020 | | | | address 0 -number of values 32+memory write enable 0 |
| ADC_INT_ME_M_1 | APPB | write | 0x24043800 +(117*4) | 0x24062440 | | | | address 1- pong memory address for channel 1 |
| ADC_INT_ME_M_2 | APPB | write | 0x24043800 +(118*4) | 0x00008420 | | | | address 1- number of values 32+memory write enable 1 |
| ADC_INT_ME_M_2 | APPB | write | 0x24043800 +(118*4) | 0x00000420 | | | | address 1-number of values 32+memory write enable 0 |
| ADC_CH_OFFSET_SET | APPB | write | 0x24043800 +(78*4) | 0x0000_0000 | | | clear all bits | Offset value for this particular channel. This offset value describes the number of clock phases (corresponding to this particular channel) after which this particular channel should be sampled |
| ADC_CH_FREQ_OFFSET | APPB | write | 0x24043800 +(94*4) | 0x0000_0001 | | | set bit0 clear all other bits | Sampling frequency value for this particular channel. This sampling frequency value specifies the frequency of phases (corresponding to this particular channel) on which this particular channel is sampled |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informati on | comments |
|--------------------|-------------|----------------|------------------------|-------------|-----------------|----------------------------|---|--|
| ADC_CTRL_R | APPB | write | 0x24043800 +(1*4) | 0x0800_0C3D | | | set bit0,2,3,4,5,10,11,27 clear all other bits | adcen, adc sw,clear diff |
| ADC clk div | APB | write | 0x24043800 +(3*4) | 0x0000_0004 | | | set bit2 clear all other bits | These bits control the adc clock division factor clock_freq = input_clock_freq/(2*division factor) |
| channel enable reg | APB | write | 0x24043800 +(119*4) | 0x8000_0001 | | | set bit31,0 | internal DMA enable[31] and channel 1 enable |
| AUXADC | SPI | write | 0x24043800 +(512+8) | 0x0000_0400 | | | set bit10 clear all bits | aux_adc_en = 1 aux_adc_dyn_en = 0 |
| AUXADCREG0 | SPI | write | 10'h110 | 0x040A00 | | | set bit11 | cal_en=1. Here cal_en set to 1, here we have to wait for 144 clk_in cycles(144*0.2us=30us for clk_in 5MHz) |
| SPAREREG2 | SPI | write | 10'h1C1 | | | | Check/Poll if Bit[0] is 1 then Check/Poll if Bit[0] is 0 | |
| AUXADCREG2 | SPI | read | 10'h312 | | | | read back data and store it in variable AUXADC_CAL DATA | Calib word read back. Read back calibrated values |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informati on | comments |
|------------------------------------|-------------|----------------|---------------------|----------------------------------|-----------------|----------------------------|--|--|
| AUXADCREG1 | SPI | write | 10'h111 | enter the stored cal data | | | set bit0,7 write stored calibrated values as mentioned in AUXADCREG1[17:13] beside coloumn. | Calib write manually. Manual calibration register programming(AUXADCREG1[12:8]=AUXADC_CALDATA[10:6], AUXADCREG1[7]=1'b1, AUXADCREG1[6:2]=AUXADC_CALDATA[5:0], AUXADCREG1[0]=1'b1 Here output will be available after 3 clocks of clk_in(3*0.2us=0.6us) |
| AUXADC | SPI | write | 0x24043800 +(512+8) | 0x0000_0C00 | | | set bit10,11 | aux_adc_en = 1 aux_adc_dyn_en = 1 |
| <u>INTR_STATU S_REG</u> | APB | read | 0x24043800 +(10*4) | | | | poll for bit7 | |
| Repeat STEPS 9, 10, 11 | APB | write | | | | | | |
| <u>INTR_CLEAR _REG</u> | APB | write | 0x24043820 | 0x00000100 | | | set bit8 clear all bits | Clear first_mem_switch |

19.2.6 Channel selection

| aux_adc_ip_sel<4:0> | inp | aux_adc_in_sel<3:0> | inn |
|---------------------|------------|---------------------|------------|
| 00000 | ULP_GPIO_0 | 0000 | ULP_GPIO_1 |
| 00001 | ULP_GPIO_2 | 0001 | ULP_GPIO_3 |
| 00010 | ULP_GPIO_4 | 0010 | ULP_GPIO_5 |

| aux_adc_ip_sel<4:0> | inp | aux_adc_in_sel<3:0> | inn |
|----------------------------------|-----------------|----------------------------------|-------------|
| 00011 | ULP_GPIO_6 | 0011 | ULP_GPIO_11 |
| 00100 | ULP_GPIO_8 | 0100 | ULP_GPIO_9 |
| 00101 | ULP_GPIO_10 | 0101 | ULP_GPIO_7 |
| 00110 | GPIO_25 | 0110 | GPIO_26 |
| 00111 | GPIO_27 | 0110 | GPIO_28 |
| 01000 | GPIO_29 | 1000 | GPIO_30 |
| 01001 | GPIO_23 | 1001 | GPIO_24 |
| 01010 | ULP_GPIO_1 | 1010 | opamp1_out |
| 01011 | ULP_GPIO_3 | 1011 | opamp2_out |
| 01100 | ULP_GPIO_5 | 1100 | opamp3_out |
| 01101 | ULP_GPIO_11 | 1101 | dac_out_1 |
| 01110 | ULP_GPIO_9 | | |
| 01111 | ULP_GPIO_7 | | |
| 10000 | GPIO_26 | | |
| 10001 | GPIO_28 | | |
| 10010 | GPIO_30 | | |
| 10011 | GPIO_24 | | |
| 10100 | opamp1_out | | |
| 10101 | opamp2_out | | |
| 10110 | opamp3_out | | |
| 10111 | temp_sensor_out | | |
| 11000 | dac_out_1 | | |

Here channel selection can be done by setting [21:17] bits as shown in the following register

| Register | SPI/APB | read/ write | Address | Data | address | Wait time (in us) | set/clear bit informat ion | comment s |
|----------------------------|---------|----------------|---------------------|-------------|---------|-------------------------|-------------------------------------|---|
| ADC_SINGL_E_CH_CTRL | APB | write | 0x24043800 +(113*4) | 0x0000_0000 | | | clear all bits | an_perif_adc_ip_sel =ch_index[21:17] Give adc positive input at AGPIO[0] (default auxadc_ip_sel is 0 which gives AGPIO[0] data to auxadc_ip) an_perif_adc_ip_sel =20 => auxadc_ip from opamp1 an_perif_adc_ip_sel =21 => auxadc_ip from opamp2 an_perif_adc_ip_sel =22 => auxadc_ip from opamp3 an_perif_adc_ip_sel =23 => auxadc_ip from temp_sensor an_perif_adc_ip_sel =24 => auxadc_ip from auxdacout |

Note:

We need to write following registers whenever we are coming back from deep sleep to wakeup mode. This enables manual calibration write

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informa tion | comments |
|------------------------|-------------|----------------|---------|--|-----------------|----------------------------|--|--|
| POWERGATE REG WRITE | SPI | write | 0x142 | 0x00C800 | 5080C800 | 100 | set bit11, clear bit10,9,8 auxadc_pg_en_b =1, auxadc_bypass_i so_gen=0 auxadc_isolatio_e nable=0 auxdac_pg_enb=0 | Note: It is a spi register auxadc_pg_en_b =1, auxadc_bypass_i so_gen=0 auxadc_isolatio_e nable=0 auxdac_pg_enb=0 |
| AUXADCREG 1 | SPI | write | 10'h111 | enter the stored cal data | | | set bit0,7 write stored calibrated values as mentioned in beside coloumn. | Calib write manually. Manual calibration register programming(AUXADCREG1[17:1 3]=AUXADCREG2[15:11], AUXADCREG1[12:8]=AUXADCREG2[1 0:6], AUXADCREG1[7]= 1'b1, AUXADCREG1[6:2]=AUXADCREG2[5:0], AUXADCREG1[0]= 1'b1) Here output will be available after 3 clocks of clk_in(3*0.2us=0.6 us) |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informatio n | comments |
|----------------------|-------------|----------------|---------------------|-------------|-----------------|----------------------------|-------------------------------------|---|
| ULP_DYN_CLK_CTRL_REG | APPB | write | 0x24041400 +0xA0 | 0X0006_3800 | | | set bit11,12,13,17,18 | Note: It is APB write Aux mem en=1 Aux clk en=1 aux pclk en=1 udma_clk_enable =1; ir_clk_enable=1 |
| MISC_CONFIG_REG | APPB | write | 0x24041400 +0x00 | 0x3007_FFE0 | | | set bit 5 to 18,28,29 | |
| ULP_TA_CLK_GEN_REG | APPB | write | 0x24041400 +0x14 | 0x0000_0001 | | | set bit0 clear all other bits | ta ref clk selection |
| ULP_AUX_CLK_GEN | APPB | write | 0x24041400 +0x34 | 0x0000_0001 | | | set bit0 clear all other bits | ulp aux clk enable |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informatio n | comments |
|--------------------|-------------|----------------|---------------------|-------------|-----------------|----------------------------|-------------------------------------|---|
| ADC_SINGLE_CH_CTRL | APPB | write | 0x24043800 +(113*4) | 0x0000_0000 | | | clear all bits | an_perif_adc_ip_sel =ch_index[21:17] Give adc positive input at AGPIO[0] (default auxadc_ip_sel is 0 which gives AGPIO[0] data to auxadc_ip) an_perif_adc_ip_sel =20 => auxadc_ip from opamp1 an_perif_adc_ip_sel =21 => auxadc_ip from opamp2 an_perif_adc_ip_sel =22 => auxadc_ip from opamp3 an_perif_adc_ip_sel =23 => auxadc_ip from temp_sensor an_perif_adc_ip_sel =24 => auxadc_ip from auxdacout |
| ADC_SINGLE_CH_CTRL | APPB | write | 0x24043800 +(114*4) | 0x0000_0000 | | | clear all bits | an_perif_adc_in_sel =ch_index[25:22] |
| ADC_SEQ_CTL | APPB | write | 0x24043800 +(115*4) | 0x0001_0001 | | | set bit0,16 | To enable disable per channel ping-pong operation (One-hot coding) for channel 1 |
| ADC_INT_ME_M_1 | APPB | write | 0x24043800 +(117*4) | 0x24062040 | | | | address 0 -ping memory address for channel 1 |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informatio n | comments |
|-----------------------|-------------|----------------|------------------------|-------------|-----------------|----------------------------|-------------------------------------|--|
| ADC_INT_ME APB M_2 | | write | 0x24043800 +(118*4) | 0x00008020 | | | | address 0- number of values 32+memory write enable 1 |
| ADC_INT_ME APB M_2 | | write | 0x24043800 +(118*4) | 0x00000020 | | | | address 0 - number of values 32+memory write enable 0 |
| ADC_INT_ME APB M_1 | | write | 0x24043800 +(117*4) | 0x24062440 | | | | address 1- pong memory address for channel 1 |
| ADC_INT_ME APB M_2 | | write | 0x24043800 +(118*4) | 0x00008420 | | | | address 1- number of values 32+memory write enable 1 |
| ADC_INT_ME APB M_2 | | write | 0x24043800 +(118*4) | 0x00000420 | | | | address 1-number of values 32+memory write enable 0 |
| ADC_CH_OF FSET | APB | write | 0x24043800 +(78*4) | 0x0000_0000 | | | clear all bits | Offset value for this particular channel. This offset value describes the number of clock phases (corresponding to this particular channel) after which this particular channel should be sampled |

| Register | SPI/ APB | read/ write | Address | Data | 32'h address | Wait time (in us) | set/clear bit informatio n | comments |
|-------------------------------|-------------|----------------|------------------------|-------------|-----------------|----------------------------|---|--|
| ADC_CH_FREAPPB Q_OFFSET | | write | 0x24043800 +(94*4) | 0x0000_0001 | | | set bit0 clear all other bits | Sampling frequency value for this particular channel. This sampling frequency value specifies the frequency of phases (corresponding to this particular channel) on which this particular channel is sampled |
| ADC_CTRL_R EG | APB | write | 0x24043800 +(1*4) | 0x0800_0C3D | | | set bit0,2,3,4,5,10, 11,27 clear all other bits | adcen, adc sw,clear diff |
| ADC clk div | APB | write | 0x24043800 +(3*4) | 0x0000_0004 | | | set bit2 clear all other bits | These bits control the adc clock division factor clock_freq = input_clock_freq/ (2*division factor) |
| channel enable reg | APB | write | 0x24043800 +(119*4) | 0x8000_0001 | | | set bit31,0 | internal DMA enable[31] and channel 1 enable |
| AUXADC | SPI | write | 0x24043800 +(512+8) | 0x0000_0C00 | | | set bit10,11 | aux_adc_en = 1 aux_adc_dyn_en = 1 |
| <u>INTR_STATUS_REG</u> | APB | read | 0x24043800 +(10*4) | | | | poll for bit7 | |
| Repeat STEPS 10, 11, 12 | APB | write | | | | | | |
| <u>INTR_CLEAR_REG</u> | APB | write | 0x24043820 | 0x00000100 | | | set bit8 clear all bits | Clear first_mem_switch |

Note:

After writing calibration bits manually, we need to wait for 3 clk_in cycles to get the digital_op<11:0>

If POC is 0 and all power supplies vddulp, dvdd, avdd, vref are there then AUXADC will give the output bits as
vip=[0 vref]; %input voltage ranging from [0 vref]

vin=((78+8*shift_gain)/(143+8*shift_gain))*avdd; %adjacent value in single ended mode and vin ranging from [0 avdd/2]in differential mode

vdiff=vip-vin;

vref=(64/(89+8*gain))*avdd;

%% Modelled output of adc is given by the following equation

ADCOUT=2047.5+(2047.5*vdiff/vref); % this is in binary offset mode

ADCOUT=(2047.5*vdiff/vref); % this is in 2's complement mode, which we are using here.

19.2.7 Register Summary

| Address | Register Name | Register Description |
|----------|---------------|--|
| 10'h 110 | AUXADCREG0 | AUXADC config register |
| 10'h 111 | AUXADCREG1 | Manual mode settings for comparator and caparray calibration |
| 10'h 112 | AUXADCREG2 | Comparator and caparray calib read back |

Register Description

AUXADCREG0

| Address: 10'h 110 | | | | |
|-----------------------------|--------|-----------------|---------------|---|
| Bit | Access | Function | Default Value | Description |
| [21:19] | R/W | reserved | 0 | Reserved bits |
| 18 | R/W | sel_pin_en | 1 | 0 – Enable is given from SPI 1 – Enable is controlled from SOC |
| 17 | R/W | bypass_noiseavg | 0 | 1 to bypass noise average 0 to enable noise averaging |
| 16 | R/W | enadc_r | 0 | 1 to adc enable from register 0 to enable adc from pin |

| Address: 10'h 110 | | | | |
|------------------------------|---------------|-----------------|----------------------|---|
| Bit | Access | Function | Default Value | Description |
| [15:14] | R/W | adc_clk_sel | 0 | auxadc clock select 00 => clk_out=clk_in 01 => clk_out=clk_in/2 10 => clk_out=clk_in/4 11 => clk_out=clk_in/8 |
| [13:12] | R/W | delay_sel | 0 | to select sam_en_n on time in single ended mode 00 => sam_en_n_ontime=tdelay of buffer 01 => sam_en_n_ontime=2*tdelay of buffer 10 => sam_en_n_ontime=3*tdelay of buffer 11 => sam_en_n_ontime=4*tdelay of buffer |
| 11 | R/W | cal_en | 0 | 0 to disable calibration at bootup 1 to enable calibration at bootup |
| [10:9] | R/W | clk_div_sel | 1 | to clock division select of calibration clock 00 => clk_calib=clk_in/2 01 => clk_calib=clk_in/4 10 => clk_calib=clk_in/8 11 => clk_calib=clk_in/16 |

| Address: 10'h 110 | | | | |
|------------------------------------|---------------|-------------------------|----------------------|---|
| Bit | Access | Function | Default Value | Description |
| [8:6] | R/W | calib_clk_delay_se l | 0 | <p>to select delay between calib_clk_cmp and calib_clk_caparr</p> <pre> 3'd0: calib_clk_caparr = clk_calib_dly[24]; // 5.25n delay 3'd1: calib_clk_caparr = clk_calib_dly[29]; // 6.3n 3'd2: calib_clk_caparr = clk_calib_dly[49]; // 10.5n 3'd3: calib_clk_caparr = clk_calib_dly[57]; // 11.8n 3'd4: calib_clk_caparr = clk_calib_dly[109]; // 23.5n 3'd5: calib_clk_caparr = clk_calib_dly[115]; // 24.7n 3'd6: calib_clk_caparr = clk_calib_dly[224]; // 47n 3'd7: calib_clk_caparr = clk_calib_dly[234]; // 49.9n </pre> |
| [5:4] | R/W | capcal_avg_dura tion | 0 | <p>avg duration between caparr calibration</p> <pre> case(capcal_avg_duration) 2'd0: average_length = 4'd0; // 1 -clocks 2'd1: average_length = 4'd1; // 2 -clocks 2'd2: average_length = 4'd3; // 4 -clocks 2'd3: average_length = 4'd7; // 8 -clocks endcase </pre> |
| [3:2] | R/W | gain | 0 | to control gain error of auxadc |
| [1:0] | R/W | shift_gain | 0 | to control offset error of auxadc |

AUXADCREG1

| Address: 10'h 111 | | | | |
|--------------------------|---------------|-----------------|----------------------|--------------------|
| Bit | Access | Function | Default Value | Description |
| [21:19] | R/W | reserved | 0 | Reserved bits |

| Address: 10'h 111 | | | | |
|--------------------------|---------------|-------------------|----------------------|--|
| Bit | Access | Function | Default Value | Description |
| 18 | R/W | manual_clk_select | 0 | 1 to select caparray clock from clk_in 0 to select clock from internal one shot generator |
| [17:13] | R/W | manual_cap_pbits | 0 | manual caparray calibration bits |
| [12:8] | R/W | manual_cap_nbits | 0 | manual caparray calibration bits |
| 7 | R/W | manual_cap_cal_en | 0 | manual caparray calibration select |
| [6:2] | R/W | manual_cmp_bits | 0 | manual comparatot calibration bits |
| 1 | -- | Reserved | 0 | Reserved |
| 0 | R/W | manual_cmp_cal_en | 0 | manual comparatot calibration enable |

AUXADCREG2

| Address: 10'h 112 | | | | |
|--------------------------|---------------|-----------------|----------------------|------------------------------------|
| Bit | Access | Function | Default Value | Description |
| [21:16] | R | reserved | 0 | reserved |
| [15:11] | R | caparr_calib_p | 0 | manual p-caparray calibration bits |
| [10:6] | R | caparr_calib_n | 0 | manual n-caparray calibration bits |
| [5:0] | R | cmp_calib | 0 | manual caparray calibration select |

19.3 AUX_LDO

19.3.1 General Description

This LDO gives **1.6V to 2.8V**(steps of 80mV) Output voltage with a maximum load current of **25mA** with a dropout voltage of **300mV**. It is external compensated LDO which is stable for load current ranging from 0 to 25mA, with the load cap 1uF.

The LDO reference voltage is 1.2V.

AUX_VBATT_AVDD is the output of this ldo. Analog Peripherals like opamps, comparators, DAC and ADC will work on AUX_VBATT_AVDD.

This ldo can be disabled and AUX_VBATT_AVDD can be connected to an external supply for Ultra-low-power direct battery peripheral operation to save the LDO quiescent current.

19.3.2 Functional Description

- ULP_VDD_33 and MUXED_ULP_SS_VDD supplies should be available for the ldo to work.
- It uses 1.2V from ULP_BG as reference.
- Select the control depending on required output voltage. By default this ldo is enabled.
- Make BYPASS_LDO high to bypass ldo, if required. In bypass mode, output is equal to ULP_VDD_33.

19.3.3 Output Programming

$$V_{out} = v_{ref} * (4/3 + ctrl/15) = 1.6 + 0.08 * ctrl$$

| LDO_CTRL<3:0> | Vout (V) |
|---------------|----------|
| 0 | 1.6 |
| 1 | 1.68 |
| 2 | 1.76 |
| 3 | 1.84 |
| 4 | 1.92 |
| 5 | 2 |
| 6 | 2.08 |
| 7 | 2.16 |
| 8 | 2.24 |
| 9 | 2.32 |
| 10 | 2.4 |
| 11 | 2.48 |
| 12 | 2.56 |
| 13 | 2.64 |
| 14 | 2.72 |
| 15 | 2.8 |

19.3.4 Register Summary

Base Address: 0x24043800

| Register Name | Offset | Reset Value | Description |
|---------------|--------|--------------|-------------------------------|
| LDO | 0x210 | 0x 0000 0053 | Programs LDO CTRL, ENABLE etc |

1345 . Register Summary

19.3.5 Register Description

LDO

| Bit | Access | Function | Default Value | Description | Dynamic controllable |
|-----|--------|------------------|---------------|--|----------------------|
| 3:0 | R/W | LDO_Ctrl | 3 | Word to set the output voltage | No |
| 4 | R/W | LDO_DEFAULT_MODE | 1 | 0 : normal mode 1 : default mode (1.8V) | No |
| 5 | R/W | BYPASS_LDO | 0 | 1 - To enable bypass mode | Yes |
| 6 | R/W | ENABLE | 1 | 1 - To turn on ldo | Yes |
| 7 | R/W | Dyn_en | 0 | Dynamic Enable | No |

1346 . Register Description

19.4 Digital to Analog Converter

19.4.1 General Description

The AUXDAC takes 10 bit inputs and generates corresponding buffered analog voltage output.

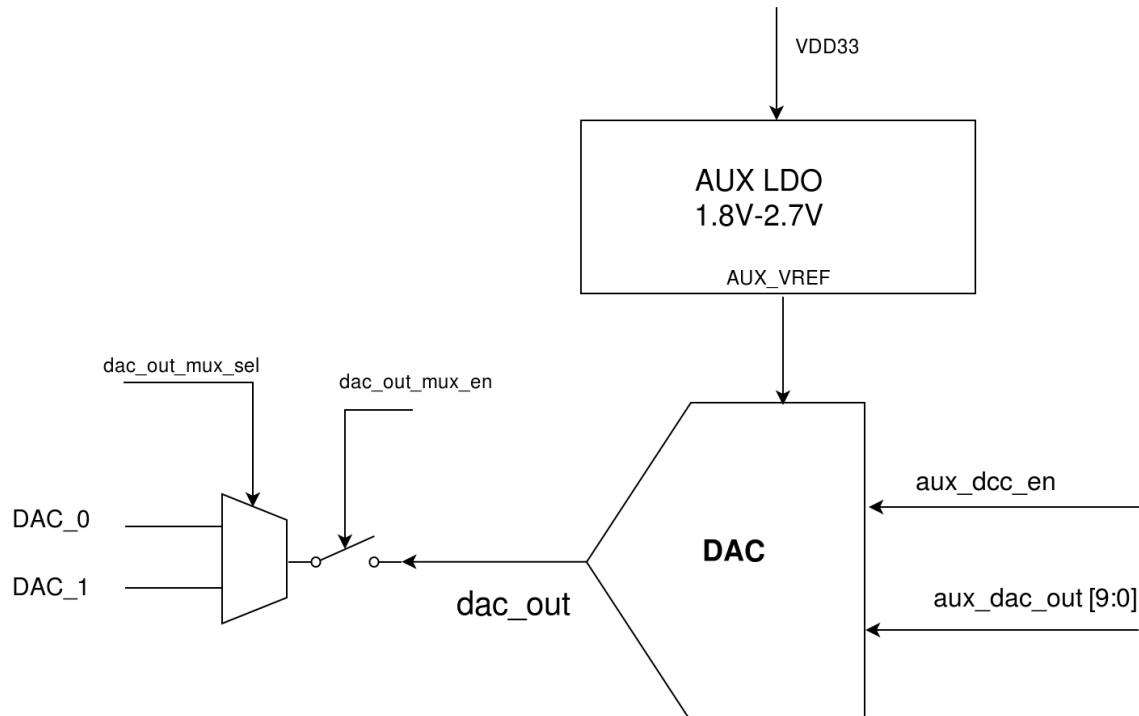
19.4.2 Features

The AUXDAC can take 10 bit digital inputs and convert them into analog voltage within range $5*vdd/36$ to $31*vdd/36$. Vdd can vary from 1.8 volts to 3.6 volts.

The output of the Aux DAC can be multiplexed onto one of two pins.

The AUXDAC has two modes: Operational mode and Shutdown mode.

19.4.3 Functional Description



Block Diagram of Digital to Analog Converter

Operation mode: For DAC operation the following conditions should be met

- vdd, vref, and avdd pins should have stable voltages and reset_n should be high .
- Pins endac_in and auxdac_pg_en_b need to be high
- Clock provided.

Shutdown mode: For shutdown:

- endac_in and auxdac_pg_en_b pins need to be low.

19.4.4 Register Summary

| Address | Register Name | Description |
|---------|---------------|---------------------------|
| 11A | AUXDACREG0 | To enable AUXDAC and rrbs |

19.4.5 Register Description

AUXDACREG0

| 10'h11A | | | | |
|----------------|---------------|------------------|----------------|---|
| Bit | Access | Function | Default | Description |
| [13:6] | R/W | preset_rrbs<7:0> | 8'd170 | 7:4 for ulsb rotation bits programming 3:0 for msb rotation bits programming |
| 5 | R/W | prbs_start | 1 | 1 to select rrbs rotation 0 to disable rrbs rotation |
| 4:2 | -- | Reserved | 0 | Reserved |
| 1 | R/W | sel_pin_eni | 1 | 1 to sel pin enable 0 to select from register |
| 0 | R/W | endaci_r | 0 | 1 to enable dac 0 to disable dac |

19.4.6 AUXDAC OUTPUT MUX

| | | |
|---------------------------|------|------|
| auxdac_out_mux_sel | 0 | 1 |
| GPIO | DAC0 | DAC1 |

19.5 OPAMPS

19.5.1 General Description

The opamps top consists of 3 general purpose Operational Amplifiers (OPAMP) offering rail-to-rail inputs and outputs.

Each of the three opamps has 2 inputs (inp, inn) and 1 output.

19.5.2 Features

All the opamps can take inputs from GPIOs and their outputs can be seen on GPIOs. For details refer to the Pin multiplexing. AnalogPeripherals-I/OPinMultiplexing

An opamp can be configured in either low power mode or high power mode. Make lp_mode bit high to enable low power mode.

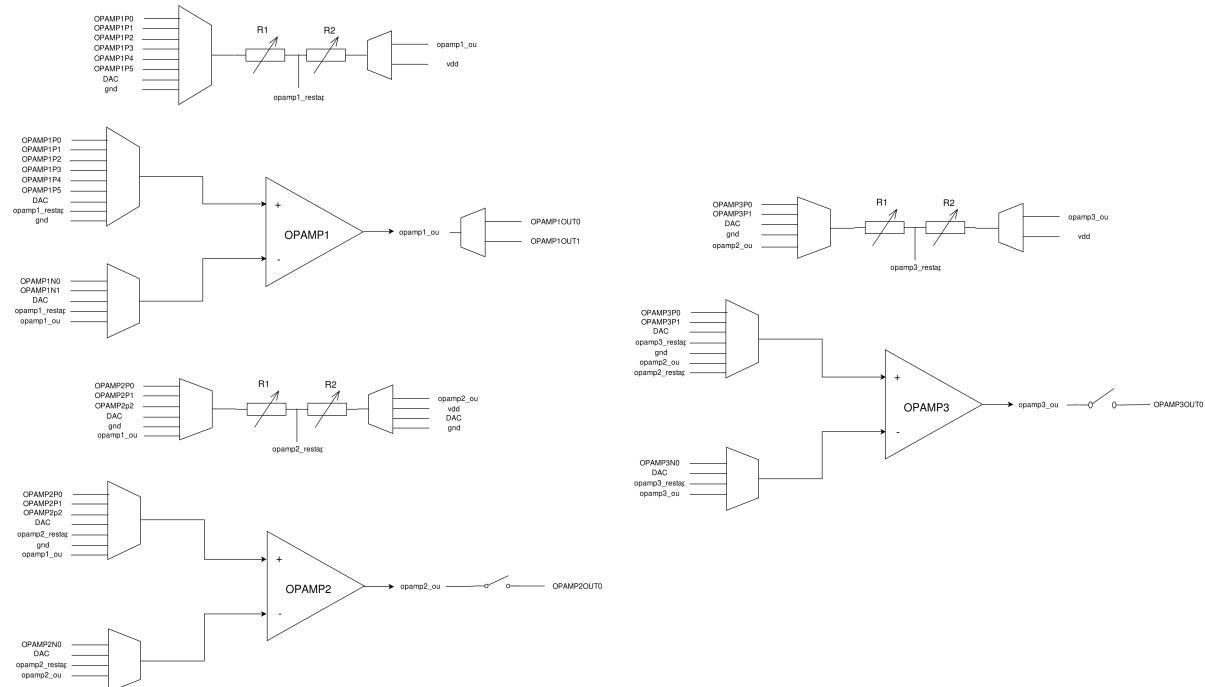
The opamps top consists of 3 general purpose Operational Amplifiers (OPAMP) offering rail-to-rail inputs and outputs.

The opamps can be configured as:

1. Unity gain amplifier
2. Trans-Impedance Amplifier (TIA)
3. Non-inverting Programmable Gain Amplifier (PGA)
4. Inverting Programmable Gain Amplifier (PGA)

5. Non-inverting Programmable hysteresis comparator
6. Inverting Programmable hysteresis comparator
7. Cascaded Non-Inverting PGA
8. Cascaded Inverting PGA
9. Two opamps Differential Amplifier
10. Instrumentation Amplifier
 - 7,8,9 are configured by cascading 2 opamps.
 - 10 is configured by cascading 3 opamps.

19.5.3 Block Diagram



19.5.4 Functional Description

Following is needed for OpAmp operation:

- ULP_VDD_33, AUX_VBATT_AVDD and MUXED_ULP_SS_VDD supplies should be available.
- Configured power mode (normal mode / low power mode using lp_mode bit).
- Configure the opamp into one of the possible configurations (as described earlier) and then enable the opamp. By default opamps are disabled.
- Opamp's gain ($R2/R1$ or $1+R2/R1$) can be configured using Resistor Banks
- To see an opamp's output on its respective GPIO, make out_mux_en high

19.5.5 Input Selection

Every opamp will have Vinp mux to select "inp", Vinn mux to select "inn" and Resistor mux for feedback.

Vinp mux selection

| inp_sel | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---------|----------|----------|-----------|----------|----------|----------|---------------|---------|-----|
| Opamp 1 | OPAMP1P0 | OPAMP1P1 | OPAMP1P2 | OPAMP1P3 | OPAMP1P4 | OPAMP1P5 | DAC | res_tap | gnd |
| Opamp 2 | OPAMP2P0 | OPAMP2P1 | opamp2_p2 | DAC | res_tap | gnd | OPAMP1_o | --ut | -- |
| Opamp 3 | OPAMP3P0 | OPAMP3P1 | DAC | res_tap | gnd | OPAMP2_o | OPAMP2_re--ut | stop | -- |

Vinn mux selection

| inn_sel | 0 | 1 | 2 | 3 | 4 |
|---------|----------|----------|---------|---------|-----|
| Opamp1 | OPAMP1N0 | OPAMP1N1 | DAC | res_tap | out |
| Opamp2 | OPAMP2N0 | DAC | res_tap | out | -- |
| Opamp3 | OPAMP3N0 | DAC | res_tap | out | -- |

Resistor mux selection

| res_mux_sel | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-------------|----------|----------|-----------|----------|--------------|--------------|-----|-----|
| Opamp1 | OPAMP1P0 | OPAMP1P1 | OPAMP1P2 | OPAMP1P3 | OPAMP1P4 | OPAMP1P5 | DAC | gnd |
| Opamp2 | OPAMP2P0 | OPAMP2P1 | opamp2_p2 | DAC | gnd | OPAMP1_out-- | -- | -- |
| Opamp3 | OPAMP3P0 | OPAMP3P1 | DAC | gnd | OPAMP2_out-- | -- | -- | -- |

19.5.6 Configuring the Opamps

Each Opamp can be configured by its respective controls.

- Select the positive input using inp_sel
- Select the negative input using inn_sel
- Select en_res_bank if resistor bank is used.
- Select R1 and R2 using R1_sel and R2_sel respectively.

- Use res_to_out_vdd to connect resistor bank either to output or vdd.
- Select lp_mode to enable low power mode

There will be more than one option for some controls, and they have to be programmed depending on the input that need to be selected.

19.5.7 Standalone mode

Each Opamp can be used as standalone amplifier. In this mode, positive input, negative input and the output are routed from/to external I/Os and uses external feedback. Opamps can also be cascaded to support some configurations.

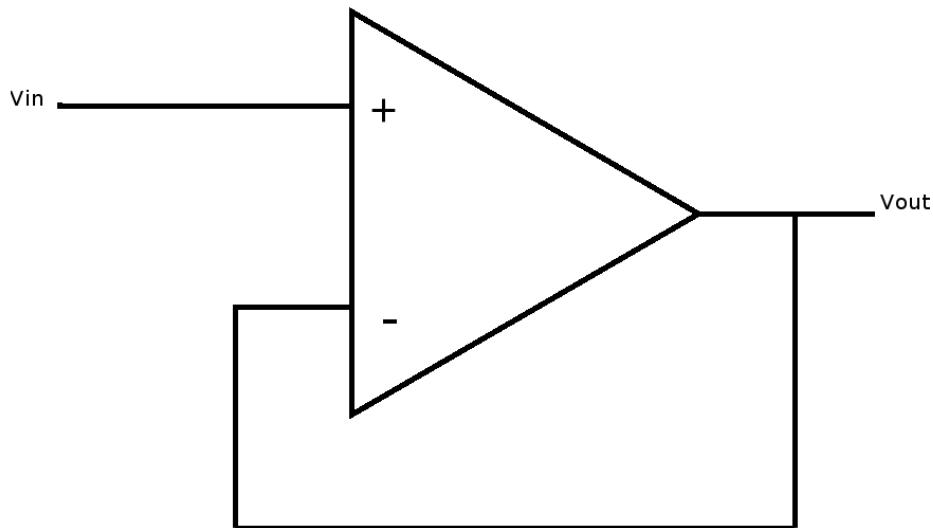
| | inp_sel | inn_sel | en_res_bank | res_mux_sel | R1_sel | R2_sel | res_to_out_vdd |
|-----------------|---------|---------|-------------|-------------|--------|--------|----------------|
| opamp14'd1-4'd5 | 3'd0 | 1'd0 | - | - | - | - | - |
| opamp23'd1 | 2'd0 | 1'd0 | - | - | - | - | - |
| opamp33'd1 | 2'd0 | 1'd0 | - | - | - | - | - |

19.5.8 Built-in modes

Unity Gain Buffer or Voltage Follower

In the unity gain buffer configuration the output is connected to inverting input internally and the input has to be applied to non-inverting input. It has a 3dB bandwidth greater than 6MHz.

| | inp_sel | inn_sel | en_res_bank | res_mux_sel | R1_sel | R2_sel | res_to_out_vdd |
|-----------------|---------|---------|-------------|-------------|--------|--------|----------------|
| opamp14'd0-4'd5 | 3'd4 | 1'b0 | - | - | - | - | - |
| opamp23'd0-3'd2 | 2'd3 | 1'b0 | - | - | - | - | - |
| opamp33'd0-3'd1 | 2'd3 | 1'b0 | - | - | - | - | - |

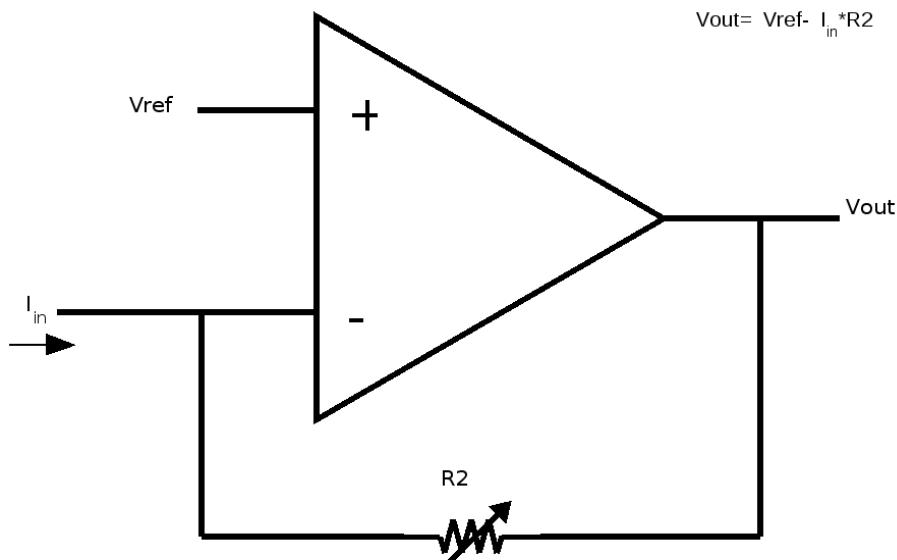


$$\text{offset} = \text{Voff} = \text{vout} - \text{vin}$$

Trans-Impedance Amplifier (TIA)

The TIA converts an internal or external current to an output voltage. It uses an internal resistor to convert input current to output voltage. The resistor is connected from inverting pin to output. The resistor value can be programmed from $20\text{K}\Omega$ to $1\text{M}\Omega$. A reference voltage has to be applied to non inverting input of opamp. Then the output of opamp is $\text{Vref} - \text{Iin} \times R$.

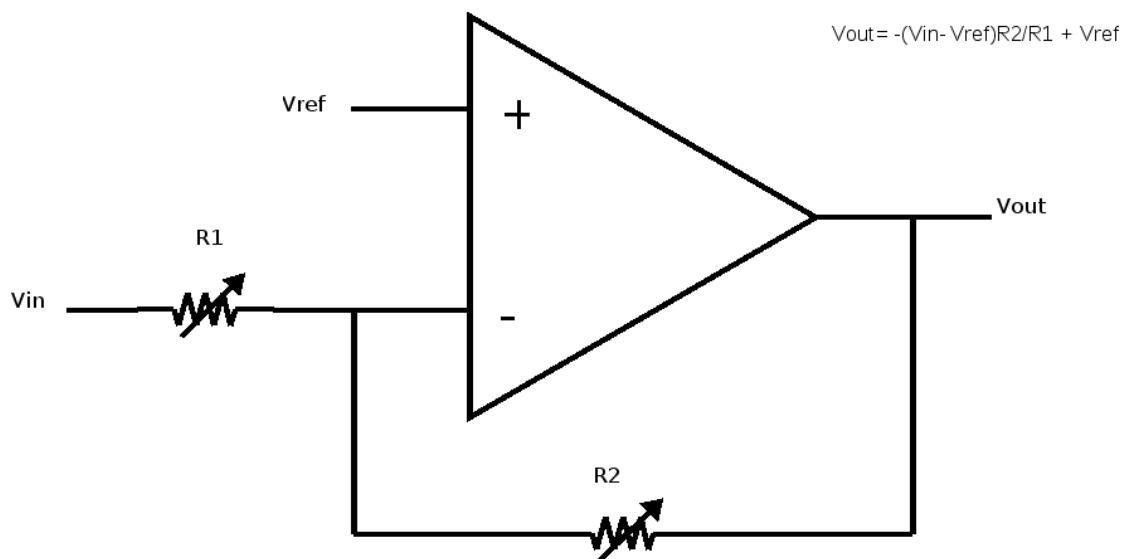
| | inp_sel | inn_sel | en_res_bank | res_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|--------|---------|-----------|-------------|-------------|-------------|-------------|----------------|
| opamp1 | 4'd6 | 3'd0-3'd1 | 1 | 3'd0-3'd1 | 0 | 0-7 | 1'd0 |
| opamp2 | 3'd3 | 2'd0 | 1 | 3'd0 | 0 | 0-7 | 2'd0 |
| opamp3 | 3'd2 | 2'd0 | 1 | 3'd0 | 0 | 0-7 | 1'd0 |



Inverting PGA

This mode amplifies an internal or external signal. In inverting amp configuration gain is $-R_2/R_1$, where R_1, R_2 are selected from $R1_sel, R2_sel$ controls.

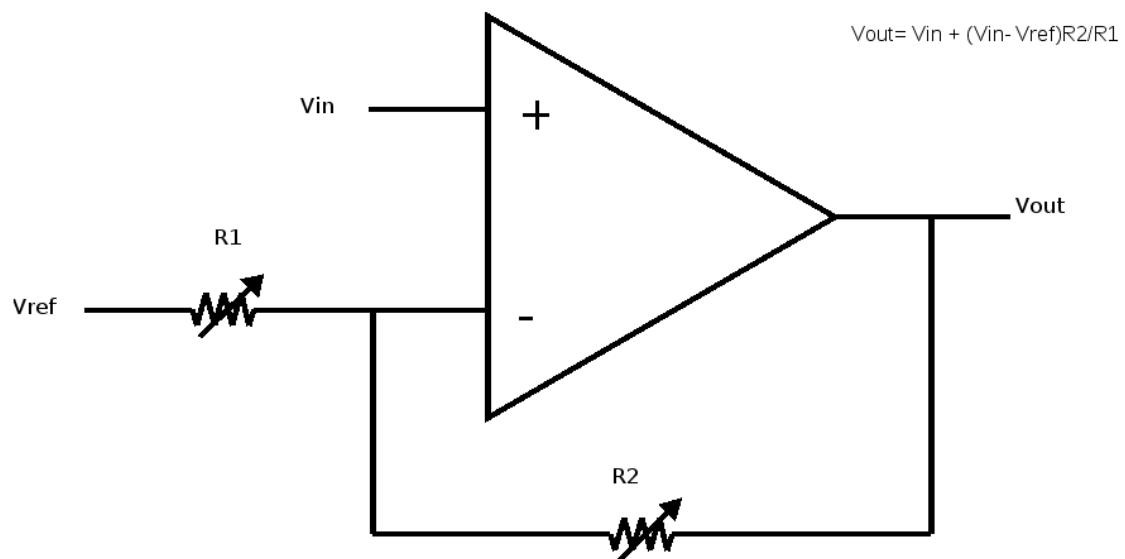
| | inp_sel | inn_selen | res_bank | res_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|------------|---------|-----------|----------|-------------|-------------|-------------|----------------|
| opamp14'd6 | 3'd3 | 1 | | 3'd0-3'd5 | 1-3 | 0-7 | 1'b0 |
| opamp23'd3 | 2'd2 | 1 | | 3'd0-3'd2 | 1-3 | 0-7 | 2'd0 |
| opamp33'd2 | 2'd2 | 1 | | 3'd0-3'd1 | 1-3 | 0-7 | 1'b0 |



Non-Inverting PGA

This mode amplifies an internal or external signal. In non inverting amp configuration gain is $1+R_2/R_1$, where R_1 , R_2 are selected from $R1_sel$, $R2_sel$ controls.

| | inp_sel | inn_selen_res_bank | res_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|---------------------|---------|--------------------|-------------|-------------|-------------|----------------|
| opamp14'd0-4'd53'd3 | 1 | | 3'd6 | 1-3 | 0-7 | 1'b0 |
| opamp23'd0-3'd22'd2 | 1 | | 3'd3 | 1-3 | 0-7 | 2'd0 |
| opamp33'd0-3'd12'd2 | 1 | | 3'd2 | 1-3 | 0-7 | 1'b0 |



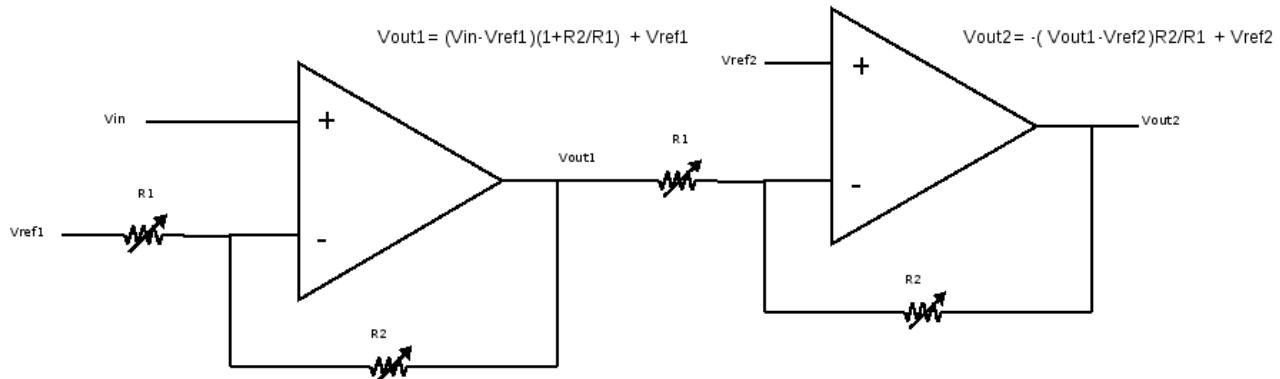
$$V_{out} = V_{in} + (V_{in} - V_{ref})R_2/R_1 + V_{off}(1+R_2/R_1)$$

Cascaded Inverting PGA

The 2 opamps, one in non inverting and the other in inverting PGA configuration can be cascaded to get cascaded inverting PGA using following settings.

| inp_sel | inn_selen_res_bankres_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|---------------------|-------------------------------|-------------|-------------|----------------|
| opamp14'd0-4'd53'd3 | 1 | 3'd6 | 1-3 | 0-7 |
| opamp23'd3 | 2'd2 | 1 | 3'd5 | 1-3 |

This can be implemented using opamp2 and opamp3 also in the same way.

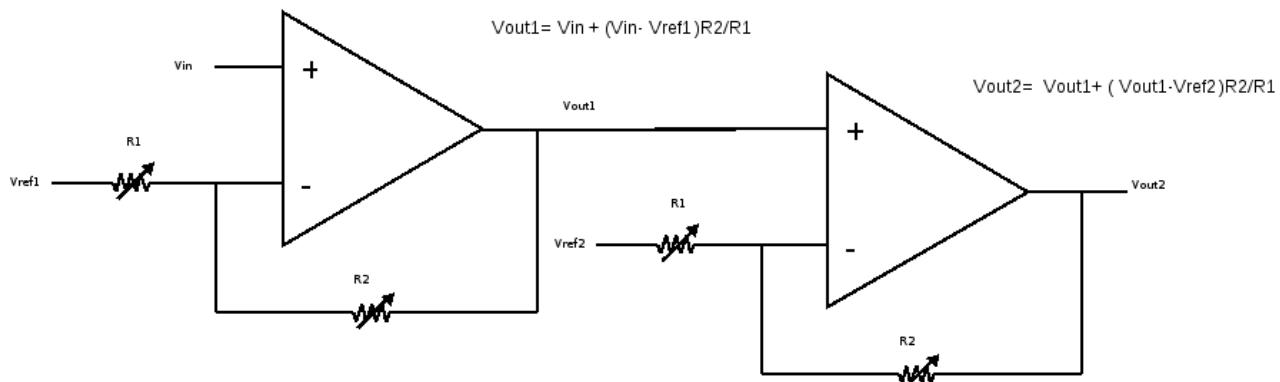


Cascaded Non-Inverting PGA

The 2 opamps each in non inverting PGA configuration can be cascaded to get cascaded non inverting PGA using following settings.

| inp_sel | inn_selen_res_bankres_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|---------------------|-------------------------------|-------------|-------------|----------------|
| opamp14'd0-4'd53'd3 | 1 | 3'd6 | 1-3 | 0-7 |
| opamp23'd6 | 2'd2 | 1 | 3'd3 | 1-3 |

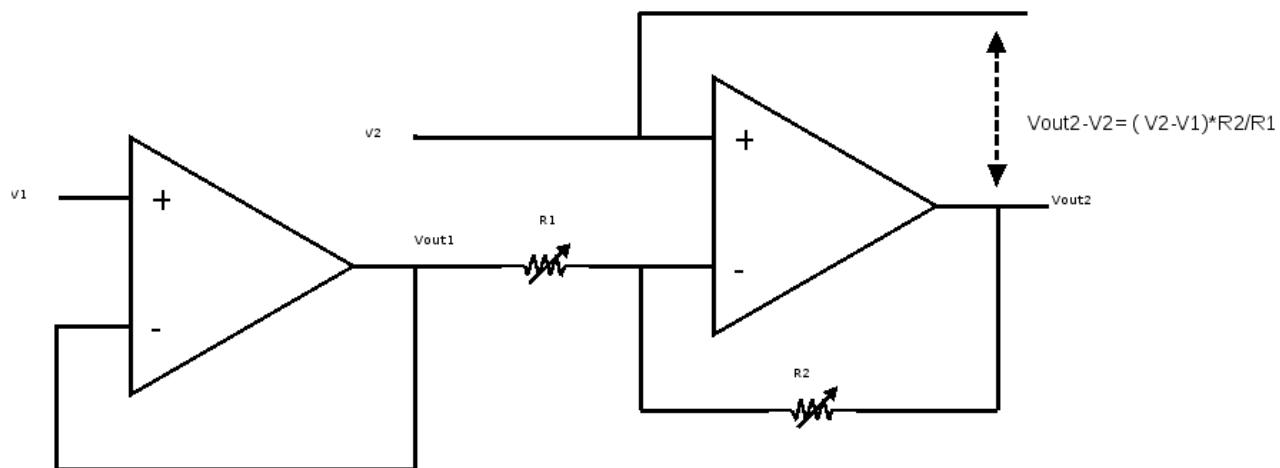
This can be implemented using opamp2 and opamp3 also in the same way.



Two opamps Differential Amplifier

This configuration can be achieved by following settings. The 1st Opamp is in UGB mode and 2nd Opamp is in PGA mode. The differential output is taken between output and non inverting input of 2nd Opamp.

| | inp_sel | inn_selen_res_bank | res_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|--------|-----------|--------------------|-------------|-------------|-------------|----------------|
| opamp1 | 4'd0-4'd4 | 53'd4 | 0 | -- | -- | -- |
| opamp2 | 3'd0-3'd2 | 22'd2 | 1 | 3'd5 | 1-3 | 0-7 |

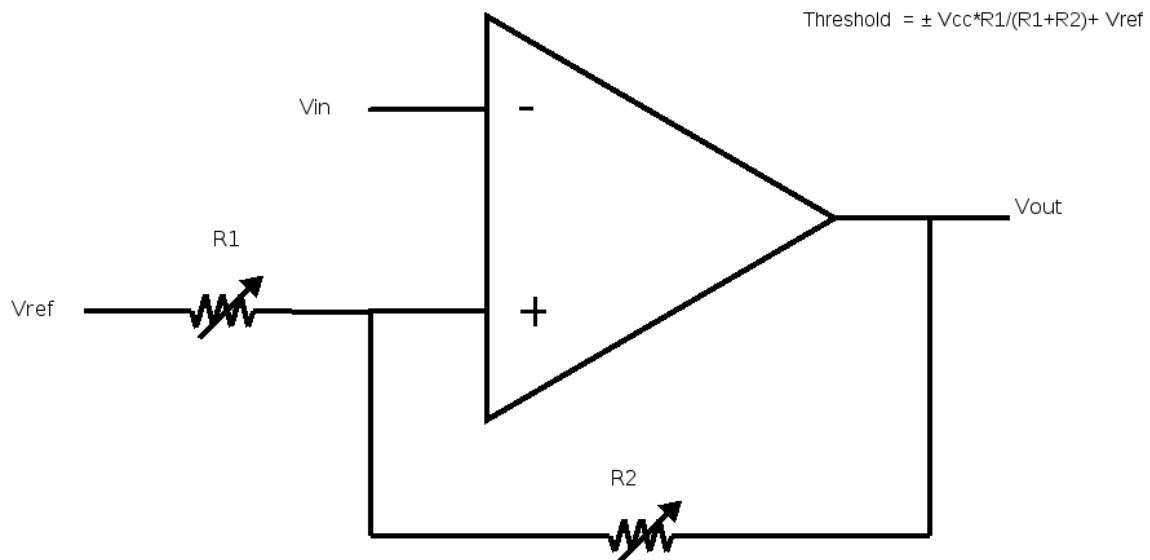


Inverting Comparator with Programmable Hysteresis

In both inverting and non-inverting comparator configurations, the non inverting input of Opamp is connected to resbank.

The Opamp can be configured as inverting comparator using these settings.

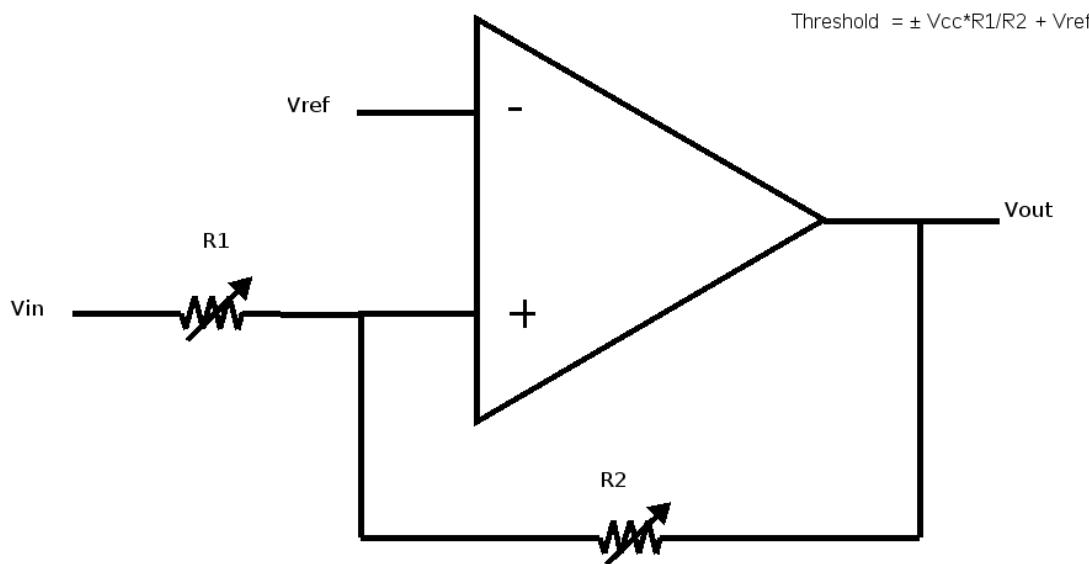
| | inp_sel | inn_sel | en_res_bank | res_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd | |
|--------|---------|-----------|-------------|-------------|-------------|-------------|----------------|------|
| opamp1 | 4'd7 | 3'd0-3'd1 | 1 | 1 | 6'd3 | 1-3 | 0-7 | 1'b0 |
| opamp2 | 3'd4 | 2'd0 | 1 | 1 | 3'd3 | 1-3 | 0-7 | 2'd0 |
| opamp3 | 3'd3 | 2'd0 | 1 | 1 | 3'd2 | 1-3 | 0-7 | 1'b0 |



Non-Inverting Comparator with Programmable Hysteresis

The Opamp can be configured as non inverting comparator using following settings.

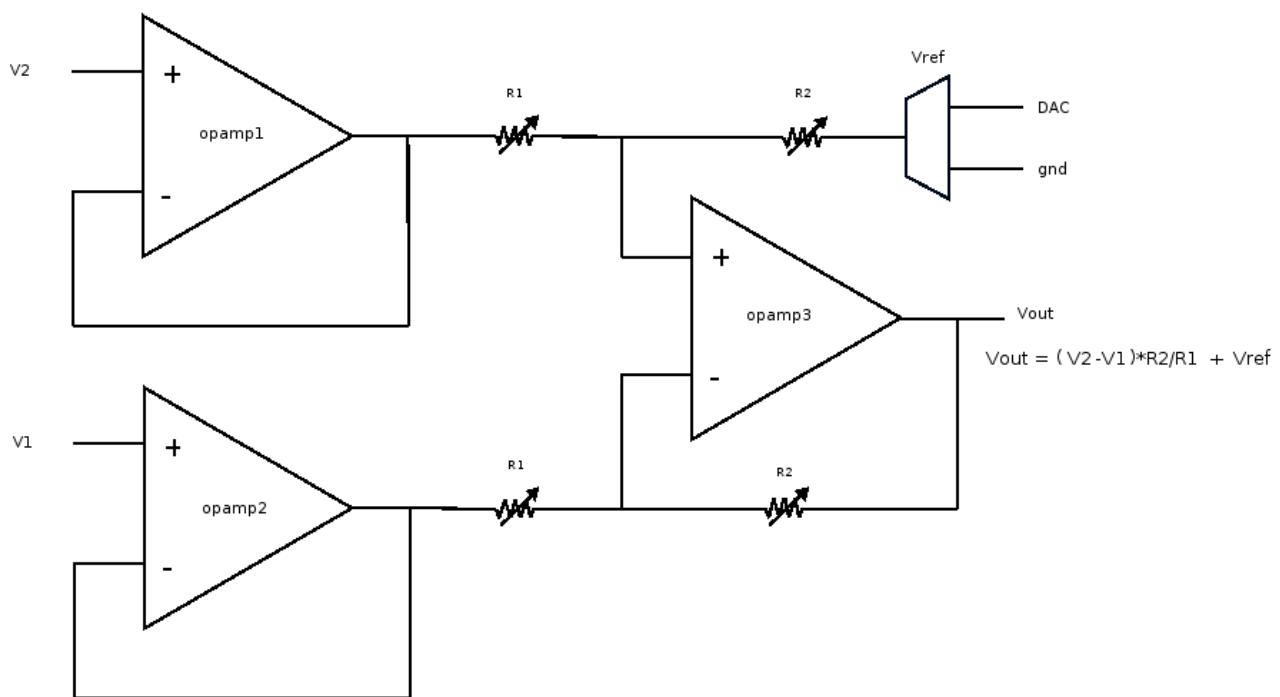
| | inp_sel | inn_selen | res_bank | res_mux | sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|------------|---------|-----------|----------|-----------|-----|-------------|-------------|----------------|
| opamp14'd7 | 3'd2 | 1 | | 3'd0-3'd5 | 1-3 | 0-7 | 1'b0 | |
| opamp23'd4 | 2'd1 | 1 | | 3'd0-3'd2 | 1-3 | 0-7 | 2'd0 | |
| opamp33'd3 | 2'd1 | 1 | | 3'd0-3'd1 | 1-3 | 0-7 | 1'b0 | |



Instrumentation Amplifier

In this mode opamp1 and opamp2 are configured as voltage follower and opamp3 will amplify the differential voltage of opamp1 and opamp2's outputs.

| inp_sel | inn_selen_res_bank | res_mux_sel | R1_sel<1:0> | R2_sel<2:0> | res_to_out_vdd |
|---------------------|--------------------|-------------|-------------|-------------|----------------|
| opamp14'd0-4'd53'd4 | 0 | -- | -- | -- | -- |
| opamp23'd0-3'd22'd3 | 1 | 3'd5 | 1-3 | 0-7 | 2'd2 |
| opamp33'd6 | 2'd2 | 1 | 3'd4 | 1-3 | 0-7 |
| | | | | | 1'd0 |



19.5.9 Resistor banks

There are two sets of resistor banks, one for R1 and other for R2. R1 can be programmed using R1_sel<1:0> and R2 can be programmed using R2_sel<2:0>.

| R1_sel<1:0> | R1 (KΩ) |
|-------------|-----------|
| 0 | 0 (short) |
| 1 | 20 |
| 2 | 60 |
| 3 | 140 |

| R2_sel<2:0> | R2 (KΩ) |
|-------------|---------|
| 0 | 20 |
| 1 | 30 |
| 2 | 40 |
| 3 | 60 |
| 4 | 120 |
| 5 | 250 |
| 6 | 500 |
| 7 | 1000 |

19.5.10 Opamp's output on GPIO

```
opamp1_out - opamp1_out_mux_en=1, opamp1_out_mux_sel=0 - AGPIO4
opamp1_out - opamp1_out_mux_en=1, opamp1_out_mux_sel=1 - AGPIO15
opamp2_out - opamp2_out_mux_en=1 - AGPIO9
opamp3_out - opamp3_out_mux_en=1 - TopGPIO2
```

19.5.11 Guidelines for 'Ton' Selection

Ton represents the time required for opamp output to settle to 12 bit accuracy from enable signal.

| Configuration | Gain | HP/LP mode | TYP | MAX | Units |
|-----------------------------------|------|------------|--------|--------|-------|
| Non Inverting Amplifier | 2 | HP | 285n | 846n | s |
| | | LP | 933n | 2.1u | s |
| | 50 | HP | 9.1u | 12.62u | s |
| | | LP | 14.53u | 21.1u | s |
| Inverting Amplifier | -1 | HP | 293n | 846n | s |
| | | LP | 959n | 2.09u | s |
| | -50 | HP | 9.09u | 12.64u | s |
| | | LP | 14.49u | 21.15u | s |
| Instrumentation Amplifier | 1 | HP | 285n | 1.01u | s |
| | | LP | 774n | 1.75u | s |
| | 50 | HP | 9.16u | 12.91u | s |
| | | LP | 14.98u | 21.81u | s |
| Cascaded Non Inverting Amplifier4 | | HP | 380n | 1.471u | s |
| | | LP | 900n | 1.86u | s |
| | 2601 | HP | 18u | 25u | s |
| | | LP | 29u | 42u | s |

19.5.12 Register Summary

Base Address: 0x24043800

| Register Name | Offset | Reset Value | Description |
|---------------|--------|--------------|-----------------|
| OPAMP1 1 | 0x214 | 0x 0000 0004 | Programs opamp1 |
| OPAMP2 1 | 0x218 | 0x 0000 0004 | Programs opamp2 |
| OPAMP3 1 | 0x21C | 0x 0000 0004 | Programs opamp3 |

[1347 . Register Summary](#)

19.5.13 Register Description

OPAMP_1

| Bit | Access | Function | Default value | Description |
|-------|--------|------------------------|---------------|--|
| 31 | R/W | opamp1_dyn_en | 0 | Dynamic Enable For Opamp1 signals |
| 30:27 | R/W | vref_mux_sel | 0 | Vref Mux Sel |
| 26 | | | | |
| 25:22 | R/W | vref_mux_en | 0 | VRef Mux Enable |
| 21 | R/W | mems_res_bank_en | 0 | Enable Mems Res Bank |
| 20 | R/W | opamp1_out_mux_sel | 0 | 1 - To connect opamp1 output to pad |
| 19:16 | R/W | opamp1_inp_sel | 0 | Selecting +ve input of opamp |
| 15:13 | R/W | opamp1_inn_sel | 0 | Selecting -ve input of opamp |
| 12 | R/W | opamp1_out_mux_en | 0 | Out Mux Enable |
| 11 | R/W | opamp1_res_to_out_vdd0 | | 0 – connect resbank to out 1 – connect resbank to vdd |
| 10:8 | R/W | opamp1_res_mux_sel | 0 | Selecting input for resistor bank |
| 7 | R/W | opamp1_en_res_bank | 0 | 0 – disable 1 – enable resistor bank |
| 6:4 | R/W | opamp1_R2_sel | 0 | Programmability to select resister bank R2 |
| 3:2 | R/W | opamp1_R1_sel | 1 | Programmability to select resister bank R1 |
| 1 | R/W | opamp1_lp_mode | 0 | 0 – normal mode 1 – low power mode |
| 0 | R/W | opamp1_enable | 0 | 0 - disable 1 - to enable opamp 1 |

1348 . Register Description

OPAMP_2

| Bit | Access | Function | Default value | Description |
|-------|--------|------------------------|---------------|--|
| 31:20 | | | | |
| 19 | R/W | opamp2_dyn_en | 0 | Dynamic Enable For Opamp2 signals |
| 18:16 | R/W | opamp2_inp_sel | 0 | Selecting +ve input of opamp |
| 15:14 | R/W | opamp2_inn_sel | 0 | Selecting -ve input of opamp |
| 13 | R/W | opamp2_out_mux_en | 0 | Out Mux Enable |
| 12:11 | R/W | opamp2_res_to_out_vdd0 | | 0 – connect resbank to out 1 – connect resbank to vdd 2 – connect resbank to DAC 3 – connect resbank to gnd |
| 10:8 | R/W | opamp2_res_mux_sel | 0 | Selecting input for resistor bank |

| Bit | Access | Function | Default value | Description |
|-----|--------|--------------------|---------------|--|
| 7 | R/W | opamp2_en_res_bank | 0 | 0 – disable 1 – enable resistor bank |
| 6:4 | R/W | opamp2_R2_sel | 0 | Programmability to select resister bank R2 |
| 3:2 | R/W | opamp2_R1_sel | 1 | Programmability to select resister bank R1 |
| 1 | R/W | opamp2_lp_mode | 0 | 0 – normal mode 1 – low power mode |
| 0 | R/W | opamp2_enable | 0 | 0 - disable 1 - to enable opamp 2 |

1349 . Register Description

OPAMP_3

| Bit | Access | Function | Default value | Description |
|-------|--------|------------------------|---------------|--|
| 31:20 | | | | |
| 18 | R/W | opamp3_dyn_en | 0 | Dynamic Enable For Opamp3 signals |
| 17:15 | R/W | opamp3_inp_sel | 0 | Selecting +ve input of opamp |
| 14:13 | R/W | opamp3_inn_sel | 0 | Selecting -ve input of opamp |
| 12 | R/W | opamp3_out_mux_en | 0 | Out Mux Enable |
| 11 | R/W | opamp3_res_to_out_vdd0 | | 0 – connect resbank to out 1 – connect resbank to vdd |
| 10:8 | R/W | opamp3_res_mux_sel | 0 | Selecting input for resistor bank |
| 7 | R/W | opamp3_en_res_bank | 0 | 0 – disable 1 – enable resistor bank |
| 6:4 | R/W | opamp3_R2_sel | 0 | Programmability to select resister bank R2 |
| 3:2 | R/W | opamp3_R1_sel | 1 | Programmability to select resister bank R1 |
| 1 | R/W | opamp3_lp_mode | 0 | 0 – normal mode 1 – low power mode |
| 0 | R/W | opamp3_enable | 0 | 0 - disable 1 - to enable opamp 3 |

1350 . Register Description

19.6 Temperature Sensor: RO Based and BJT Based

19.6.1 General Description

There are two independent temperature sensors integrated. Ring Oscillator(RO) based temperature sensor and BJT based temperature sensor.

BJT based sensor works for temperature range from -40degree to 125degree and voltage variation from 1.8V to 3.6V. It outputs the digital word having resolution of nearly 1 degree C. The conversion time is 2 clock cycles of ADC after turning ON the temperature sensor.

RO based sensor outputs 2 clocks (clk_f1, clk_f2). The temperature is determined by counting the clocks and applying an equation.

19.6.2 BJT Temperature Sensor

Register Summary

| Address | Register Name |
|---------------------|----------------|
| base address | 0x24043800 |
| 1E0 | TS_PTAT_ENABLE |

Register Description

| TS_PTAT_ENABLE | | | | |
|----------------------------|------------------------|----------------------------------|-----|--|
| <u>Offset Address: 1E0</u> | | | | |
| <u>Bit</u> | <u>Access Function</u> | <u>Default Value Description</u> | | |
| [31:1] | R | Reserved | 0x0 | Reserved |
| [0] | R/W | ts_ptat_enable | 0x0 | BJT based Temperature sensor enable 1 : Enable 0 : Disable |

19.6.3 Ring Oscillator Based Temperature Sensor

General Features

Temperature reading from the RO based temp sensor in degrees Celsius is accessible through APB through the register "Temperature Read".

The sequence described at the end of this section needs to be followed for obtaining a fresh temperature reading.

Register Summary

| Address | Register Name |
|---------------------|---------------|
| base address | 0x2404_8500 |

| Address | Register Name |
|---------|---------------------------------------|
| 0x00 | TS ENABLE AND TEMPERATURE DONE |
| 0x04 | TS SLOPE SET |
| 0x08 | TS NOMINAL SETTINGS |
| 0x0C | TS COUNTS READ |
| 0x10 | TEMPERATURE READ |

Register Description

| TS ENABLE AND TEMPERATURE DONE | | | | |
|---------------------------------------|---------------|-----------------------|----------------------|---|
| Address: 0x00 | | | | |
| Bit | Access | Function | Default Value | Description |
| 31:13 | N/A | Reserved | 0 | Reserved |
| 12 | R | temp_measurement_done | 0 | temperature measurement done indication |
| 11:2 | R/W | cont_count_freeze | 10'd255 | count of reference clock on which ptat clock counts |
| 1 | R/W | ref_clk_sel | 0 | 0: use KHz RO clock from analog as reference clock 1: use MCU FSM clock as reference clock |
| 0 | W | temp_sens_en | 0 | Temperature sensing enable Note: Self clearing register. |

| TS SLOPE SET | | | | |
|----------------------|---------------|-----------------|----------------------|--|
| Address: 0x04 | | | | |
| Bit | Access | Function | Default Value | Description |
| 31:29 | N/A | Reserved | 0 | Reserved |
| 28 | R/W | bjt_based_temp | 0 | Temperature is updated through which is calculated using bjt based 1 - through spi 0- through calculation RO based |

TS SLOPE SET

Address: 0x04

| Bit | Access | Function | Default Value | Description |
|-------|--------|-----------------|---------------|---|
| 27 | W | temp_updated | 0 | Temperature updated signal for the reg to capture this temperature |
| 26:16 | R/W | temperature_spi | 11'd0 | Temperature known |
| 15:10 | N/A | Reserved | 0 | Reserved |
| 9:0 | R/W | slope | 10'h03B | This is the slope of ptat clock variation with respect to temperature changes. This is one time measurement for one package after chip arrives from fab. This is signed bit. |

TS NOMINAL SETTINGS

Address: 0x08

| Bit | Access | Function | Default Value | Description |
|-------|--------|---------------------|---------------|--------------------------------------|
| 31:23 | N/A | Reserved | 0 | Reserved |
| 22:16 | R/W | nominal temperature | 7'd25 | calibrated temperature |
| 15:10 | N/A | Reserved | 0 | Reserved |
| 9:0 | R/W | f2_nominal | 10'd270 | ptat clock count during calibration. |

TS COUNTS READ

Address: 0x0C

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------|---------------|------------------------|
| 31:26 | N/A | Reserved | 0 | Reserved |
| 25:16 | R | count_f1 | 0 | count of clk_f1 cycles |
| 15:10 | N/A | Reserved | 0 | Reserved |
| 9:0 | R | count_f2 | 0 | count of clk_f2 cycles |

TEMPERATURE READ

Address: 0x10

| Bit | Access | Function | Default Value | Description |
|-------|--------|----------------|---------------|--|
| 31:11 | N/A | Reserved | 0 | Reserved |
| 10:0 | R | temperature_rd | 0 | Temperature value for read in signed format. |

RO Temp Sensor Read Sequence

1. Program the nominal temperature and f2_nominal bits in register at address **508** in bit positions [22:16] and [9:0] respectively.
2. In register **504**, disable BJT based temperature bit. And program [9:0] bits for slope.
3. In register **500**, Program bits 11:2 for const count freeze, and enable temp sensor calculation. Make bit[0] to 1.
4. Wait until you can read 1 in bit 12 of **500**. i.e., temp_measurement_done.
5. Read 10:0 bits in **510** for temperature calculated.

Values:

| | | | | |
|-------|-----|----------|---|----------------------|
| Write | 508 | 0019020E | for f2 nominal and nominal temperature | 0 |
| Write | 504 | 0000003B | for slope and disabling BJT based | 0 |
| Write | 500 | 000003FD | To give clock1 count and enable calculation | 0 |
| Read | 500 | -- | For temperature measurement done signal | wait (read[12] == 1) |
| Read | 510 | -- | temperature = read[10:0] | 0 |

19.7 Touch Capacitance sensor

19.7.1 General Description

The touch capacitance sensor can detect changes in the input capacitance at a pin and represent it in the form of changes in raw counts. The increase in input capacitance is an indication of the presence of a human body near the touch panel connected to the pin(s). The increased capacitance would be detected as an increase in the raw counts.

If the count value increases by more than a set percentage from threshold value, it is considered as touch.

When multiple pins are used it is possible to detect touch location by processing the count values across the pins and how the counts varies over time. This is accomplished in the Touch peripheral driver/APIs.

Features

Analog

Following are the features of the Touch Capacitance sensor Analog block:

1. Capacitance sensor GPIO connection details are found at AnalogPeripherals-I/O Pin Multiplexing
2. Capacitance sensor has 8 input channels, 1 cap input and 1 resistor input.
3. All the input channels are reconnected to GPIOs.

-
4. Capacitive input and resistor input are connected to two GPIOs each. Select one of the GPIOs using sel_Cint_res_in1 and sel_res_out1.
 5. Has an option to use external reference instead of internal programmable reference. To choose external reference, make bg_ref_sel low.
 6. The internal reference can be programmed using 3 bit word vref_sel.

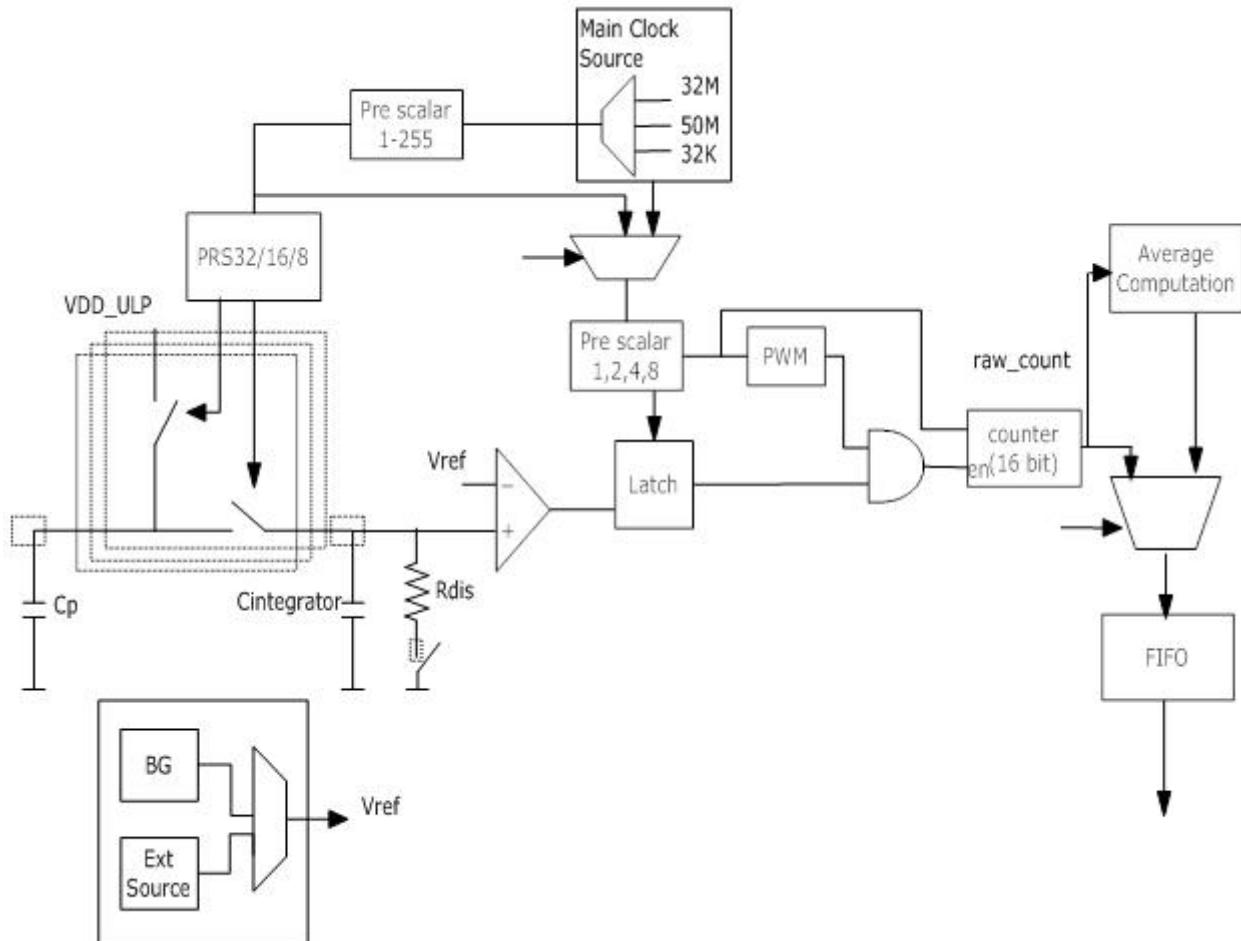
Digital

Following are the features of the Touch Capacitance sensor Digital controller block:

1. Ability to Selects one of the 8 input clock sources rf_ref_clk , ulp_32khz_ro_clk , ulp_32khz_rc_clk , ulp_32khz_xtal_clk , ulp_32mhz_rc_clk , ulp_20mhz_ro_clk , soc_clk and ulp_doubler_clk
2. Ability to Control the rate of scanning for all sensors with configurable inter sensor scan ON time
3. Supports both samples streaming and cumulative average mode
4. Uses PWM for generating a measurement window for counting the raw sample counts
5. Has 8 to 16 bit PWM resolution
6. Has clock divider, which supports up to 256 division factor
7. Supports APB interface
8. Has asynchronous FIFO size of 64x16
9. Supports DMA flow control signals to access data via DMA to reduce processor load
10. 8,16 and 32-bit pseudo-random number for generating two non overlapping streams with configurable delay
11. Programmable polynomial and seed values for pseudo-random number generator
12. Provides Wake up indication after capacitive touch sensing
13. Programmable Vref to get accurate sensing
14. Raises an interrupt after FIFO occupancy crosses the configurable threshold value

Capacitive touch Sensor

19.7.2 Block Diagram



19 . Block Diagram of Capsense

19.7.3 Functional Description

- ULP_VDD_33 and MUXED_ULP_SS_VDD supplies should be available for the cap sensor to work.
- The input channels will be selected using sense_on<7:0> signal. If sense_on<n> signal is high, nth channel will be selected. sense_on<7:0> is a one-hot code.
- Clocks fsw1, fsw2 and flatch will come from capsense digital, and have programmable dividers. dividers
- The Cintegrator is an external cap of 2nF that has to be connected to capacitive input of capsensor.

The capacitance sensor detects the input capacitance and represent it in the form of count which is computed as follows:

$$\text{count} = \text{Rdis} * \text{fsw1} * \text{Cp} * (\text{VBATT} - \text{vref}) / \text{vref} \quad (\text{vref has to be trimmed to get nominal count of 0.5 - 0.8})$$

Rdis - discharging resistor

fsw1 - frequency of fsw1 clock coming from capsense digital

Cp - Input/sensor capacitance

For Example: Rdis=30KΩ, fsw1=1MHz, Cp=10pF

| VBATT | Vref | count |
|-------|------|-------|
| 3.6 | 1 | 0.78 |
| 3 | 0.9 | 0.7 |
| 2.4 | 0.7 | 0.73 |
| 1.8 | 0.5 | 0.78 |

Real value will be count * PWM_on_period.

19.7.4 Resistor Selection

| Operating Freq | Rdis |
|----------------|-------|
| 10MHz | 3KΩ |
| 1MHz | 30KΩ |
| 100KHz | 300KΩ |

The above resistor values are optimally recommended to make capsense work for voltage range of 1.8-3.6V, when input cap Cp=10pF

19.7.5 Vref Selection

| vref_sel<2:0> | vref | units |
|---------------|------|-------|
| 0 | 0.5 | V |
| 1 | 0.6 | V |
| 2 | 0.7 | V |
| 3 | 0.8 | V |
| 4 | 0.9 | V |
| 5 | 1.0 | V |
| 6 | 1.1 | V |
| 7 | 1.2 | V |

19.7.6 GPIO Selection

Sensor Input

| sense_on<7:0> | GPIO |
|---------------|---------|
| 8'h01 | AGPIO8 |
| 8'h02 | AGPIO9 |
| 8'h04 | AGPIO10 |
| 8'h08 | AGPIO7 |
| 8'h10 | AGPIO6 |
| 8'h20 | AGPIO3 |
| 8'h40 | AGPIO0 |
| 8'h80 | AGPIO11 |

Integration Cap input

| sel_Cint_res_in1 | GPIO |
|------------------|--------|
| 0 | AGPIO8 |
| 1 | AGPIO4 |

Discharging Resistor input

| sel_res_out1 | GPIO |
|--------------|---------|
| 0 | AGPIO10 |
| 1 | AGPIO5 |

19.7.7 Register Summary

Base Address: 0x24042C00

There are 10 registers each of which is 32 bits wide.

CONFIG_REG_0_0 uses p-clock.

CONFIG_REG_1_1 , CONFIG_REG_1_2 ,...., CONFIG_REG_1_8 , CONFIG_REG_1_9 use core-clock.

| Register Name | Offset | Reset Value | Description |
|----------------|--------|-------------|-------------|
| CONFIG_REG_0_0 | 0x000 | | |
| CONFIG_REG_1_1 | 0x100 | | |
| CONFIG_REG_1_2 | 0x104 | | |
| CONFIG_REG_1_3 | 0x108 | | |
| CONFIG_REG_1_4 | 0x10C | | |
| CONFIG_REG_1_5 | 0x110 | | |
| CONFIG_REG_1_6 | 0x114 | | |
| CONFIG_REG_1_7 | 0x118 | | |
| CONFIG_REG_1_8 | 0x11C | | |
| CONFIG_REG_1_9 | 0x120 | | |
| FIFO address | 0x004 | | |

1351 . Register Summary

19.7.8 Register Description

CONFIG_REG_0_0

| Address : 0x000 | | | | |
|-----------------|--------|--------------------|---------------|---------------------------|
| Bits | Access | Name | Default value | Description |
| 31:29 | -- | Reserved | | |
| 28 | R | fifo_empty | 1'd0 | FIFO empty status bit |
| 27:22 | R/W | fifo_aempty_thrlid | 6'd0 | Threshold for fifo aempty |

| Address : 0x000 | | | | |
|------------------------|---------------|-------------------|----------------------|--|
| Bits | Access | Name | Default value | Description |
| 21:16 | R/W | fifo_afull_thrlid | 6'd0 | Threshold for fifo afull |
| 15 | R/W | cts_static_clk_en | 1'd0 | 1 - Clocks are not gated 0 - Clocks are gated |
| 14 | R/W | clk_sel2 | 1'd0 | Mux select for clock_mux_2 |
| 13:10 | R/W | pre_scalar_2 | 4'd0 | Division factor for clock divider |
| 9:2 | R/W | pre_scalar_1 | 8'd0 | Division factor for clock divider |
| 1:0 | R/W | clk_sel1 | 2'd0 | Mux select for clock_mux_1 |

FIFO

| Address : 0x004 | | | | |
|------------------------|---------------|-------------|----------------------|--|
| Bits | Access | Name | Default value | Description |
| 31:0 | R/W | FIFO | 32'd0 | Used for FIFO reads & write operations |

CONFIG_REG_1_1

| Address : 0x100 | | | | |
|------------------------|---------------|--------------|----------------------|--|
| Bits | Access | Name | Default value | Description |
| 31:20 | -- | Reserved | | |
| 19 | R/W | ext_trig_en | 1'd0 | Select bit for NPSS clock / Enable 1 : NPSS clock 0 : Enable |
| 18 | R/W | ext_trig_sel | 1'd0 | Select bit for NPSS 1ms / 1s clock 1 : 1 ms clock 0 : 1s clock |
| 17:16 | R/W | bit_sel | 2'd0 | Selects different set of 12 bits to be stored in FIFO |

Address : 0x100

| Bits | Access | Name | Default value | Description |
|-------------|---------------|-----------------|----------------------|--|
| 15 | R/W | bypass | 1'd0 | Bypass signal 1 – Bypass the Random number generator output to the Non-overlapping stream generator and to give “clk” as input to the Non-Overlapping stream generator. 0 – Use Random number generator output bit as input to Non-Overlapping stream generator. |
| 14 | W | reset_wr_fifo | 1'd0 | Resets the signal fifo_wr_int 0 - Reset 1 - Out of reset |
| 14 | R | fifo_aempty | 1'd0 | FIFO aempty status bit |
| 13:12 | R/W | sample_mode | 2'd0 | Select bits for FIFO write and FIFO average sample_mode[1] : 1 - No FIFO Write 0 - FIFO Write sample_mode[0] : 1 - Average 0 - No average |
| 11 | R/W | cnt_onehot_mode | 1'd0 | Continuous or One hot mode 1 – Continuous 0 – One hot |
| 10 | R/W | soft_reset_2 | 1'd0 | Reset the FIFO write and FIFO read occupancy pointers 1 - Reset 0 - Out of reset |
| 9 | R/W | enable | 1'd0 | Enable signal 1 – enable the cap sensor module 0 – disable the cap sensor module |

Address : 0x100

| Bits | Access | Name | Default value | Description |
|-------------|---------------|---------------------------|----------------------|--|
| 8 | R/W | Ack for wake up interrupt | 1'd0 | Ack for wake up interrupt |
| 7:3 | R/W | buffer_delay | 5'd0 | <p>Delay of buffer. Delay programmed will be equal to delay in nano seconds. Max delay value is 32.</p> <p>Default delay should be programmed before using Capacitive touch sensor module.</p> |
| 2 | R/W | seed_load | 1'd0 | <p>Seed of polynomial</p> <p>1 – to load the seed</p> <p>0 – loading of seed is not allowed</p> |
| 1:0 | R/W | polynomial_len | 2'd0 | Length of polynomial |

CONFIG_REG_1_2

Address : 0x104

| Bits | Access | Name | Default value | Description |
|-------------|---------------|----------------|----------------------|--------------------|
| 31:16 | R/W | pwm_off_period | 16'd0 | PWM OFF period |
| 15:0 | R/W | pwm_on_period | 16'd0 | PWM ON period |

CONFIG_REG_1_3

Address : 0x108

| Bits | Access | Name | Default value | Description |
|-------------|---------------|-------------|----------------------|--|
| 31:0 | R/W | prs_seed | 32'd0 | Pseudo random generator (PRS) seed value |

CONFIG_REG_1_4

| Address : 0x10C | | | | |
|------------------------|---------------|-------------|----------------------|---|
| Bits | Access | Name | Default value | Description |
| 31:0 | R/W | prs_poly | 32'd0 | Polynomial programming register for PRS generator |

CONFIG_REG_1_5

| Address : 0x110 | | | | |
|------------------------|---------------|--------------------|----------------------|--------------------------------------|
| Bits | Access | Name | Default value | Description |
| 31:16 | R/W | N_sample_count | 16'd0 | Number of repetitions of sensor scan |
| 15:0 | R/W | inter_sensor_delay | 16'd0 | Inter-sensor scan delay value |

CONFIG_REG_1_6

| Address : 0x114 | | | | |
|------------------------|---------------|-------------|----------------------|---|
| Bits | Access | Name | Default value | Description |
| 31:0 | R/W | sensor_cfg | 32:d0 | Register of scan controller containing the programmed bit map |

CONFIG_REG_1_7

| Address : 0x118 | | | | |
|------------------------|---------------|-------------------|----------------------|---|
| Bits | Access | Name | Default value | Description |
| 31:16 | R/W | wake_up_threshold | 16'd0 | Wakeup threshold |
| 15 | R/W | wakeup_mode | 1:d0 | Select bit for high/low mode 1 : Wakeup if count is greater than threshold 0 : Wakeup if count is lesser than threshold Wakeup mode will only work for average mode. |

Address : 0x118

| Bits | Access | Name | Default value | Description |
|-------------|---------------|-----------------|----------------------|---|
| 14:6 | R/W | ref_volt_config | 9'd0 | Configuration value of reference voltage |
| 5:4 | -- | Reserved | | |
| 3:0 | R/W | valid_sensors | 4'd0 | Value of number of sensors valid in the bit map |

CONFIG_REG_1_8

Address : 0x11C

| Bits | Access | Name | Default value | Description |
|-------------|---------------|-------------|----------------------|----------------------|
| 31:0 | R | prs_state | 32:d0 | Current state of PRS |

CONFIG_REG_1_9

Address : 0x120

| Bits | Access | Name | Default value | Description |
|-------------|---------------|-------------|----------------------|--|
| 31:10 | -- | Reserved | | |
| 9:0 | R/W | trig_div | 10:d0 | Allows one pulse for every ' <i>trig_div</i> ' no. of pulses of 1 ms clock |

IPMU_SPARE_REG2

| Bits | Interface | Access | Address | Default Value | Description |
|-------------|------------------|---------------|----------------|----------------------|---|
| 13 | SPI | R/W | 0x141 | 1 | Int cap input selection for cap sensor |
| 14 | SPI | R/W | 0x141 | 1 | Resistance input selection for cap sensor |

1352 . Register Description

20 Security Architecture

- General Description
- Features
- Register Summary
- Register Description

20.1 General Description

A rich set security features are provided to differentiate the end product.

20.2 Features

- GlobalPlatform[®] - Trusted Execution Environment compatible architecture with separate processor for secure applications.
- Hardware device identity and key storage with PUF based secure Roots-of-trust (RoT).
- True Random Number Generator.
- High Performance Security Accelerators: AES128/256, SHA256/384/512, RSA, ECC, ECDH, CRC
- FIPS140-2 certifiable
- Secure XIP from Flash with inline AES engine
- Secure Boot loading performed by the secure processor.
- Secure Firmware upgrade
- Tamper detection with Hardware disable, Secure RTC, Secure Hardware Watchdog and other Secure Peripherals connected to the Trusted Execution Environment.
- Programmable Secure Hardware Write protect for Flash sectors.

20.3 Register Summary

There are no registers in this section

20.4 Register Description

There are no registers in this section

21 In-System Programming (ISP)

21.1 General Description

In System Programming(ISP) is programming or reprogramming of the flash through boot loader. This can be done after the part is integrated on end user board.

21.2 Features

ISP can be done through any of the following interfaces

| Interface | Pins |
|-------------|-----------------------|
| UART | Rx: GPIO_8 Tx: GPIO_9 |
| USB | - |
| HSPI | GPIO_25 - GPIO_30 |
| SDIO | GPIO_25 - GPIO_30 |

21.3 Functional Description

Boot loader can be requested to boot in ISP mode by pulling down the GPIO_34 pin with 4.7K ohms of resistor. This pin has to be left unconnected during reset for the boot loader to bypass ISP and execute the code that is present in flash. ISP mode can be used to reprogram the flash, if the application codes uses JTAG pins for functional use. On boot up, if the application code goes into a state where JTAG interface is not functioning, ISP mode can be used to gain the control and to reprogram the flash.

21.4 Register Summary

There are no registers in this module

21.5 Register Description

There are no registers in this module

22 Boot Process and Bootloader

22.1 General Description

The Bootloader controls the initial operation of the device after any form of reset. The Bootloader supports Flash programming and initial startup of the application code. It also provides APIs to the application code for programming the Flash.

22.2 Features

- Two Bootloaders - Security Bootloader and Application Bootloader
- Support for ISP (In-System Programming) through multiple interfaces - UART, SPI, USB, USB-CDC and SDIO
- Auto-detection of ISP interface
- Support for secure boot
- Support for secure firmware upgrade using PUF based Roots-of-Trust (RoT)
- Anti-rollback protection
- Secure Key Management and Protection
- Support for different flash protection levels and write-protected Flash
- Secure XIP from Flash
- Support for multiple isolated images and selection
- Fail-proof migration of current active firmware to new (update) firmware
- Public key cryptography (digital signature) based authentication
- Provision to move to factory default firmware and keys

22.3 Functional Description

The Redpine MCU includes two Bootloaders - Security Bootloader and Application Bootloader. The Security Bootloader runs on the Security processor and the Application Bootloader runs on the Cortex M4 processor. On any reset, execution will always start in Security Bootloader, which is responsible for all security features, ISP and firmware upgradation. Once the Security Bootloader finishes its tasks, it enables the Application Bootloader. The Application Bootloader is responsible for transferring data to RAM from Flash and also for executing the wakeup sequence.

The following are the sources which can trigger the Bootloader:

- Primary reset (RESET_N_PAD)
- Power on reset (POC_IN)
- Watchdog reset
- Black out monitor
- Reset request through SYSRESETREQn bit in the Cortex-M4 processor

22.4 Secure Bootup

On reset, the Security Bootloader configures the module hardware based on the configuration present in the eFuse. It also passes the required information from the eFuse to the Application Bootloader. The Security Bootloader validates the integrity and authenticity of the firmware in the Flash and invokes the Application Bootloader. It detects and prevents execution of unauthorized software during the boot sequence. The Bootloader uses public & private key based digital signatures to recognize authentic software. The Security Bootloader provides provision for inline execution (XIP) of encrypted firmware from Flash. The Bootloader provides 3 flash protection levels which can be used to secure different sections of the Flash for different purposes:

- Protection level 1: Stored at manufacturing, not allowed to modify by the Security Bootloader
- Protection level 2: Allowed to modify by the Bootloader only, usually used to maintain secure information used/consumed by Bootloader
- Protection level 3: Allowed to modify by the Bootloader only, usually used to maintain protected firmware images.

The protection levels are written to Flash during the manufacturing process. The write-protection feature prevents the application program from changing the Flash protection levels. The Bootloader supports multiple isolated firmware and provision to select the firmware to execute on bootup.

The Security Bootloader is enabled or disabled during the manufacturing process.

22.5 Secure Firmware Upgrade (ISP)

The secure firmware upgrade feature of the Bootloader checks the authenticity of the new firmware image along with its integrity. The Bootloader supports the following interfaces to upgrade the firmware:

- UART
- SPI
- USB
- USB-CDC
- SDIO

The Bootloader automatically detects the host interface in use and configures the host interface hardware accordingly. The Bootloader updates the image only after successfully validating the authenticity and integrity of the image. It prevents downgrade to a lower version of firmware using the anti-rollback feature, if it is enabled. The Bootloader also supports transparent migration to a wirelessly updated image and protection against failures by providing recovery mechanisms.

22.6 Secure Key management

The Bootloader supports robust key management and secure key upgradation. The Bootloader's key management section maintains the following types of keys:

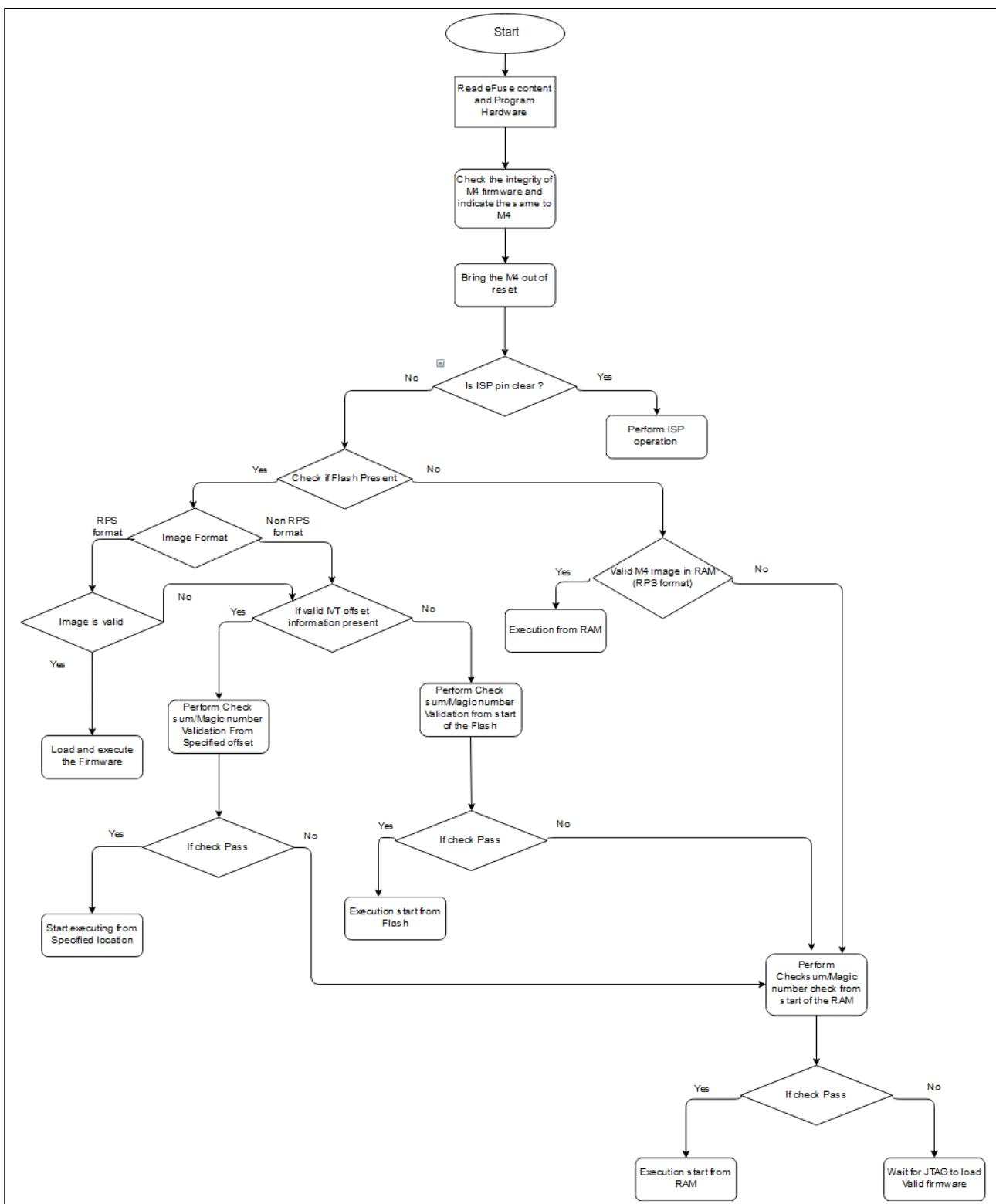
| Key Type | Description |
|----------------------|--|
| Master key | The Master key is used to validate the integrity of other keys and can be used to store the other keys in encrypted form. It can be a unique key generated using PUF, specific to a device to protect against cloning. |
| Firmware key | The Firmware key is used to validate the integrity of images and can be used to store the images in encrypted form. It can be a unique key generated using PUF, specific to a device to protect the firmware image |
| Firmware Upgrade key | The Bootloader uses the Firmware Upgrade key to decrypt (if encrypted) and check the integrity of new firmware image. |
| Public key | The Bootloader uses the Public key to validate the signature of the image during upgradation and secure bootup |
| PUF keys | The PUF keys are a table of keycode, which is used to retrieve the keys stored in PUF during the manufacturing process. The Master key and the Firmware key can be PUF keys. |
| User keys | The Bootloader provides provision to maintain keys used by the user application. |

22.7 Secure Zone

The Secure Zone provides a secure execution environment to store confidential data and to run secure applications. The Bootloader configures Secure Zone, secure firmware upgrade and secure bootup in "Secure Zone enabled" mode. This mode is programmed during the manufacturing process.

22.8 Bootloader Flowchart

The following diagram shows the top level flow for the Bootloader. For a more detailed description of the secure boot features, refer to the WiSeMCU_MCU_Security_Programmers_Reference.pdf. Please contact your Sales or Support/Applications contact at Redpine Signals, Inc., to obtain this document.



20 . Boot loader Flowchart

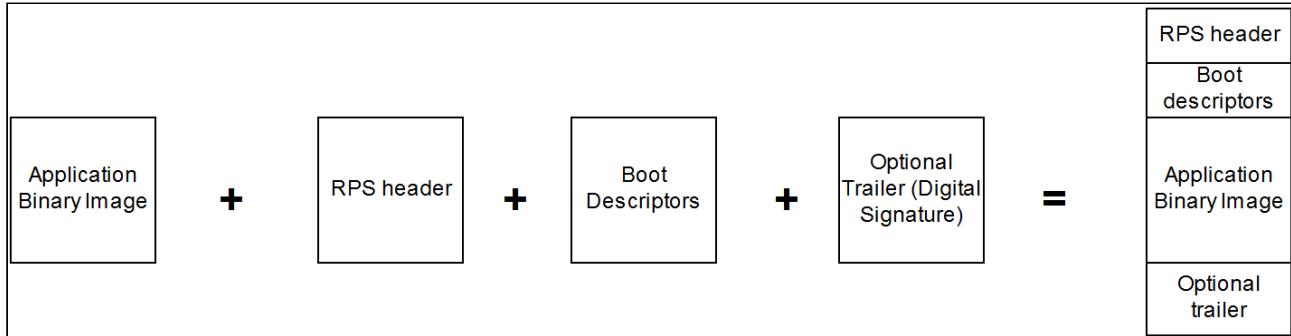
The Boot loader support two formats for the firmware image:

1. RPS format
2. Non RPS format

22.8.1 RPS Format

The RPS Format is a binary executable format understood by the Boot loader to perform the required integrity and authenticity checks and load and execute the application.

The Firmware Image in RPS format includes an RPS header, Boot descriptors, Application's binary image and an optional trailer (digital signature).



RPS Header:

This information is used by the Bootloader to process the configuration related to the firmware image.

| Field | Size | Description |
|---------------|--------|--|
| Control flags | 16 bit | <p>Bit map which Indicates image information</p> <p>BIT(0) :</p> <ul style="list-style-type: none"> 0 - NWP processor image 1 - MCU image <p>BIT(1):</p> <ul style="list-style-type: none"> 0 - Image is not encrypted 1 - Image is encrypted <p>BIT(2) :</p> <ul style="list-style-type: none"> 0 - CRC based integrity check 1 - MIC based integrity check <p>BIT(3) :</p> <ul style="list-style-type: none"> 0 - Digitally not signed 1 - Digitally Signed (in this case the Digital signature is attached in the Trailer section of the image) |

| Field | Size | Description |
|----------------------|---------|---|
| sha_type | 16 bit | Represents the SHA size used to compute the digest for the digital signature 1 - SHA_256 2 - SHA_384 3 - SHA_512 |
| Reserved | 32 bit | |
| image_size | 32 bit | Size of the image |
| fw_version | 32 bit | Firmware version number |
| flash_address | 32 bit | Address location in flash to store the image |
| crc | 32 bit | CRC of the image (polynomial to use can be decided at the time of manufacturing) |
| mic | 128 bit | MIC of the image |
| public key reference | 32 bit | Number to match with public key present in the device to validate the digital signature |
| Reserved | 16 bit | |

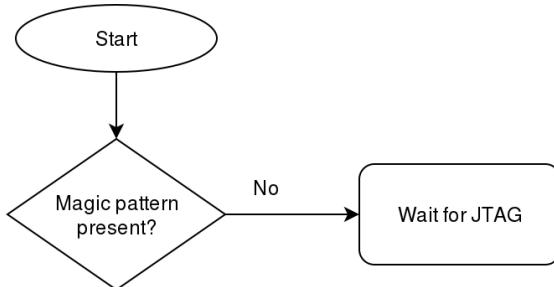
Boot Descriptors

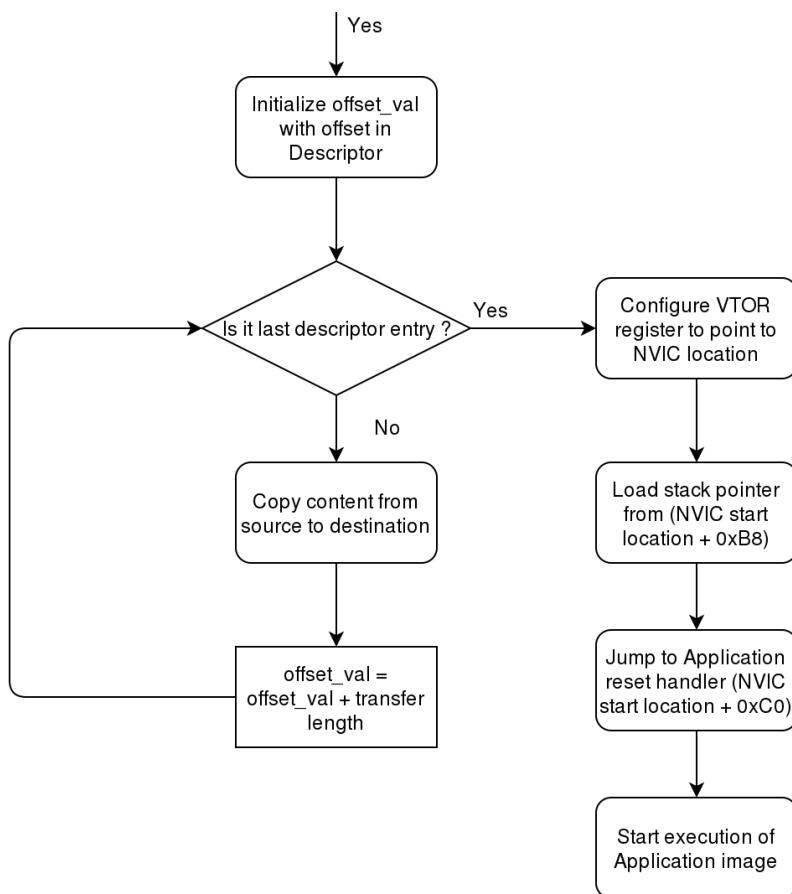
The Boot Descriptors are executed by the Boot loader to load the firmware. By using these boot descriptors, the application can instruct the Boot loader to load the image (usually initialized data section) to different parts of the on-chip RAM before start of execution.

| Field | Size | Description |
|------------------|----------|---|
| magic_pattern | 16 bit | Pattern for identification of a valid Flash content (0x5aa5) |
| offset | 16 bit | Offset of the binary image where the transfer should start from flash to RAM |
| IVT offset | 32 bit | Value to program VTOR register |
| bootload_entries | 56 bytes | Boot loader descriptor entries which are executed by the Boot loader while loading firmware - see table below for more details. |

Bootloader Descriptor Entries' Format:

| Field | Size | Description |
|------------|--------|--|
| length | 24 bit | Length of transfer to destination |
| reserved | 7 bit | |
| last entry | 1 bit | If set indicate it is last boot descriptor entry |
| dst_addr | 32 bit | destination address |





22.8.2 Non-RPS Format

The Non-RPS Format is a generic format where the application image is placed directly in Flash. In this format the Boot loader checks the validity of the image present in Flash using two approaches (before starting execution of the application image):

- Checksum based validation: The Boot loader computes the checksum of CSUM_OFFSET (0xec) locations of NVIC and compares the result with the checksum present at CSUM_OFFSET (0xec) location in NVIC table.
- Magic number based validation: The Boot loader checks whether the magic number (0x10AD10AD) is present at 0xEC location of NVIC table or not.

The approach to choose can be decided at the time of manufacturing.

22.9 Register Summary

There are no registers in this module.

22.10 Register Description

There are no registers in this module.

23 HRM Revision History

| Revision No. | Version No. | Date | Changes |
|--------------|-------------|----------------|---------------------|
| 1 | v0.9 | January 2018 | Preliminary version |
| 2 | v1.0 | September 2018 | Initial release |