RM0082
Reference manual

SPEAr300

Introduction

This reference manual provides complete hardware information for application developers of the SPEAr300 embedded MPU.

The SPEAr300 is a member of the SPEAr3xx family (includes SPEAr300, SPEAr310 and SPEAr320).

SPEAr3xx devices all feature ARM926EJ-S core running up to 333 MHz, an external DDR2 Memory Interface, a common set of powerful on-chip peripherals. Each member of the SPEAr3xx family has a specific set of IPs implemented in its Reconfigurable Array Subsystem (RAS). In the SPEAr300, the following IPs are implemented in the RAS.

- FSMC NAND/NOR Flash interface
- SDIO controller
- Color LCD controller (CLCD)
- Telecom IP with TDM interface, camera interface, I2S, 18 GPIOs (G8 and G10), DAC, SPI_I2C chip selects.
- Keyboard controller

For the pin out, ordering information, mechanical, electrical and timing characteristics, please refer to the SPEAr300 Datasheet.

For information on the ARM926EJ-S core, please refer to the ARM926EJ-S Technical Reference Manual.

Contents

1 Acronyms 52

2 Preface 54
2.1 Terms & conditions 54
2.2 Conventions 54
2.2.1 Numbering 54
2.2.2 Bits 54
2.2.3 Typographical 54

3 Reference documentation 55

4 Product overview 56
4.1 Device overview 56
4.1.1 Main features 57
4.2 Architecture properties 58
4.3 System architecture overview 59
4.3.1 Core architecture 59
4.4 CPU subsystem 60
4.5 Multilayer bus matrix 60
4.6 Dynamic memory controller 60
4.7 Basic subsystem 61
4.8 High speed connectivity subsystem 61
4.9 Low speed connectivity subsystem 61
4.10 Application subsystem 62
4.11 Reconfigurable logic array subsystem 62
4.12 Clock and reset system 63

5 Pin description 64
5.1 Required external components 64
5.2 Dedicated pins 64
5.3 Shared I/O pins (PL_GPIOs) 70
5.3.1 PL_GPIO pin description 70
5.3.2 Alternate functions 70

5.3.3 Boot pins . . . 70
5.3.4 GPIOs . . . 71
5.3.5 Multiplexing scheme . . . 71
5.4 PL_GPIO pin sharing for debug modes . . . 79

6 Memory map . . . 82

7 CPU subsystem_ARM926EJ-S . . . 85
7.1 Overview . . . 85
7.2 Functional description . . . 86
7.3 Main function description . . . 86
7.3.1 Memory management unit . . . 86
7.3.2 Caches and write buffer . . . 87
7.3.3 Bus interface unit . . . 87

8 CPU subsystem_Vectored interrupt controller (VIC) . . . 89
8.1 Overview . . . 89
8.2 Block diagram . . . 89
8.3 Main functions description . . . 90
8.3.1 Interrupt request logic . . . 90
8.3.2 Non-vectored FIQ interrupt logic . . . 90
8.3.3 Non-vectored IRQ interrupt logic . . . 91
8.3.4 Vectored interrupts . . . 91
8.3.5 Interrupt priority logic . . . 91
8.3.6 Software interrupts . . . 91
8.3.7 AHB slave interface . . . 91
8.4 Interrupt connection table . . . 92
8.5 How to reduce interrupt latency . . . 93
8.6 Programming model . . . 93
8.6.1 Register map . . . 93
8.6.2 Register description . . . 95
8.6.3 VICIRQSTATUS register . . . 95
8.6.4 VICFIQSTATUS register . . . 96
8.6.5 VICRAWINTR register . . . 96
8.6.6 VICINTSELECT register . . . 96
8.6.7 VICINTENABLE register . . . 97

8.6.8 VICINTENCLEAR register . . . 97
8.6.9 VICSOFTINT register . . . 97
8.6.10 VICSOFTINTCLEAR register . . . 98
8.6.11 VICPROTECTION register . . . 98
8.6.12 VICVECTADDR register . . . 98
8.6.13 VICDEFVECTADDR register . . . 99
8.6.14 VICVECTADDR register . . . 99
8.6.15 VICVECTCNTL register . . . 99
8.6.16 Peripheral identification registers . . . 99
8.6.17 VICPERIPHID0 register . . . 100
8.6.18 VICPERIPHID1 register . . . 100
8.6.19 VICPERIPHID2 register . . . 100
8.6.20 VICPERIPHID3 register . . . 101
8.6.21 Identification registers . . . 101
8.6.22 VICPCELLID0 register . . . 101
8.6.23 VICPCELLID1 register . . . 101
8.6.24 VICPCELLID2 register . . . 102
8.6.25 VICPCELLID3 register . . . 102

9 Bus interconnection matrix . . . 103
9.1 ICM . . . 104

10 DDR memory controller (MPMC) . . . 106
10.1 Overview . . . 106
10.2 Signal description . . . 106
10.3 Features overview . . . 108
10.4 Main block description . . . 109
10.4.1 AHB-Memory controller interfaces . . . 110
10.4.2 Arbiter . . . 118
10.4.3 Write data queue . . . 118
10.4.4 DRAM command processing . . . 119
10.4.5 Latency . . . 119
10.5 Multi-port arbiter . . . 120
10.5.1 Arbitration overview . . . 121
10.5.2 Understanding round-robin operation . . . 121
10.5.3 Understanding port priority . . . 122
10.5.4 Understanding relative priority . . . 122

10.5.5 Understanding port ordering . . . 124
10.5.6 Weighted round-robin arbitration summary . . . 125
10.5.7 Priority relaxing . . . 127
10.5.8 Port pairing . . . 129
10.5.9 Error conditions . . . 131
10.5.10 Command queue with placement logic . . . 131
10.6 Core command queue with placement logic . . . 132
10.6.1 Rules of the placement algorithm . . . 132
10.6.2 Command execution order after placement . . . 134
10.7 Low power operation . . . 135
10.7.1 Low power modes . . . 135
10.7.2 Low power mode control . . . 136
10.8 Additional features . . . 140
10.8.1 Out-of-range address checking . . . 140
10.8.2 Mobile devices DQS . . . 141
10.8.3 Half datapath option . . . 141
10.8.4 User-defined registers . . . 142
10.9 Address mapping . . . 142
10.9.1 DDR SDRAM address mapping options . . . 142
10.9.2 Maximum address space . . . 143
10.9.3 Memory mapping to address space . . . 143
10.10 DCC tuning timing . . . 144
10.11 External pin connection Of DDR interface in SPEAr300 . . . 145
10.12 Initialization protocol . . . 145
10.13 Register interface . . . 149
10.13.1 Register overview . . . 149
10.13.2 MPMC base address In SPEAr300 . . . 149
10.13.3 Register map . . . 149
10.13.4 Register description . . . 156
10.13.5 MEM0_CTL register . . . 156
10.13.6 MEM1_CTL register . . . 157
10.13.7 MEM2_CTL register . . . 157
10.13.8 MEM3_CTL register . . . 158
10.13.9 MEM4_CTL register . . . 158
10.13.10 MEM5_CTL register . . . 159
10.13.11 MEM6_CTL register . . . 159

10.13.12 MEM7_CTL register 160
10.13.13 MEM8_CTL register 160
10.13.14 MEM9_CTL register 161
10.13.15 MEM10_CTL register 161
10.13.16 MEM11_CTL register 162
10.13.17 MEM12_CTL register 162
10.13.18 MEM13_CTL register 163
10.13.19 MEM14_CTL register 163
10.13.20 MEM15_CTL register 164
10.13.21 MEM16_CTL register 164
10.13.22 MEM17_CTL register 164
10.13.23 MEM18_CTL register 165
10.13.24 MEM19_CTL register 165
10.13.25 MEM20_CTL register 166
10.13.26 MEM21_CTL register 166
10.13.27 MEM22_CTL register 167
10.13.28 MEM23_CTL register 167
10.13.29 MEM24_CTL register 168
10.13.30 MEM25_CTL register 168
10.13.31 MEM26_CTL register 169
10.13.32 MEM27_CTL register 169
10.13.33 MEM28_CTL register 170
10.13.34 MEM29_CTL register 170
10.13.35 MEM30_CTL register 171
10.13.36 MEM31_CTL/MEM32_CTL/MEM33_CTL register 171
10.13.37 MEM34_CTL register 171
10.13.38 MEM35_CTL register 172
10.13.39 MEM36_CTL register 172
10.13.40 MEM37_CTL register 172
10.13.41 MEM38_CTL register 173
10.13.42 MEM39_CTL register 173
10.13.43 MEM40_CTL register 174
10.13.44 MEM41_CTL register 174
10.13.45 MEM42_CTL register 174
10.13.46 MEM43_CTL register 175
10.13.47 MEM44_CTL register 175
10.13.48 MEM45_CTL register 175

10.13.49 MEM46_CTL register 176
10.13.50 MEM47_CTL register 176
10.13.51 MEM48_CTL register 176
10.13.52 MEM49_CTL register 177
10.13.53 MEM50_CTL register 177
10.13.54 MEM51_CTL register 177
10.13.55 MEM52_CTL/MEM53_CTL register 178
10.13.56 MEM54_CTL register 178
10.13.57 MEM55_CTL register 178
10.13.58 MEM56_CTL register 178
10.13.59 MEM57_CTL register 178
10.13.60 MEM58_CTL register 179
10.13.61 MEM59_CTL register 179
10.13.62 MEM60_CTL register 179
10.13.63 MEM61_CTL register 179
10.13.64 MEM62_CTL/MEM63_CTL/MEM64_CTL register 180
10.13.65 MEM65_CTL register 180
10.13.66 MEM66_CTL register 180
10.13.67 MEM67_CTL register 180
10.13.68 MEM68_CTL register 181
10.13.69 MEM[69-97]_CTL register 181
10.13.70 MEM[98-99]_CTL register 181
10.13.71 MEM100_CTL register 182
10.13.72 MEM101_CTL register 182
10.13.73 MEM102_CTL register 183
10.13.74 MEM103_CTL register 183
10.13.75 MEM104_CTL register 183
10.13.76 MEM105_CTL register 184
10.13.77 MEM106_CTL register 184
10.13.78 MEM107_CTL register 184
10.13.79 MEM108_CTL register 184
10.14 Summary of memory controller parameters 185

11 Clock & reset system 201

11.1 Clock generation scheme 202
11.1.1 Jitter at PLL output clock 202
11.2 Clock distribution scheme 203

11.2.1 Processor clock . . . 203
11.2.2 DDR controller clock . . . 203
11.2.3 Bus clocks . . . 204
11.2.4 Configurable logic clock . . . 204
11.2.5 Clock synthesizer . . . 205
11.2.6 Crystal connection . . . 207
11.3 RTC oscillator . . . 207
11.3.1 Crystal connection . . . 207

12 Miscellaneous registers (Misc) . . . 208
12.1 Signal description . . . 208
12.2 Overview features . . . 209
12.3 Register address map . . . 209
12.4 Miscellaneous register local space . . . 210
12.4.1 Overview . . . 210
12.4.2 Miscellaneous register local space address map . . . 210
12.4.3 SoC_CFG_CTR register . . . 213
12.4.4 DIAG_CFG_CTR register . . . 216
12.4.5 PLL 1/2_CTR registers . . . 218
12.4.6 PLL1/2_FRQ registers . . . 220
12.4.7 PLL1/2_MOD registers . . . 221
12.4.8 PLL_CLK_CFG register . . . 222
12.4.9 CORE_CLK_CFG register . . . 225
12.4.10 PRPH_CLK_CFG register . . . 226
12.4.11 PERIP1_CLK_ENB register . . . 228
12.4.12 RAS_CLK_ENB register . . . 230
12.4.13 PRSC1/2/3_CLK_CFG register . . . 231
12.4.14 AMEM_CFG_CTRL register . . . 232
12.4.15 Auxiliary clock synthesizer registers . . . 233
12.4.16 Soft reset control . . . 234
12.4.17 PERIP1_SOF_RST register . . . 234
12.4.18 RAS_SOF_RST register . . . 236
12.4.19 SoC configuration basic parameters . . . 238
12.4.20 ICM1-8_ARB_CFG register . . . 238
12.4.21 DMA_CHN_CFG register . . . 240
12.4.22 USB2_PHY_CFG register . . . 241
12.4.23 MAC_CFG_CTR register . . . 242

12.4.24 Special configuration parameters . . . 243
12.4.25 Powerdown_CFG_CTR register . . . 243
12.4.26 COMPSSTL_1V8_CFG/DDR_2V5_COMPENSATION register . . . 244
12.4.27 COMPCOR_3V3_CFG register . . . 244
12.4.28 DDR_PAD register . . . 245
12.4.29 Memory BIST execution control . . . 247
12.4.30 BIST1_CFG_CTR register . . . 247
12.4.31 BIST2_CFG_CTR register . . . 249
12.4.32 BIST3_CFG_CTR register . . . 250
12.4.33 BIST4_CFG_CTR register . . . 251
12.4.34 BIST1_STS_RES register . . . 252
12.4.35 BIST2_STS_RES register . . . 254
12.4.36 BIST3_STS_RES register . . . 255
12.4.37 BIST4_STS_RES register . . . 257
12.4.38 BIST5_RSLT_REG register (Reserved) . . . 258
12.4.39 Diagnostic functionality . . . 260
12.4.40 SYSERR_CFG_CTR register . . . 260
12.4.41 USB_TUN_PRM register . . . 262
12.4.42 PLGPIOn_PAD_PRG Registers . . . 262
12.5 Miscellaneous register global space . . . 267
12.5.1 Overview . . . 267
12.5.2 Miscellaneous register global space address map . . . 268
12.5.3 RAS1/2_GPP_INP register . . . 268
12.5.4 RAS1/2_GPP_OUT register . . . 269

13 LS_Synchronous serial peripheral (SSP) . . . 270
13.1 Overview . . . 270
13.2 Block diagram . . . 270
13.3 Signal interfaces . . . 271
13.4 Main functions description . . . 271
13.4.1 APB slave interface . . . 271
13.4.2 Register block . . . 272
13.4.3 Clock prescaler . . . 272
13.4.4 Transmit FIFO . . . 272
13.4.5 Receive FIFO . . . 272
13.4.6 Transmit and receive logic . . . 272
13.4.7 Interrupt generation logic . . . 273

13.4.8 DMA interface . . . 273
13.4.9 Synchronizing registers and logic . . . 273
13.5 SSP operation . . . 273
13.5.1 Configuring the SSP . . . 273
13.5.2 Enable SSP operation . . . 274
13.5.3 Programming the SSPCR0 control register . . . 275
13.5.4 Programming the SSPCR1 control register . . . 275
13.5.5 Frame format . . . 276
13.6 Programming model . . . 277
13.6.1 External pin connections . . . 277
13.6.2 Register map . . . 277
13.6.3 Register description . . . 278
13.6.4 SSPCR0 register . . . 278
13.6.5 SSPCR1 register . . . 279
13.6.6 SSPDR register . . . 279
13.6.7 SSPSR register . . . 280
13.6.8 SSPCPSR register . . . 280
13.6.9 SSPIMSC register . . . 281
13.6.10 SSPRIS register . . . 281
13.6.11 SSPMIS Register . . . 282
13.6.12 SSPICR register . . . 282
13.6.13 SSPDMACR register . . . 282
13.6.14 PHERIPHID0 register . . . 283
13.6.15 PHERIPHID1 register . . . 283
13.6.16 PHERIPHID2 register . . . 283
13.6.17 PHERIPHID3 register . . . 283
13.6.18 PCELLID0 register . . . 284
13.6.19 PCELLID1 register . . . 284
13.6.20 PCELLID2 register . . . 284
13.6.21 PCELLID3 register . . . 284
13.7 Interrupts . . . 284
13.7.1 SSPRXINTR . . . 285
13.7.2 SSPTXINTR . . . 285
13.7.3 SSPRORINTR . . . 285
13.7.4 SSPRTINTR . . . 285
13.7.5 SSPINTR . . . 285

14 BS_System controller 286
14.1 Overview 286
14.2 Block diagram 287
14.3 Main function description 287
14.3.1 System mode control 287
14.3.2 System control state machine 290
14.3.3 Interrupt response mode 291
14.3.4 Reset control 291
14.3.5 Core clock control 291
14.3.6 Watchdog module clock enable generation 291
14.4 Programming model 292
14.4.1 Register map 292
14.4.2 Register description 293
14.4.3 SCCTRL register 293
14.4.4 SCSYSSTAT register 294
14.4.5 SCIMCTRL register 295
14.4.6 SCIMSTAT register 295
14.4.7 SCXTALCTRL register 296
14.4.8 SCPLLCTRL register 296

15 BS_Serial memory interface 298
15.1 Overview 298
15.2 Block diagram 298
15.3 Main functions description 299
15.3.1 Clock prescaler 299
15.3.2 Data processing and control 299
15.4 Operation modes 300
15.4.1 Hardware mode 300
15.4.2 Software mode 301
15.5 Data transfers 301
15.5.1 Read request 301
15.5.2 Write request 302
15.5.3 Write burst mode 303
15.5.4 Read while write mode 303
15.5.5 Erase and write status register 304
15.6 Timings 304

15.6.1 Latencies . . . 304
15.7 How to boot from external memory . . . 305
15.8 Programming model . . . 306
15.8.1 External pin connection . . . 306
15.8.2 Register description . . . 306
15.8.3 SMI_CR1 register . . . 306
15.8.4 SMI_CR2 register . . . 308
15.8.5 SMI_SR register . . . 309
15.8.6 SMI_TR register . . . 311
15.8.7 SMI_RR register . . . 311

16 BS_Watchdog timer . . . 313
16.1 Overview . . . 313
16.2 Block diagram . . . 313
16.3 Main functions description . . . 314
16.3.1 AMBA APB interface . . . 314
16.3.2 Free running counter . . . 314
16.4 Clock signals . . . 314
16.5 Programming model . . . 315
16.5.1 Register map . . . 315
16.5.2 Register description . . . 316
16.5.3 WdogLoad register . . . 316
16.5.4 WdogValue register . . . 316
16.5.5 WdogControl register . . . 316
16.5.6 WdogIntClr register . . . 316
16.5.7 WdogRIS register . . . 317
16.5.8 WdogMIS register . . . 317
16.5.9 WdogLock register . . . 317

17 BS_General purpose timers . . . 318
17.1 Overview . . . 318
17.2 Programming model . . . 318
17.2.1 External pin connection . . . 318
17.2.2 Register map . . . 320
17.2.3 Register description . . . 321
17.2.4 Timer_control register . . . 321

17.2.5 TIMER_STATUS_INT_ACK register 322
17.2.6 TIMER_COMPARE register 323
17.2.7 TIMER_COUNT register 324
17.2.8 TIMER_REDG_CAPT register 324
17.2.9 TIMER_FEDG_CAPT register 324

18 BS_General purpose input/output (GPIO) 325
18.1 Overview 325
18.2 Functional description 325
18.2.1 Block diagram 325
18.2.2 Signal interfaces 325
18.3 Main functions description 326
18.3.1 APB slave interface 326
18.3.2 Interrupt detection logic 326
18.3.3 Mode control 327
18.4 How to 327
18.4.1 Read from and write to input/output lines 327
18.4.2 Control interrupt generation 327
18.5 Programming model 328
18.5.1 Register map 328
18.5.2 Register description 330
18.5.3 GPIODIR register 330
18.5.4 GPIODATA register 330
18.5.5 GPIOIS register 330
18.5.6 GPIOIBE register 331
18.5.7 GPIOIEV register 331
18.5.8 GPIOIE register 331
18.5.9 GPIORIS register 332
18.5.10 GPIOMIS register 332
18.5.11 GPIOIC register 332

19 BS_DMA controller 333
19.1 Overview 333
19.2 Block diagram 334
19.3 Signal interfaces 334
19.4 Main functions description 335

19.4.1 AHB slave interface . . . 335
19.4.2 AHB master interfaces . . . 335
19.4.3 DMA interface . . . 336
19.5 Scatter/gather . . . 336
19.5.1 How to program the DMAC for scatter/gather DMA . . . 337
19.6 Interrupt requests . . . 337
19.6.1 How to operate single combined DMACINTR interrupt request signal 338
19.7 Programming model . . . 338
19.7.1 Register map . . . 338
19.7.2 Register description . . . 340
19.7.3 DMACIntStatus register . . . 340
19.7.4 DMACIntTCStatus register . . . 340
19.7.5 DMACIntTCClear register . . . 340
19.7.6 DMACIntErrorStatus register . . . 341
19.7.7 DMACIntErrClr register . . . 341
19.7.8 DMACRawIntTCStatus register . . . 342
19.7.9 DMACRawIntErrorStatus register . . . 342
19.7.10 DMACEnbldChns register . . . 342
19.7.11 DMACSoftBReq register . . . 343
19.7.12 DMACSoftSReq register . . . 343
19.7.13 DMACSoftLBReq register . . . 343
19.7.14 DMACSoftLSReq register . . . 344
19.7.15 DMAC configuration register . . . 344
19.7.16 DMACSync register . . . 345
19.7.17 DMACCnSrcAddr register . . . 345
19.7.18 DMACCnDestAddr register . . . 346
19.7.19 DMACCnLLI register . . . 346
19.7.20 DMACCn control register . . . 347
19.7.21 DMAC Configuration register . . . 349
19.7.22 DMACPeriphID register . . . 351
19.7.23 DMACPCellID register . . . 351

20 BS_Real time clock . . . 352
20.1 Overview . . . 352
20.2 Programming model . . . 352
20.2.1 Register map . . . 352
20.2.2 Register description . . . 352

20.2.3 CONTROL register . . . 352
20.2.4 STATUS register . . . 353
20.2.5 TIME register . . . 354
20.2.6 DATE register . . . 355
20.2.7 ALARM TIME registers . . . 355
20.2.8 ALARM DATE registers . . . 356
20.2.9 REGxMC register . . . 356

21 AS_Cryptographic co-processor (C3) . . . 357
21.1 Overview . . . 357
21.2 Functional description . . . 358
21.2.1 Device summary . . . 358
21.3 Block diagram . . . 359
21.4 Main functions description . . . 359
21.4.1 HIF (High speed bus interface) . . . 360
21.4.2 SIF (Slave bus interface) . . . 360
21.4.3 IDS (Instruction dispatchers sub-system) . . . 360
21.4.4 Channel . . . 360
21.4.5 CCM (Coupling/Chaining module) . . . 361
21.5 Processing overview . . . 362
21.6 Programming model . . . 362
21.6.1 Register map . . . 362
21.6.2 System registers (C3_SYS) . . . 363
21.6.3 Register configuration . . . 364
21.6.4 Register description . . . 364
21.6.5 Master interface register (C3_HIF) . . . 368
21.6.6 Register configuration . . . 369
21.6.7 Register Description . . . 370
21.6.8 Memory page (HIF_MP) . . . 370
21.6.9 Memory size register (HIF_MSIZE) . . . 370
21.6.10 Memory base address register (HIF_MBAR) . . . 371
21.6.11 Memory control register (HIF_MCAR) . . . 371
21.6.12 Memory page base address register (HIF_MPBAR) . . . 373
21.6.13 Memory access address register (HIF_MAAR) . . . 374
21.6.14 Memory access data register (HIF_MADR) . . . 375
21.6.15 Byte bucket base address register (HIF_NBAR) . . . 375

21.6.16 Byte bucket control register (HIF_NCR) . . . 376
21.6.17 Instruction dispatcher registers (C3_IDn) . . . 377
21.6.18 Channel registers (C3_CHn) . . . 382
21.6.19 Channel ID register (CH_ID) . . . 382
21.7 Channel ID . . . 382
21.8 DES channel . . . 383
21.8.1 Overview . . . 383
21.8.2 Instruction set . . . 383
21.8.3 DES instructions . . . 383
21.8.4 DES START instruction . . . 383
21.8.5 ECB . . . 384
21.8.6 CBC . . . 384
21.8.7 DES APPEND instruction . . . 385
21.8.8 ECB . . . 385
21.8.9 CBC . . . 385
21.8.10 Register set . . . 386
21.8.11 DES register description . . . 386
21.8.12 Data input/output registers (DES_DATAINOUT) . . . 386
21.8.13 Feedback registers (DES_FEEDBACK) . . . 387
21.8.14 Control and status register (DES_CONTROL_STATUS) . . . 387
21.8.15 TKey registers (DES_KEY) . . . 388
21.8.16 Channel ID (DES_ID) . . . 388
21.9 AES channel . . . 388
21.9.1 Overview . . . 388
21.9.2 Instruction set . . . 388
21.9.3 AES instructions . . . 388
21.9.4 AES START instruction . . . 389
21.9.5 ECB . . . 389
21.9.6 CBC . . . 389
21.9.7 CTR . . . 390
21.9.8 AES APPEND instruction . . . 390
21.9.9 ECB . . . 390
21.9.10 CBC . . . 391
21.9.11 CTR . . . 391
21.10 Register set . . . 391
21.10.1 Register configuration . . . 391
21.10.2 Data input/output registers (AES_DATAIN_OUT) . . . 392

21.10.3 Feedback registers (AES_FEEDBACK) . . . 392
21.10.4 Counter registers (AES_COUNTER) . . . 393
21.10.5 Control and status register (AES_CONTROL_STATUS) . . . 393
21.10.6 Key registers (AES_KEY) . . . 395
21.10.7 Channel ID (AES_ID) . . . 395
21.11 Unified hash with HMAC channel . . . 395
21.11.1 Overview . . . 395
21.11.2 Instruction set . . . 395
21.11.3 HASH instruction . . . 395
21.11.4 HASH [MD5/SHA1] instructions . . . 395
21.11.5 INIT . . . 396
21.11.6 APPEND . . . 396
21.11.7 END . . . 396
21.11.8 HASH CONTEXT instruction . . . 397
21.11.9 SAVE . . . 397
21.11.10 RESTORE . . . 397
21.11.11 HMAC instruction . . . 398
21.11.12 HMAC [MD5/SHA1] instructions . . . 398
21.11.13 INIT . . . 398
21.11.14 APPEND . . . 398
21.11.15 END . . . 399
21.11.16 HMAC CONTEXT instruction . . . 399
21.11.17 SAVE . . . 399
21.11.18 RESTORE . . . 400
21.11.19 Register configuration . . . 400
21.11.20 Register description . . . 402
21.11.21 Control and status register (UHH_CU_CONTROL_STATUS) . . . 402
21.11.22 Data input register (UHH_DATA_IN) . . . 405
21.11.23 Control and status register (UHH_CB_CONTROL_STATUS) . . . 405
21.11.24 Channel ID (UHH_CH_ID) . . . 410

22 HS_USB2.0 host . . . 411
22.1 Overview . . . 411
22.2 Block diagram . . . 412
22.3 Main functions description . . . 412
22.3.1 AHB bus interface unit (BIU) . . . 412
22.3.2 EHCI host controller . . . 412

22.3.3 OHCI host controller 413
22.4 EHCI host controller blocks 413
22.4.1 List processor 413
22.4.2 Operational registers 413
22.4.3 Start-Of-Frame (SOF) generator 413
22.4.4 Packet buffer 413
22.4.5 Root hub 414
22.5 OHCI host controller blocks 414
22.5.1 HCI master block 415
22.5.2 HCI slave block 415
22.5.3 List processor block 416
22.5.4 RootHub and HSIE blocks 416
22.5.5 Digital PLL block (DPLL) 416
22.5.6 HSIE functionality 416
22.5.7 RootHub port configuration 417
22.6 Programming model 417
22.6.1 External pin connections 417
22.6.2 UHC interrupts 417
22.6.3 Register map 418
22.6.4 Register descriptions of EHCI 421
22.6.5 HCCAPBASE register 421
22.6.6 HCSPARAMS register 421
22.6.7 HCCPARAMS register 423
22.6.8 USBCMD register 424
22.6.9 USBSTS register 428
22.6.10 USBINTR register 430
22.6.11 FRINDEX register 431
22.6.12 CTRLDSSEGMENT register 432
22.6.13 PERIODICLISTBASE register 432
22.6.14 SYNCLISTADDR register 433
22.6.15 CONFIGFLAG register 433
22.6.16 PORTSC registers 434
22.6.17 INSNREG00 register 439
22.6.18 INSNREG01 register 439
22.6.19 INSNREG02 register 440
22.6.20 INSNREG03 register 440
22.6.21 INSNREG05 register 440

22.6.22 Register description of OHCI 440
22.6.23 Operation registers 440
22.6.24 The control and status partition 441
22.6.25 HcRevision register 441
22.6.26 HcControl register 441
22.6.27 HcCommandStatus register 443
22.6.28 HcInterruptStatus register 445
22.6.29 HcInterruptEnable register 447
22.6.30 HcInterruptDisable register 447
22.6.31 Memory pointer partition 448
22.6.32 HcHCCA register 448
22.6.33 HcPeriodCurrentED register 449
22.6.34 HcControlHeadED register 449
22.6.35 HcControlCurrentED register 449
22.6.36 HcBulkHeadED register 450
22.6.37 HcBulkCurrentED register 450
22.6.38 HcDoneHead register 451
22.6.39 Frame counter partition 451
22.6.40 HcFmInterval register 451
22.6.41 HcFmRemaining register 452
22.6.42 HcFmNumber register 453
22.6.43 HcPeriodicStart register 453
22.6.44 HcLSThreshold register 453
22.6.45 Root hub partition 454
22.6.46 HcRhDescriptorA register 454
22.6.47 HcRhDescriptorB register 456
22.6.48 HcRhStatus register 457
22.6.49 HcRhPortStatus[1:NDP] register 458

23 HS_USB 2.0 device 463

23.1 Overview 463
23.2 Block diagram 464
23.3 Main functions description 465
23.3.1 UTLI 465
23.3.2 Interrupt manager 465
23.3.3 SOF tracker 466
23.3.4 Receive FIFO controller 466

23.3.5 Endpoint FIFO controller (Transmit FIFO controller) . . . 467
23.3.6 Control and status registers . . . 468
23.3.7 AHB slave-only interface . . . 468
23.3.8 DMA (AHB master interface) . . . 468
23.3.9 DMA transfer engine . . . 468
23.3.10 DMA controller . . . 469
23.3.11 AHB interface . . . 469
23.3.12 CSRs slave access . . . 469
23.4 Theory of operation . . . 470
23.4.1 DMA mode . . . 470
23.4.2 In operation (Data transfer To USB host) . . . 471
23.4.3 Out operation (Data transfer from USB host) . . . 473
23.4.4 Slave-only mode . . . 474
23.5 Data memory structure in DMA mode . . . 478
23.5.1 SETUP data memory structure . . . 478
23.5.2 OUT data memory structure . . . 479
23.5.3 IN data memory structure . . . 481
23.6 Operation modes In DMA mode . . . 484
23.6.1 Packet-per-buffer mode . . . 484
23.6.2 Buffer fill mode (OUT) . . . 484
23.6.3 Buffer fill mode (IN) . . . 484
23.6.4 Threshold enable . . . 484
23.6.5 Burst split enable . . . 485
23.7 USB plug detect . . . 485
23.8 Programming model . . . 486
23.8.1 External pin connection . . . 486
23.8.2 Register map . . . 486
23.8.3 Register description . . . 490
23.8.4 Device configuration register . . . 490
23.8.5 Device control register . . . 492
23.8.6 Device status register . . . 494
23.8.7 Device interrupt register . . . 495
23.8.8 Device interrupt mask register . . . 497
23.8.9 Endpoint interrupt register . . . 497
23.8.10 Endpoint interrupt mask register . . . 497
23.8.11 Endpoint control register . . . 498

23.8.12 Endpoint status register 499
23.8.13 Endpoint buffer size and received packet frame number register 501
23.8.14 Endpoint maximum packet size and buffer size register 502
23.8.15 Endpoint setup buffer pointer register 503
23.8.16 Endpoint data description pointer register 503
23.8.17 UDC20 endpoint register 503

24 HS_Media independent interface (MII) 505

24.1 Overview 505
24.1.1 MAC core main features 505
24.1.2 MAC-AHB main features 505
24.2 Functional description 505
24.2.1 Block diagram 505
24.3 Main functions description 506
24.3.1 AHB slave interface 506
24.3.2 AHB master interface 506
24.3.3 DMA controller 507
24.3.4 Transmit and receive FIFO 507
24.3.5 MAC management counters 507
24.3.6 Power management module (PMT) 508
24.4 DMA descriptors 508
24.4.1 Transmit descriptors 509
24.4.2 Receive descriptors 513
24.5 How to initialize DMA 515
24.6 Interrupt management 516
24.7 Programming model 517
24.7.1 Register map 517
24.7.2 Register description 521
24.7.3 Bus mode register (Register0, DMA) 521
24.7.4 Transmit poll demand register (Register1, DMA) 523
24.7.5 Receive poll demand register (Register2, DMA) 523
24.7.6 Receive descriptor list address register (Register3, DMA) 523
24.7.7 Transmit descriptor list address register (Register4, DMA) 524
24.7.8 Status register (Register 5, DMA) 524
24.7.9 Operation mode register (Register 6, DMA) 529
24.7.10 Interrupt enable register (Register7, DMA) 532

24.7.11 Missed frame and buffer overflow counter register (Register8, DMA) . 533
24.7.12 Current host transmit descriptor register (Register18, DMA) 533
24.7.13 Current host receive descriptor register (Register19, DMA) 534
24.7.14 Current host transmit buffer address register (Register20, DMA) 534
24.7.15 Current host receive buffer address register (Register21, DMA) 534
24.7.16 MAC configuration register (Register0, MAC) 534
24.7.17 MAC frame filter register (Register1, MAC) 537
24.7.18 Hash table high register (Register2, MAC) 539
24.7.19 Hash table low register (Register3, MAC) 539
24.7.20 MII address register (Register4, MAC) 540
24.7.21 MII data register (Register5, MAC) 541
24.7.22 Flow control register (Register6, MAC) 541
24.7.23 VLAN tag register (Register7, MAC) 543
24.7.24 Wake-up frame filter register (Register10, MAC) 543
24.7.25 PMT control and status register (Register11, MAC) 544
24.7.26 Interrupt status register (Register 14, MAC) 545
24.7.27 Interrupt mask register (Register 15, MAC) 546
24.7.28 MAC address0 high register (Register16, MAC) 546
24.7.29 MAC address0 low register (Register17, MAC) 546
24.7.30 MAC address1 high register (Register18, MAC) 547
24.7.31 MAC address1 low register (Register19, MAC) 548
24.8 MMC registers 548
24.8.1 MMC control register 548
24.8.2 MMC receive interrupt register 549
24.8.3 MMC transmit interrupt register 550
24.8.4 MMC receive interrupt mask register 552
24.9 Clocks with MII 553

25 LS_JPEG codec 555
25.1 Overview 555
25.2 Signal interfaces 555
25.3 Functional description 556
25.3.1 Block diagram 556
25.3.2 Main functions description 557
25.3.3 Codec core 557
25.3.4 Codec controller 557
25.3.5 DMAC 557

25.3.6 FIFO buffers . . . 558
25.3.7 Internal memories . . . 558
25.4 Programming model . . . 558
25.4.1 Register map . . . 558
25.4.2 Register description . . . 560
25.4.3 JPGCreg0 register . . . 560
25.4.4 JPGCReg1 register . . . 560
25.4.5 JPGCreg2 register . . . 561
25.4.6 JPGCreg3 register . . . 562
25.4.7 JPGCreg4-7 register . . . 562
25.4.8 JPGC control status register . . . 563
25.4.9 JPGC bytes from Fifo to core register . . . 564
25.4.10 JPGC bytes from core to Fifo register . . . 565
25.4.11 JPGC bust count beforeInit . . . 565
25.4.12 DMAC registers . . . 565
25.4.13 JPGCFifoIn register . . . 565
25.4.14 JPGCFifoOut register . . . 566
25.4.15 JPGCqmem memory . . . 566
25.4.16 JPGChuffmin memory . . . 567
25.4.17 JPGC huffbase memory . . . 567
25.4.18 JPGChuffsymb memory . . . 567
25.4.19 JPGCDHTmem memory . . . 568
25.4.20 JPGChuffenc memory . . . 568

26 LS_Fast IrDA controller . . . 571
26.1 Overview . . . 571
26.2 Block diagram . . . 571
26.3 Main functions description . . . 572
26.3.1 Synchronization unit . . . 572
26.3.2 Demodulation unit . . . 573
26.3.3 Wrapper unit . . . 573
26.3.4 Modulation unit . . . 574
26.3.5 Baud rate generation unit . . . 574
26.3.6 FIFO unit . . . 575
26.4 Interrupt sources . . . 577
26.5 Programming model . . . 578

26.5.1 External pin connection . . . 578
26.5.2 Register map . . . 578
26.5.3 Register description . . . 579
26.5.4 IrDA_CON register . . . 579
26.5.5 IrDA_CONF register . . . 579
26.5.6 IrDA_PARA register . . . 580
26.5.7 IrDA_DV register . . . 581
26.5.8 IrDA_STAT register . . . 582
26.5.9 IrDA_TFS register . . . 582
26.5.10 IrDA_RFS register . . . 583
26.5.11 IrDA_TXB register . . . 583
26.5.12 IrDA_RXB register . . . 584
26.5.13 IrDA_IMSC register . . . 584
26.5.14 IrDA_RIS register . . . 584
26.5.15 IrDA_MIS register . . . 585
26.5.16 IrDA_ICR register . . . 586
26.5.17 IrDA_ISR register . . . 587
26.5.18 IrDA_DMA register . . . 587

27 LS_Universal asynchronous receiver/transmitter (UART) . . . 589

27.1 Overview . . . 589
27.2 Functional description . . . 590
27.2.1 Block diagram . . . 590
27.2.2 Main functions description . . . 590
27.3 Programming model . . . 594
27.3.1 Register map . . . 594
27.4 Register description . . . 595
27.4.1 UARTDR register . . . 595
27.4.2 UARTRSR/UARTECR register . . . 596
27.4.3 UARTFR register . . . 597
27.4.4 UARTIBRD register . . . 598
27.4.5 UARTFBRD register . . . 598
27.4.6 UARTLCR_H register . . . 599
27.4.7 UARTCR register . . . 601
27.4.8 UARTIFLS register . . . 602
27.4.9 UARTIMSC register . . . 603
27.4.10 UARTRIS register . . . 604

27.4.11 UARTMIS Register 604
27.4.12 UARTICR register 605
27.4.13 UARTDMACR register 606
27.5 UART modem operation 606

28 LS_I2C controller 607
28.1 Overview 607
28.2 Block diagram 607
28.3 Main functions description 608
28.3.1 APB interface 608
28.3.2 I^2C protocols 608
28.3.3 DMA controller interface 611
28.4 Operation modes 612
28.4.1 Slave mode 612
28.4.2 Master mode 614
28.4.3 Multi-master mode 615
28.5 Interrupt sources 617
28.6 Programming model 619
28.6.1 External pin connections 619
28.6.2 Register map 619
28.6.3 IC_CON register(0x000) 621
28.6.4 IC_TAR register(0x004) 623
28.6.5 IC_SAR register(0x008) 624
28.6.6 IC_HS_MADDR register(0x00C) 625
28.6.7 IC_DATA_CMD register(0x010) 625
28.6.8 IC_SS_SCL_HCNT register (0x014) 626
28.6.9 IC_SS_SCL_LCNT register(0x018) 627
28.6.10 IC_FS_SCL_HCNT register(0x01C) 628
28.6.11 IC_FS_SCL_LCNT register(0x020) 629
28.6.12 IC_HS_SCL_HCNT register(0x024) 629
28.6.13 IC_HS_SCL_LCNT register(0x028) 630
28.6.14 IC_INTR_STAT register(0x02C) 631
28.6.15 IC_INTR_MASK register(0x030) 632
28.6.16 IC_RAW_INTR_STAT register(0x034) 633
28.6.17 IC_RX_TL register(0x038) 633
28.6.18 IC_TX_TL register(0x03C) 634

28.6.19 IC_CLR_INTR register(0x040) . . . 634
28.6.20 Interrupt clearing registers(0x044 - 0x068) . . . 635
28.6.21 IC_ENABLE register(0x06C) . . . 635
28.6.22 IC_STATUS register(0x070) . . . 636
28.6.23 IC_TXFLR and IC_RXFLR registers (0x074 - 0x078) . . . 637
28.6.24 IC_TX_ABRT_SOURCE register (0x080) . . . 637
28.6.25 IC_DMA_CR register (0x088) . . . 639
28.6.26 IC_DMA_TDLR register (0x08C) . . . 640
28.6.27 IC_DMA_RDLR register (0x090) . . . 640
28.6.28 IC_COMP_PARAM1 register (0x0F4) . . . 641

29 LS_Analog to digital convertor (ADC) . . . 643

29.1 Overview . . . 643
29.2 Functional description . . . 643
29.3 Operating sequence . . . 644
29.3.1 Normal mode . . . 644
29.3.2 Enhanced mode . . . 644
29.4 Programming model . . . 645
29.4.1 External pin connection . . . 645
29.5 Register description . . . 646
29.5.1 ADC_STATUS_REG register . . . 646
29.5.2 AVERAGE_REG register . . . 648
29.5.3 SCAN RATE register . . . 649
29.5.4 ADC_CLK_REG register . . . 649
29.5.5 CHx CTRL register . . . 650
29.5.6 CHx DATA register . . . 650

30 RS_Reconfigurable array subsystem (RAS) registers . . . 652

30.1 RAS configurations . . . 652
30.1.1 NAND mode . . . 652
30.1.2 NOR mode . . . 653
30.1.3 Photoframe mode . . . 653
30.1.4 LEND_IP_PHONE (LOW END IP PHONE mode) . . . 653
30.1.5 HEND_IP_PHONE MODE (HIGH END IP PHONE) . . . 654
30.1.6 LEND_WIFI_PHONE MODE (LOW END WI-FI PHONE) . . . 654
30.1.7 HEND_WIFI_PHONE MODE (HIGH END WI-FI PHONE) . . . 654
30.1.8 ATA_PABX_wI2S (ATA PABX without I2S) . . . 654

30.1.9 ATA_PABX_I2S MODE (ATA PABX with I2S) 655
30.1.10 CAMl_LCDw MODE (8 bit CAMERA without LCD) 655
30.1.11 CAMu_LCD MODE (14 bit CAMERA with LCD) 655
30.1.12 CAMu_wLCD MODE (14 bit CAMERA without LCD) 656
30.1.13 CAMl_LCD MODE (8 bit CAMERA with LCD) 656
30.2 Maximum number of GPIOs 656
30.3 Address map 656
30.4 RAS configuration registers 657
30.4.1 RAS Register 1 (0x99000000) 658
30.4.2 RAS Register 2 (0x99000004) 659
30.5 RAS Interrupt assignments 660
30.6 RAS DMA configuration 661

31 RS_Flexible static memory controller (FSMC) 662
31.1 Overview 662
31.2 Functional description 663
31.2.1 Block diagram 663
31.3 Description 663
31.3.1 AHB interface 663
31.3.2 NAND flash controller 663
31.3.3 Asynchronous SRAM and NOR parallel Flash controller 664
31.4 Programming model 664
31.4.1 External signals 664
31.4.2 Address map 665
31.4.3 Register map 665
31.4.4 Register description 667
31.4.5 GenMemCtrl(i) registers 667
31.4.6 GenMemCtrl_tim(i) registers 669
31.4.7 GenMemCtrl_PC(i) registers 670
31.4.8 GenMemCtrl_Comm(i)/GenMemCtrl_Attrib(i) registers 671
31.4.9 GenMemCtrl_ECCr(i) registers 671
31.4.10 GenMemCtrl peripheral identification registers (GenMemCtrlPeripID0-3) . 672
31.4.11 GenMemCtrl cell Identification registers (GenMemCtrlPCellID0-3) . . . 672
31.4.12 Calculating the FSMC timing parameters 673

32 RS_SDIO controller 678
32.1 Overview 678
32.2 Main features 678
32.3 Signal interface 679
32.4 Pin signals 680
32.5 Functional overview 683
32.5.1 Sequence 684
32.6 Programmer's model 695
32.6.1 Register map 695
32.7 Register description 697
32.7.1 SDMASysAddr register 697
32.7.2 BLKSize register 697
32.7.3 BLKCount register 698
32.7.4 CMDARG register 699
32.7.5 TRMode register 699
32.7.6 CMD register 701
32.7.7 RESP(i) registers 702
32.7.8 Buf data port register 703
32.7.9 PRSTATE register 703
32.7.10 HOSTCTRL register 707
32.7.11 PWRCTL register 708
32.7.12 BLKGAPCTRL register 708
32.7.13 WKUPCTRL register 710
32.7.14 CLKCTRL register 711
32.7.15 TMOUTCTRL register 713
32.7.16 SWRES register 714
32.7.17 NIRQSTAT register 715
32.7.18 ERRIRQSTAT register 718
32.7.19 NIRQSTATEN register 720
32.7.20 ERRIRQSTATEN register 721
32.7.21 NIRQSIGEN register 722
32.7.22 ERRIRQSIGEN register 722
32.7.23 ACMD12ERSTS register 723
32.7.24 CAP1 register 725
32.7.25 CAP2 register 727
32.7.26 MAXCURR1 register 727

32.7.27 MAXCURR2 register 728
32.7.28 ACMD12FEERSTS register 728
32.7.29 FEERRINTSTS register 729
32.7.30 ADMAERRSTS register 730
32.7.31 ADMAADDR1 register 731
32.7.32 ADMAADDR2 register 731
32.7.33 SPIIRQSUPP register 732
32.7.34 SLTIRQSTS register 733
32.7.35 HCTRLVER register 733

33 RS_Color liquid crystal display controller (CLCD) 735

33.1 Overview 735
33.1.1 Number of colors supported 736
33.2 Block diagram 737
33.3 Signal interfaces 738
33.4 LCD panel signal multiplexing details 738
33.5 Main functions description 740
33.5.1 AHB slave interface 740
33.5.2 AHB master interface 740
33.5.3 Dual DMA FIFOs and associated control logic 741
33.5.4 Pixel serializer 741
33.5.5 RAM palette 744
33.5.6 Gray scaler 745
33.5.7 Upper and lower panel formatters 745
33.5.8 Panel clock generator 746
33.5.9 Timing controller 746
33.5.10 Interrupt generation 746
33.5.11 Bus architecture 747
33.5.12 LCD powering up and down sequences 747
33.6 Programming model 748
33.6.1 External pin connection 748
33.6.2 Register map 748
33.6.3 Register description 749
33.6.4 LCD timing 0 register 749
33.6.5 LCD timing 1 register 751
33.6.6 LCD timing 2 register 752

33.6.7 LCD timing 3 register . . . 754
33.6.8 LCDUPBASE and LCPLPBASE registers . . . 754
33.6.9 LCDIMSC register . . . 755
33.6.10 LCD control register . . . 756
33.6.11 LCDRIS register . . . 757
33.6.12 LCDMIS register . . . 758
33.6.13 LCDICR register . . . 758
33.6.14 LCDUPCURR and LCDLPCURR registers . . . 759
33.6.15 LCDPalette register . . . 759
33.6.16 PHERIPHID0-3 registers . . . 760
33.6.17 PCELLIDID0-3 registers . . . 761
33.7 Interrupts . . . 762
33.7.1 CLCDMBEINTR . . . 762
33.7.2 LCDVCOMPINTR . . . 762
33.7.3 CLCDLNBUINTR . . . 762
33.7.4 CLCDFUFINTR . . . 763
33.7.5 LCD powering up and powering down sequence support . . . 763
33.8 CLCD clock scheme . . . 764

34 RS_Telecom IP . . . 765
34.1 Overview . . . 765
34.2 Main features . . . 765
34.3 Pin list . . . 765
34.4 Functional overview . . . 766
34.4.1 Regs and regs_rw blocks . . . 767
34.4.2 TDM clock block . . . 767
34.4.3 TDM synchro block . . . 768
34.4.4 TDM block . . . 769
34.4.5 Action memory . . . 770
34.4.6 Switching memory . . . 770
34.4.7 Buffer memory . . . 772
34.4.8 I2S block . . . 777
34.4.9 DAC block . . . 779
34.4.10 Camera interface . . . 780
34.4.11 SPI-I2C block . . . 781
34.4.12 General purpose GPIOs G8 and G10 . . . 782

34.4.13 IT bus . . . 782
34.5 Programmer's model . . . 782
34.5.1 Address map . . . 783
34.5.2 Register set . . . 783
34.6 Description of registers . . . 784
34.6.1 Boot register . . . 784
34.6.2 TDM_conf register . . . 784
34.6.3 GPIO8_DIR register . . . 787
34.6.4 GPIO10_DIR register . . . 788
34.6.5 GPIO8_out register . . . 788
34.6.6 GPIO10_out register . . . 788
34.6.7 GPIO8_in . . . 789
34.6.8 GPIO10_in register . . . 789
34.6.9 IT-GEN register . . . 790
34.6.10 GPIOt register . . . 791
34.6.11 GPIOtt register . . . 792
34.6.12 PERS_time register . . . 792
34.6.13 PERS_data register . . . 792
34.6.14 TDM_timeslot_NBR register . . . 792
34.6.15 TDM_frame_NBR register . . . 793
34.6.16 TDM_SYNC_GEN register . . . 793
34.6.17 SPI_I2C_usage register . . . 798
34.6.18 SPI_I2C_active . . . 798
34.6.19 I2S_CONF register . . . 798
34.6.20 I2S_CONF2 register . . . 801
34.6.21 I2S_CLK_CONF register . . . 802
34.6.22 Interrupt mask register . . . 804
34.6.23 Interrupt status register . . . 805
34.7 Action memory content description . . . 806
34.7.1 Int block . . . 808

35 RS_Keyboard controller . . . 810
35.1 Overview . . . 810
35.2 Functional description . . . 810
35.2.1 General purpose input output interface . . . 810
35.2.2 Keyboard interface . . . 811

35.3 Programming model 811
35.3.1 External signals 811
35.3.2 Register map 812
35.3.3 Registers description 812

36 RS_General Purpose Input Output (GPIO) 815
36.1 Application Notes 815

37 Power and clock management 816
37.1 Overview 816
37.1.1 Power management techniques 816
37.2 System control state machine 817
37.2.1 SLEEP 818
37.2.2 DOZE (reset state) 819
37.2.3 SLOW 819
37.2.4 NORMAL 819
37.3 Dynamic frequency scaling 820
37.4 Dynamic clock switching 820
37.5 Combining frequency scaling and clock switching techniques 821
37.6 Statiscally frequency selection and clock switching OFF 821
37.7 PLLs usage 821
37.8 Power consumption 822
37.8.1 Modules power consumption 824
37.8.2 IPs power 825

38 BootROM 826
38.1 Boot Stages 826
38.2 Booting Pins 827
38.3 Hardware overview 828
38.3.1 eROM (Embedded ROM) 828
38.3.2 Shadow memory 828
38.3.3 System controller 828
38.4 Software overview 829
38.4.1 ARM processor modes 829
38.4.2 SoC peripheral interrupts 829
38.4.3 Memory Overview 829

38.4.4 X-Loader and U-boot Header 829
38.4.5 X-Loader and U-boot authentication 829
38.5 Boot flows 830
38.5.1 Serial NOR Flash boot 831
38.5.2 NAND Flash boot 832
38.5.3 USB boot 834
38.5.4 Serial (UART) Boot 839
38.5.5 Ethernet boot 841

39 Document revision history 843

List of Tables

Table 1. Acronyms . . . 52
Table 2. Typographical conventions . . . 54
Table 3. Master clock, RTC, Reset and 3.3 V comparator pin descriptions . . . 64
Table 4. Power supply pin description . . . 64
Table 5. Debug pin descriptions . . . 65
Table 6. Serial memory interface (SMI) pin description . . . 66
Table 7. USB pin descriptions . . . 66
Table 8. ADC pin description . . . 67
Table 9. DDR pin description . . . 68
Table 10. PL_GPIO pin description . . . 70
Table 11. Booting pins . . . 71
Table 12. Available peripherals in each configuration mode . . . 73
Table 13. PL_GPIO multiplexing scheme . . . 74
Table 14. Table shading . . . 79
Table 15. Ball sharing during debug . . . 80
Table 16. Main memory map . . . 82
Table 17. Multi layer CPU subsystem . . . 82
Table 18. Low speed subsystem . . . 82
Table 19. Basic subsystem . . . 83
Table 20. High speed subsystem . . . 83
Table 21. Application subsystem . . . 83
Table 22. Reconfigurable array subsystem . . . 84
Table 23. Interrupt sources . . . 92
Table 24. Interrupt latency for different types of interrupts . . . 93
Table 25. VIC interrupt control registers summary . . . 93
Table 26. VIC vector address registers summary . . . 94
Table 27. VIC interrupt vector control registers summary . . . 95
Table 28. VIC identification registers summary . . . 95
Table 29. VICIRQSTATUS register bit assignments . . . 96
Table 30. VICFIQSTATUS register bit assignments . . . 96
Table 31. VICRAWINTR register bit assignments . . . 96
Table 32. VICINTSELECT register bit assignments . . . 97
Table 33. VICINTENABLE register bit assignments . . . 97
Table 34. VICINTENCLEAR register bit assignments . . . 97
Table 35. VICSOFTINT register bit assignments . . . 98
Table 36. VICSOFTINTCLEAR register bit assignments . . . 98
Table 37. VICPROTECTION register bit assignments . . . 98
Table 38. VICVECTADDR register bit assignments . . . 99
Table 39. VICVECTCNTL registers bit assignments . . . 99
Table 40. Peripheral identification registers bit assignments . . . 100
Table 41. VICPERIPHID0 register bit assignments . . . 100
Table 42. VICPERIPHID1 register bit assignments . . . 100
Table 43. VICPERIPHID2 register bit assignments . . . 101
Table 44. VICPERIPHID3 register bit assignments . . . 101
Table 45. VICPCELLID0 register bit assignments . . . 101
Table 46. VICPCELLID1 register bit assignments . . . 101
Table 47. VICPCELLID2 register bit assignments . . . 102
Table 48. VICPCELLID3 register bit assignments . . . 102

Table 49. SoC interconnection matrix scheme . . . 103
Table 50. SoC interconnection matrix. . . . 103
Table 51. ICM master layers (Initiator) . . . 104
Table 52. ICM slaves (Targets) . . . 105
Table 53. External memory Interface signals . . . 106
Table 54. Internal signals . . . 107
Table 55. Configured AHB settings . . . 112
Table 56. READ/WRITE data alignment - Little Endian . . . 115
Table 57. READ/WRITE data alignment - Big Endian . . . 116
Table 58. AHB-Memory controller translation example . . . 117
Table 59. Round-Robin operation example . . . 122
Table 60. Relative priority example . . . 124
Table 61. Port ordering example . . . 125
Table 62. System D specifications . . . 125
Table 63. System D operation . . . 126
Table 64. System E specifications . . . 126
Table 65. System E operation . . . 127
Table 66. System F specifications . . . 128
Table 67. System F operation with priority relaxing. . . . 128
Table 68. System G specifications . . . 130
Table 69. System G operation . . . 130
Table 70. Low power mode parameters . . . 138
Table 71. Low power mode controls. . . . 139
Table 72. Memory interface buses with Half Datapath option . . . 141
Table 73. Delay parameters . . . 144
Table 74. MT47H128M8-3 (DDR2@333 MHz cl5) initialization table. . . . 146
Table 75. MT46H6M16LF-6(Low Power DDR @166 MHz cl3) initialization table . . . 147
Table 76. Parameter size to mapping conditions . . . 149
Table 77. Registers overview . . . 150
Table 78. MEM0_CTL register bit assignments . . . 156
Table 79. MEM1_CTL register bit assignments . . . 157
Table 80. MEM2_CTL register bit assignments . . . 157
Table 81. MEM3_CTL register bit assignments . . . 158
Table 82. MEM4_CTL register bit assignments . . . 158
Table 83. MEM5_CTL register bit assignments . . . 159
Table 84. MEM6_CTL register bit assignments . . . 159
Table 85. MEM7_CTL register bit assignments . . . 160
Table 86. MEM8_CTL register bit assignments . . . 160
Table 87. MEM9_CTL register bit assignments . . . 161
Table 88. MEM10_CTL register bit assignments . . . 161
Table 89. MEM11_CTL register bit assignments . . . 162
Table 90. MEM12_CTL register bit assignments . . . 162
Table 91. MEM13_CTL register bit assignments . . . 163
Table 92. MEM14_CTL register bit assignments . . . 163
Table 93. MEM15_CTL register bit assignments . . . 164
Table 94. MEM16_CTL register bit assignments . . . 164
Table 95. MEM17_CTL register bit assignments . . . 164
Table 96. MEM18_CTL register bit assignments . . . 165
Table 97. MEM19_CTL register bit assignments . . . 165
Table 98. MEM20_CTL register bit assignments . . . 166
Table 99. MEM21_CTL register bit assignments . . . 166
Table 100. MEM22_CTL register bit assignments . . . 167

Table 101. MEM23_CTL register bit assignments . . . 167
Table 102. MEM24_CTL register bit assignments . . . 168
Table 103. MEM25_CTL register bit assignments . . . 168
Table 104. MEM26_CTL register bit assignments . . . 169
Table 105. MEM27_CTL register bit assignments . . . 169
Table 106. MEM28_CTL register bit assignments . . . 170
Table 107. MEM29_CTL register bit assignments . . . 170
Table 108. MEM30_CTL register bit assignments . . . 171
Table 109. MEM31_CTL/MEM32_CTL/MEM33_CTL register bit assignments . . . 171
Table 110. MEM34_CTL register bit assignments . . . 171
Table 111. MEM35_CTL register bit assignments . . . 172
Table 112. MEM36_CTL register bit assignments . . . 172
Table 113. MEM37_CTL register bit assignments . . . 172
Table 114. MEM38_CTL register bit assignments . . . 173
Table 115. MEM39_CTL register bit assignments . . . 173
Table 116. MEM40_CTL register bit assignments . . . 174
Table 117. MEM41_CTL register bit assignments . . . 174
Table 118. MEM42_CTL register bit assignments . . . 174
Table 119. MEM43_CTL register bit assignments . . . 175
Table 120. MEM44_CTL register bit assignments . . . 175
Table 121. MEM45_CTL register bit assignments . . . 175
Table 122. MEM46_CTL register bit assignments . . . 176
Table 123. MEM47_CTL register bit assignments . . . 176
Table 124. MEM48_CTL register bit assignments . . . 176
Table 125. MEM49_CTL register bit assignments . . . 177
Table 126. MEM50_CTL register bit assignments . . . 177
Table 127. MEM51_CTL register bit assignments . . . 177
Table 128. MEM52_CTL/MEM53_CTL register bit assignments . . . 178
Table 129. MEM54_CTL register bit assignments . . . 178
Table 130. MEM55_CTL register bit assignments . . . 178
Table 131. MEM56_CTL register bit assignments . . . 178
Table 132. MEM57_CTL register bit assignments . . . 178
Table 133. MEM58_CTL register bit assignments . . . 179
Table 134. MEM59_CTL register bit assignments . . . 179
Table 135. MEM60_CTL register bit assignments . . . 179
Table 136. MEM61_CTL register bit assignments . . . 179
Table 137. MEM62_CTL/MEM63_CTL/MEM64_CTL register bit assignments . . . 180
Table 138. MEM65_CTL register bit assignments . . . 180
Table 139. MEM66_CTL register bit assignments . . . 180
Table 140. MEM67_CTL register bit assignments . . . 180
Table 141. MEM68_CTL register bit assignments . . . 181
Table 142. MEM[69-97]_CTL register bit assignments . . . 181
Table 143. MEM[98-99]_CTL register bit assignments . . . 181
Table 144. MEM100_CTL register bit assignments . . . 182
Table 145. MEM101_CTL register bit assignments . . . 182
Table 146. MEM102_CTL register bit assignments . . . 183
Table 147. MEM103_CTL register bit assignments . . . 183
Table 148. MEM104_CTL register bit assignments . . . 183
Table 149. MEM105_CTL register bit assignments . . . 184
Table 150. MEM106_CTL register bit assignments . . . 184
Table 151. MEM107_CTL register bit assignments . . . 184
Table 152. MEM108_CTL register bit assignments . . . 184

Table 153. Memory controller parameters . . . 185
Table 154. Jitter at PLL output clock . . . 202
Table 155. APB interface signals . . . 208
Table 156. Miscellaneous register main memory map . . . 209
Table 157. Miscellaneous local space registers overview . . . 210
Table 158. SoC_CFG_CTR register bit assignments . . . 213
Table 159. DIAG_CFG_CTR register bit assignments . . . 216
Table 160. PLL 1/2_CTR register bit assignments . . . 219
Table 161. PLL1/2_FRQ register bit assignments . . . 220
Table 162. PLL1/2_MOD register bit assignments . . . 222
Table 163. PLL_CLK_CFG register bit assignments . . . 223
Table 164. CORE_CLK_CFG register bit assignments . . . 225
Table 165. PRPH_CLK_CFG register bit assignments . . . 227
Table 166. PERIP1_CLK_ENB register bit assignments . . . 228
Table 167. RAS_CLK_ENB register bit assignments . . . 230
Table 168. PRSC1/2/3_CLK_CFG register bit assignments . . . 232
Table 169. AMEM_CFG_CTRL register bit assignments . . . 232
Table 170. Clock Synthesizer input frequency . . . 233
Table 171. Auxiliary clock synthesizer register bit assignments . . . 234
Table 172. PERIP1_SOF_RST register bit assignments . . . 235
Table 173. RAS_SOF_RST register bit assignments . . . 237
Table 174. Interconnection matrix . . . 238
Table 175. ICM 1-9_ARB_CFG register bit assignments . . . 238
Table 176. DMA_CHN_CFG register bit assignment . . . 241
Table 177. USB2_PHY_CFG register bit assignments . . . 242
Table 178. MAC_CFG_CTR register bit assignments . . . 242
Table 179. Powerdown_CFG_CTR register bit assignments . . . 243
Table 180. COMPSSTL_1V8_CFG/DDR_2V5_COMPENSATION register bit assignments . . . 244
Table 181. COMPCOR_3V3_CFG register bit assignments . . . 245
Table 182. DDR_PAD register bit assignments . . . 245
Table 183. BIST1_CFG_CTR register bit assignments . . . 247
Table 184. BIST2_CFG_CTR register bit assignments . . . 250
Table 185. BIST3_CFG_CTR register bit assignments . . . 251
Table 186. BIST4_CFG_CTR register bit assignments . . . 252
Table 187. BIST1_STS_RES register bit assignments . . . 253
Table 188. BIST2_STS_RES register bit assignments . . . 254
Table 189. BIST3_STS_RES register bit assignments . . . 256
Table 190. BIST4_STS_RES register bit assignments . . . 257
Table 191. BIST5_RSLT_REG register bit assignments . . . 259
Table 192. SYSERR_CFG_CTR register bit assignments . . . 260
Table 193. USB_TUN_PRM register bit assignments . . . 262
Table 194. Drive selection . . . 262
Table 195. Pull Up and Pull Down Selection . . . 263
Table 196. Slew selection . . . 263
Table 197. PLGPIO0_PAD_PRG register bit assignments . . . 263
Table 198. PLGPIO1_PAD_PRG register bit assignments . . . 264
Table 199. PLGPIO2_PAD_PRG register bit assignments . . . 265
Table 200. PLGPIO3_PAD_PRG register bit assignments . . . 266
Table 201. PLGPIO4_PAD_PRG register bit assignments . . . 267
Table 202. Miscellaneous global space registers overview . . . 268
Table 203. RAS_GPP1_IN register bit assignments . . . 268
Table 204. RAS_GPP2_IN register bit assignments . . . 268

Table 205. RAS_GPP_EXT_IN register bit assignments . . . 269
Table 206. RAS_GPP1_OUT register bit assignments . . . 269
Table 207. RAS_GPP2_OUT register bit assignments . . . 269
Table 208. RAS_GPP_EXT_OUT register bit assignments . . . 269
Table 209. SSP signal interface . . . 271
Table 210. External CS selection . . . 274
Table 211. External pin connection . . . 277
Table 212. SSP registers summary . . . 277
Table 213. SSPCR0 register bit assignments . . . 278
Table 214. SSPCR1 register bit assignments . . . 279
Table 215. SSPDR register bit assignments . . . 280
Table 216. SSPSR register bit assignments . . . 280
Table 217. SSPCPSR register bit assignments . . . 281
Table 218. SSPIMSC register bit assignments . . . 281
Table 219. SSPRIS register bit assignments . . . 281
Table 220. SSPMIS register bit assignments . . . 282
Table 221. SSPICR register bit assignments . . . 282
Table 222. SSPDMACR register bit assignments . . . 283
Table 223. PHERIPHID0 register bit assignments . . . 283
Table 224. PHERIPHID1 register bit assignment . . . 283
Table 225. PHERIPHID2 register bit assignments . . . 283
Table 226. PHERIPHID3 register bit assignments . . . 283
Table 227. PCELLID0 register bit assignments . . . 284
Table 228. PCELLID1 register bit assignment . . . 284
Table 229. PCELLID2 register bit assignment . . . 284
Table 230. PCELLID3 register bit assignment . . . 284
Table 231. System controller control and status registers summary . . . 292
Table 232. System controller identification registers summary . . . 292
Table 233. SCCTRL register bit assignments . . . 293
Table 234. SCIMCTRL register bit assignments . . . 295
Table 235. SCIMSTAT register bit assignments . . . 295
Table 236. SCXTALCTRL register bit assignments . . . 296
Table 237. SCPLLCTRL register bit assignments . . . 296
Table 238. SMI supported instruction . . . 300
Table 239. External pin connection . . . 306
Table 240. SMI registers summary . . . 306
Table 241. SMI_CR1 register bit assignments . . . 306
Table 242. SMI_CR2 register bit assignments . . . 308
Table 243. SMI_SR register bit assignments . . . 310
Table 244. SMI_TR register bit assignments . . . 311
Table 245. SMI_RR register bit assignments . . . 312
Table 246. Watchdog module counter decremented . . . 314
Table 247. Watchdog control and status registers summary . . . 315
Table 248. Watchdog identification registers summary . . . 315
Table 249. WdogControl register bit assignments . . . 316
Table 250. WdogRIS register bit assignments . . . 317
Table 251. WdogMIS register bit assignments . . . 317
Table 252. WdogLock register bit assignments . . . 317
Table 253. External pin connection . . . 318
Table 254. GPT interface signal description . . . 320
Table 255. Couple of GPTs registers summary . . . 321
Table 256. Timer_Control register bit assignments . . . 321

Table 257. PRESCALER configuration . . . 322
Table 258. TIMER_STATUS_INT_ACK register bit assignments . . . 323
Table 259. TIMER_COMPARE register bit assignments . . . 323
Table 260. TIMER_COUNT register bit assignments . . . 324
Table 261. TIMER_REDG_CAPT register bit assignments . . . 324
Table 262. TIMER_FEDG_CAPT register bit assignments . . . 324
Table 263. GPIO signal interface . . . 325
Table 264. GPIO data direction register . . . 329
Table 265. GPIO data register . . . 329
Table 266. GPIO interrupt control registers summary . . . 329
Table 267. GPIO identification registers summary . . . 329
Table 268. GPIODIR register bit assignments . . . 330
Table 269. GPIODATA register bit assignments . . . 330
Table 270. GPIOIS register bit assignments . . . 330
Table 271. GPIOIBE register bit assignments . . . 331
Table 272. GPIOIEV register bit assignments . . . 331
Table 273. GPIOIE register bit assignments . . . 331
Table 274. GPIORIS register bit assignments . . . 332
Table 275. GPIOMIS register bit assignments . . . 332
Table 276. GPIOIC register bit assignments . . . 332
Table 277. DMAC signal interface . . . 335
Table 278. DMAC global registers summary . . . 338
Table 279. DMAC channel registers summary . . . 339
Table 280. DMAC peripheral registers summary . . . 339
Table 281. DMAC cell identification registers summary . . . 339
Table 282. DMACIntStatus register bit assignments . . . 340
Table 283. DMACIntTCStatus register bit assignments . . . 340
Table 284. DMACIntTCClear register bit assignments . . . 341
Table 285. DMA CIntErrorStatus register bit assignments . . . 341
Table 286. DMACIntErrClr register bit assignments . . . 341
Table 287. DMACRawIntTCStatus register bit assignments . . . 342
Table 288. DMACRawIntErrorStatus register bit assignments . . . 342
Table 289. DMACEnbldChns register bit assignments . . . 342
Table 290. DMACSoftBReq register bit assignments . . . 343
Table 291. DMACSoftSReq register bit assignments . . . 343
Table 292. DMACSoftLBReq register bit assignments . . . 344
Table 293. DMACSoftLSReq register bit assignments . . . 344
Table 294. DMACConfiguration register bit assignments . . . 344
Table 295. DMACSync register bit assignments . . . 345
Table 296. DMACCnSrcAddr register bit assignments . . . 346
Table 297. DMACCnDestAddr register bit assignments . . . 346
Table 298. DMACCnLLI register bit assignments . . . 346
Table 299. DMACCnControl register bit assignments . . . 347
Table 300. DMAC Configuration register bit assignments . . . 349
Table 301. RTC functional registers summary . . . 352
Table 302. CONTROL register bit assignments . . . 353
Table 303. STATUS register bit assignments . . . 353
Table 304. TIME register bit assignments . . . 354
Table 305. DATE register bit assignments . . . 355
Table 306. ALARM TIME register bit assignments . . . 355
Table 307. ALARM DATE register bit assignments . . . 356
Table 308. REG1MC registers bit assignments . . . 356

Table 309. REG2MC register bit assignments . . . 356
Table 310. C3 device summary . . . 358
Table 311. C3 components system register summary . . . 362
Table 312. C3 components system registers map . . . 364
Table 313. AHB mapped registers for Master Interface (HIF) . . . 369
Table 314. AHB mapped registers for an Instruction Dispatcher (ID) . . . 377
Table 315. AHB mapped registers for Channel (CH) . . . 382
Table 316. Channel ID Table . . . 383
Table 317. DES ECB start instruction bit encoding . . . 384
Table 318. DES ECB bit ‘a’ encoding . . . 384
Table 319. DES ECB bit ‘b’ encoding . . . 384
Table 320. DES CBC START Instruction Bit Encoding . . . 384
Table 321. DES ECB APPEND Instruction bit encoding . . . 385
Table 322. DES CBC Append Instruction Bit Encoding . . . 386
Table 323. DES registers map . . . 386
Table 324. AES ECB START instruction bit encoding . . . 389
Table 325. AES ECB Bit ‘a’ encoding . . . 389
Table 326. AES CBC START instruction bit encoding . . . 389
Table 327. AES CTR START instruction bit encoding . . . 390
Table 328. AES ECB APPEND instruction bit encoding . . . 390
Table 329. AES CBC APPEND instruction bit encoding . . . 391
Table 330. AES CTR APPEND instruction bit encoding . . . 391
Table 331. AES registers map . . . 391
Table 332. HASH INIT Instruction bit encoding . . . 396
Table 333. HASH INIT Bit ‘aa’ encoding . . . 396
Table 334. HASH APPEND Instruction bit encoding . . . 396
Table 335. HASH END Instruction bit encoding . . . 397
Table 336. HASH CONTEXT SAVE Instruction bit encoding . . . 397
Table 337. HASH CONTEXT RESTORE Instruction bit encoding . . . 397
Table 338. HMAC INIT Instruction bit encoding . . . 398
Table 339. HMAC APPEND Instruction bit encoding . . . 398
Table 340. HMAC END Instruction bit encoding . . . 399
Table 341. HMAC CONTEXT SAVE Instruction bit encoding . . . 400
Table 342. HMAC CONTEXT RESTORE instruction bit encoding . . . 400
Table 343. UHH channel registers map . . . 400
Table 344. External pin connections . . . 417
Table 345. UHC registers’ base address . . . 418
Table 346. EHCI host controller capability registers summary . . . 419
Table 347. EHCI host controller operational registers summary . . . 419
Table 348. EHCI host controller auxiliary power well registers summary . . . 419
Table 349. EHCI host controller specific registers summary . . . 420
Table 350. Host controller operational registers . . . 420
Table 351. HCCAPBASE register bit assignments . . . 421
Table 352. HCSPARAMS register bit assignments . . . 421
Table 353. HCCPARAMS register bit assignments . . . 423
Table 354. USBCMD register bit assignments . . . 425
Table 355. USBSTS register bit assignments . . . 428
Table 356. USBINTR register bit assignments . . . 430
Table 357. FRINDEX register bit assignments . . . 431
Table 358. USBCMD register encoding . . . 432
Table 359. PERIODICLISTBASE register bit assignments . . . 433
Table 360. ASYNCLISTADDR register bit assignments . . . 433

Table 361. CONFIGFLAG register bit assignments 434
Table 362. PORTSC register bit assignments 434
Table 363. INSNREG01 register bit assignments 440
Table 364. INSNREG03 register bit assignments 440
Table 365. INSNREG05 register bit assignments 440
Table 366. HcRevision register bit assignments. 441
Table 367. HcControl register bit assignments. 441
Table 368. HcCommandStatus register bit assignments 444
Table 369. HcInterruptStatus register bit assignments. 446
Table 370. HcInterruptEnable register bit assignments 447
Table 371. HcInterruptDisable register bit assignments. 448
Table 372. HcHCCA register bit assignments 449
Table 373. HcPeriodCurrentED register bit assignments. 449
Table 374. HcControlHeadED register bit assignments. 449
Table 375. HcControlCurrentED register bit assignments 450
Table 376. HcBulkHeadED register bit assignments 450
Table 377. HcBulkCurrentED register bit assignments. 451
Table 378. HcDoneHead register bit assignments 451
Table 379. HcFmInterval register bit assignments 452
Table 380. HcFmRemaining register bit assignments 452
Table 381. HcFmNumber register bit assignments. 453
Table 382. HcPeriodicStart register bit assignments 453
Table 383. HcLSThreshold register bit assignments 454
Table 384. HcRhDescriptorA register bit assignments. 455
Table 385. HcRhDescriptorB register bit assignments. 457
Table 386. HcRhStatus register bit assignments 457
Table 387. HcRhPortStatus register bit assignments. 459
Table 388. Endpoints assignment 463
Table 389. SETUP data memory: status quadlet bit assignments 478
Table 390. Out data memory: buffer status quadlet bit assignments (for Non-Isochronous OUT) . . . 480
Table 391. Out data memory: buffer status quadlet bit assignments (for Isochronous OUT). 481
Table 392. . . . . In data memory: buffer status quadlet bit assignments (for Non-Isochronous IN)482
Table 393. In data memory:buffer status quadlet bit assignments (for Isochronous) 483
Table 394. Plug status register bit assignments. 486
Table 395. Plug pending register bit assignments 486
Table 396. In endpoint-specific CSRs summary. 487
Table 397. Out endpoint-specific CSRs summary 488
Table 398. Global CSRs summary 488
Table 399. UDCI CSRs summary 489
Table 400. Device configuration register bit assignments 490
Table 401. Device control register bit assignments 492
Table 402. Device status register bit assignments 494
Table 403. Device interrupt register bit assignments 496
Table 404. Device interrupt mask register bit assignments 497
Table 405. Endpoint interrupt register bit assignments 497
Table 406. Endpoint interrupt mask register bit assignments. 497
Table 407. Endpoint control register bit assignments. 498
Table 408. Endpoint status register bit assignments 500
Table 409. Endpoint buffer size/received packet frame number register bit assignments 501
Table 410. Endpoint maximum packet size/buffer size register bit assignments 503
Table 411. Endpoint SETUP buffer pointer register bit assignments 503
Table 412. Endpoint data description pointer register bit assignments 503

Table 413. Endpoint register bit assignments . . . 503
Table 414. Transmit descriptor 0 (TDES0) . . . 510
Table 415. Transmit descriptor 1 (TEDS1) . . . 512
Table 416. Transmit descriptor 2 (TDES2) . . . 512
Table 417. Transmit descriptor 3 (TDES3) . . . 512
Table 418. Receive descriptor 0 (RDES0) . . . 513
Table 419. Receive descriptor 1 (RDES1) . . . 514
Table 420. Receive descriptor 2 (RDES2) . . . 515
Table 421. Receive descriptor 3 (RDES3) . . . 515
Table 422. MAC-UNIV DMA registers summary . . . 517
Table 423. MAC-UNIV MAC global registers summary . . . 517
Table 424. MMC (MAC management counters) registers . . . 518
Table 425. Bus mode register bit assignments . . . 522
Table 426. Transmit poll demand register bit assignments . . . 523
Table 427. Receive poll demand register bit assignments . . . 523
Table 428. Receive descriptor list address register bit assignments . . . 524
Table 429. Transmit descriptor list address register bit assignments . . . 524
Table 430. Status register bit assignments . . . 525
Table 431. EB field bit assignments . . . 526
Table 432. TS filed bit assignments . . . 526
Table 433. RS field bit assignments . . . 527
Table 434. NIS field bit assignments . . . 527
Table 435. AIS field bit assignments . . . 527
Table 436. Operation mode register bit assignments . . . 530
Table 437. TTC field bit assignments . . . 530
Table 438. RFD field bit assignments . . . 531
Table 439. RFA field bit assignments . . . 531
Table 440. RTC field bit assignments . . . 532
Table 441. Interrupt enable register bit assignments . . . 532
Table 442. Missed frame and buffer overflow counter register bit assignments . . . 533
Table 443. MAC configuration register bit assignments . . . 535
Table 444. IFG field bit assignments . . . 536
Table 445. BL field bit assignments . . . 537
Table 446. MAC frame filter register bit assignments . . . 538
Table 447. PCF field bit assignments . . . 539
Table 448. MII address register bit assignments . . . 540
Table 449. CR field bit assignments . . . 541
Table 450. MII data register bit assignments . . . 541
Table 451. Flow control register bit assignments . . . 542
Table 452. PLT field bit assignments . . . 542
Table 453. VLAN tag register bit assignments . . . 543
Table 454. 4 bit command registers . . . 544
Table 455. PMT CSR bit assignments . . . 545
Table 456. Interrupt status register bit assignments . . . 545
Table 457. Interrupt mask register bit assignments . . . 546
Table 458. MAC address0 high register bit assignments . . . 546
Table 459. MAC Address0 low register bit assignments . . . 547
Table 460. MAC Address1 high register bit assignments . . . 547
Table 461. MAC address byte . . . 547
Table 462. MAC Address1 low register bit assignments . . . 548
Table 463. MMC control register bit assignments . . . 548
Table 464. MMC receive interrupt register bit assignments . . . 549

Table 465. MMC transmit interrupt register bit assignments . . . 550
Table 466. MMC receive interrupt mask register bit assignments . . . 552
Table 467. GPIO signal interface . . . 555
Table 468. JPGC memory map . . . 558
Table 469. JPGC codec core registers . . . 559
Table 470. JPGC codec controller registers . . . 559
Table 471. JPGC FIFO registers . . . 559
Table 472. JPGC internal memories . . . 560
Table 473. JPGCreg0 register bit assignments . . . 560
Table 474. JPGCreg1 register bit assignments . . . 561
Table 475. JPGCreg2 register bit assignments . . . 562
Table 476. JPGCReg3 register bit assignments . . . 562
Table 477. JPGCreg4-7 register bit assignments . . . 563
Table 478. JPGC control status register bit assignments . . . 564
Table 479. JPGC bytes from Fifo to core register bit assignments . . . 564
Table 480. JPGC bytes from core to Fifo register bit assignments . . . 565
Table 481. JPGCbust Count before Init register bit assignments . . . 565
Table 482. JPGC Fifo in register bit assignments . . . 566
Table 483. JPGC Fifo out register bit assignments . . . 566
Table 484. JPGCqmem memory map . . . 566
Table 485. JPGCHuffMin memory map . . . 567
Table 486. JPGC huffbase memory map . . . 567
Table 487. JPGC huffsymb memory map . . . 567
Table 488. JPGCDHTmem memory map . . . 568
Table 489. JPGCHuffEnc memory map . . . 569
Table 490. Location of AC huffman codes in JPGCHuffEnc memory . . . 569
Table 491. Location of DC huffman codes in JPGCHuffEnc memory . . . 570
Table 492. Settings of K,L and (N+1) parameters for SIR,MIR and FIR in baud rate generation unit 575
Table 493. Request signals . . . 575
Table 494. FIrDA controller interrupt summary . . . 577
Table 496. FIrDA controller control and status registers summary . . . 578
Table 497. FIrDA controller data registers summary . . . 578
Table 498. FIrDA controller interrupt and DMA registers summary . . . 578
Table 499. IrDA_CON register bit assignments . . . 579
Table 500. IrDA_CONF register bit assignments . . . 579
Table 501. IrDA_PARA register bit assignments . . . 581
Table 502. IrDA_DV register bit assignments . . . 582
Table 503. IrDA_STAT register bit assignments . . . 582
Table 504. IrDA_TFS register bit assignments . . . 583
Table 505. IrDA_RFS register bit assignments . . . 583
Table 506. IrDA_TXB register bit assignments . . . 584
Table 507. IrDA_RXB register bit assignments . . . 584
Table 508. IrDA_IMSC register bit assignments . . . 584
Table 509. IrDA_RIS register bit assignments . . . 585
Table 510. IrDA_MIS register bit assignments . . . 586
Table 511. IrDA_ICR register bit assignments . . . 586
Table 512. IrDA_ISR register bit assignments . . . 587
Table 513. IrDA_DMA register bit assignments . . . 587
Table 514. UART interrupt summary together with combined outputs . . . 592
Table 515. UART base address . . . 594
Table 516. UART data registers summary . . . 594
Table 517. UART error status/clear registers summary . . . 594

Table 518. UART control and status register summary . . . 594
Table 519. UART interrupts and DMA registers summary . . . 595
Table 520. UART identification register summary . . . 595
Table 521. UART data register summary . . . 596
Table 522. UARTRSR register bit assignments . . . 596
Table 523. UARTECR register bit assignments . . . 596
Table 524. UARTFR register bit assignments . . . 597
Table 525. UARTIBRD register bit assignments . . . 598
Table 526. UARTFBRD register bit assignments . . . 598
Table 527. Typical baud rate and divisors . . . 599
Table 528. UARTLCR_H register bit assignments . . . 599
Table 529. Truth table for SPS, EPS and PEN bits . . . 600
Table 530. UARTCR register bit assignments . . . 601
Table 531. UARTIFLS register bit assignments . . . 602
Table 532. UARTIMSC register bit assignments . . . 603
Table 533. UARTRIS register bit assignments . . . 604
Table 534. UARTMIS register bit assignments . . . 604
Table 535. UARTICR register bit assignments . . . 605
Table 536. UARTDMACR register bit assignments . . . 606
Table 537. Meaning of UART modem input/output in DTE and DCE modes . . . 606
Table 538. First byte assignment in addressing slave protocol . . . 609
Table 539. I^2C controller interrupt sources . . . 617
Table 540. External pin connections . . . 619
Table 541. I2C registers . . . 619
Table 542. IC_CON register bit assignments . . . 622
Table 543. IC_TAR register bit assignments . . . 624
Table 544. IC_SAR register bit assignments . . . 625
Table 545. IC_HS_MADDR register bit assignments . . . 625
Table 546. IC_DATA_CMD register bit assignments . . . 625
Table 547. IC_SS_SCL_HCNT register bit assignments . . . 626
Table 548. IC_SS_SCL_HCNT sample calculations . . . 627
Table 549. IC_SS_SCL_LCNT register bit assignments . . . 627
Table 550. IC_SS_SCL_LCNT sample calculations . . . 627
Table 551. IC_FS_SCL_HCNT register bit assignments . . . 628
Table 552. IC_FS_SCL_HCNT sample calculations . . . 628
Table 553. IC_FS_SCL_LCNT register bit assignments . . . 629
Table 554. IC_FS_SCL_LCNT sample calculations . . . 629
Table 555. IC_HS_SCL_HCNT register bit assignments . . . 630
Table 556. IC_HS_SCL_HCNT sample calculations . . . 630
Table 557. IC_HS_SCL_LCNT register bit assignments . . . 631
Table 558. IC_HS_SCL_LCNT sample calculations . . . 631
Table 559. IC_INTR_STAT register bit assignments . . . 631
Table 560. IC_INTR_MASK register bit assignments . . . 632
Table 561. IC_RAW_INTR_STAT register bit assignments . . . 633
Table 562. IC_RX_TL register bit assignments . . . 634
Table 563. IC_TX_TL register bit assignments . . . 634
Table 564. IC_CLR_INTR register bit assignments . . . 635
Table 565. Interrupt clearing registers . . . 635
Table 566. IC_ENABLE register bit assignments . . . 636
Table 567. IC_STATUS register bit assignments . . . 636
Table 568. IC_TXFLR and IC_RXFLR register bit assignments . . . 637
Table 569. IC_TX_ABRT_SOURCE register bit assignments . . . 638

Table 570. IC_DMA_CR register bit assignments . . . 640
Table 571. IC_DMA_TDLR register bit assignments . . . 640
Table 572. IC_DMA_RDLR register bit assignments . . . 640
Table 573. IC_COMP_PARAM register bit assignments . . . 641
Table 574. External pin connection . . . 645
Table 575. ADC registers summary . . . 645
Table 576. ADC_STATUS_REG register . . . 646
Table 577. Conversion data bits position in AVERAGE_REG (High Resolution = 0) . . . 648
Table 578. Conversion data bits position in AVERAGE_REG (High Resolution = 1) . . . 648
Table 579. SCAN RATE register bit assignments . . . 649
Table 580. ADC_CLK_REG register bit assignments . . . 649
Table 581. CHx CTRL register bit assignments . . . 650
Table 582. CHx DATA register (normal mode) bit assignments . . . 651
Table 583. CHx DATA register (HIGH RESOLUTION mode bit assignments . . . 651
Table 584. RAS address space . . . 657
Table 585. FSMC address space . . . 657
Table 586. APB address space . . . 657
Table 587. RAS memory map . . . 657
Table 588. RAS Register 1 description . . . 658
Table 589. RAS Register 2 description . . . 659
Table 590. RAS Interrupt assignment . . . 660
Table 591. RAS DMA configuration . . . 661
Table 592. Parallel NOR flash . . . 664
Table 593. NAND Flash . . . 665
Table 594. FSMC Address Map . . . 665
Table 595. FSMC control and timing registers summary . . . 666
Table 596. FSMC identification registers summary . . . 667
Table 597. GenMemCtrl(i) register bit assignments . . . 667
Table 598. GenMemCtrl_tim(i) register bit assignments . . . 669
Table 599. GenMemCtrl_PC(i) register bit assignments . . . 670
Table 600. GenMemCtrl_Comm(i)/GenMemCtrl_Attrib(i) register bit assignments . . . 671
Table 601. GenMemCtrlECCr(i) register bit assignments . . . 671
Table 602. GenMemCtrlPeriphID0 register bit assignments . . . 672
Table 603. GenMemCtrlPeriphID1 register bit assignments . . . 672
Table 604. GenMemCtrlPeriphID2 register bit assignments . . . 672
Table 605. GenMemCtrlPeriphID3 register bit assignments . . . 672
Table 606. GenMemCtrlPeriphID3 register bit assignments . . . 673
Table 607. GenMemCtrlPeriphID1 register bit assignments . . . 673
Table 608. GenMemCtrlPCellID2 register bit assignments . . . 673
Table 609. GenMemCtrlPCellID3 register bit assignments . . . 673
Table 610. Signal interface . . . 679
Table 611. AHB master/Target interface . . . 680
Table 612. SD2.0/SDIO2.0/MMC4.2 card interface . . . 680
Table 613. System Interface . . . 682
Table 614. RAM Interface signals . . . 682
Table 615. SDIO registers map . . . 695
Table 616. Register field types . . . 696
Table 617. SDMASysAddr register bit assignments . . . 697
Table 618. BLKSize register bit assignments . . . 697
Table 619. BLKCount register bit assignments . . . 699
Table 620. ARG register bit assignments . . . 699
Table 621. TRMODE register bit assignments . . . 699

Table 622. Determination of transfer type . . . 700
Table 623. CMD register bit assignments . . . 701
Table 624. Relation between parameters and the name of response type . . . 702
Table 625. RESP register bit assignments . . . 702
Table 626. Response bit definition for each response type . . . 703
Table 627. BufDataPort register bit assignments . . . 703
Table 628. PRSTATE register bit assignments . . . 703
Table 629. HOSTCTRL register bit assignments . . . 707
Table 630. PWRCTRL register bit assignments . . . 708
Table 631. BLKGAPCTRL register bit assignments . . . 708
Table 632. WKUPCTRL register bit assignments . . . 710
Table 633. CLKCTRL register bit assignments . . . 712
Table 634. TMOUTCTRL register bit assignments . . . 713
Table 635. SWRES register bit assignments . . . 714
Table 636. NIRQSTAT register bit assignments . . . 715
Table 637. Relation between transfer complete and data time out error . . . 717
Table 638. Relation between command complete and time out error . . . 718
Table 639. ERRIRQSTAT register bit assignments . . . 718
Table 640. Relation between command CRC error end time out error . . . 720
Table 641. NIRQSTATEN register bit assignments . . . 721
Table 642. ERRIRQSTATEN register bit assignments . . . 721
Table 643. NIRQSIGEN register bit assignments . . . 722
Table 644. ERRIRQSIGEN register bit assignments . . . 723
Table 645. ACMD12ERSTS register bit assignments . . . 723
Table 646. Relation between auto CMD12 CRC error and auto CMD12 timeout error . . . 724
Table 647. CAP1 register bit assignments . . . 725
Table 648. CAP2 register bit assignments . . . 727
Table 649. MAXCURR1 register bit assignments . . . 727
Table 650. MAXCURR2 register bit assignments . . . 728
Table 651. Maximum current value definition . . . 728
Table 652. ACMD12FEERSTS register bit assignments . . . 728
Table 653. FEERRINTSTS register bit assignments . . . 729
Table 654. ADMAERRSTS register bit assignments . . . 730
Table 655. ADMAERRSTS bits[1:0] definition . . . 731
Table 656. ADMAADDR register bit assignments . . . 732
Table 657. 32 bit address ADMA . . . 732
Table 658. 64 bit Address ADMA . . . 732
Table 659. SPIIRQSUPP register bit assignments . . . 733
Table 660. SLTIRQSTS register bit assignments . . . 733
Table 661. HCTRLVER register bit assignments . . . 733
Table 662. CLCD signal interface . . . 738
Table 663. LCD STN panel signal multiplexing . . . 739
Table 664. LCD TFT panel signal multiplexing . . . 739
Table 665. LBLP, DMA FIFO output bit 31 to bit 16 . . . 742
Table 666. LBLP, DMA FIFO output bit15 to bit 0 . . . 742
Table 667. BBBP, DMA FIFO output bit 31 to bit 16 . . . 742
Table 668. BBBP, DMA FIFO output bit15 to bit 0 . . . 742
Table 669. LBBP, DMA FIFO output bit 31 to bit 16 . . . 743
Table 670. LBBP, DMA FIFO output bit15 to bit 0 . . . 743
Table 671. RGB Mode data format . . . 743
Table 672. Palette data storage . . . 745
Table 673. CLCD configuration registers . . . 748

Table 674. Color palette register . . . 749
Table 675. Identification register . . . 749
Table 676. LCDTiming0 register bit assignments . . . 750
Table 677. LCDTiming1 register bit assignments . . . 751
Table 678. LCDTiming2 register bit assignments . . . 752
Table 679. LCDTiming3 register bit assignments . . . 754
Table 680. LCDUPBASE register bit assignments . . . 755
Table 681. LCDLPBASE register bit assignments . . . 755
Table 682. LCDIMSC register bit assignments . . . 755
Table 683. LCDControl register bit assignments . . . 756
Table 684. LCDRIS register bit assignments . . . 758
Table 685. LCDMIS register bit assignments . . . 758
Table 686. LCDICR register bit assignments . . . 759
Table 687. LCDUPCURR register bit assignments . . . 759
Table 688. LCDLPCURR register bit assignments . . . 759
Table 689. LCDPalette register bit assignments . . . 759
Table 690. PHERIPHID0 register bit assignments . . . 760
Table 691. PHERIPHID1 register bit assignments . . . 760
Table 692. PHERIPHID2 register bit assignments . . . 761
Table 693. PHERIPHID3 register bit assignments . . . 761
Table 694. PCCELLIDIDO register bit assignments . . . 761
Table 696. PCELLIDID2 register bit assignments . . . 761
Table 698. Telecom block pin signals . . . 765
Table 699. I2S interface pins . . . 777
Table 700. Camera interface signal . . . 781
Table 701. Telecom address map . . . 783
Table 702. Telecom registers . . . 783
Table 703. Boot register (Offset 0x00) . . . 784
Table 704. TDM_conf register (Offset 0x04) . . . 785
Table 705. GPIO8_DIR register (Offset 0x08) . . . 787
Table 706. GPIO10_DIR register (Offset 0x0C) . . . 788
Table 707. GPIO8_out register (Offset 0x10) . . . 788
Table 708. GPIO10_out register (Offset 0x14) . . . 789
Table 709. GPIO8_in (Offset 0x18) . . . 789
Table 710. GPIO10_in register (Offset 0x1C) . . . 790
Table 711. IT_GEN register (Offset 0x24) . . . 790
Table 712. GPIOt register (Offset 0x28) . . . 792
Table 713. GPIOtt register (Offset 0x2C) . . . 792
Table 714. PERS_time register (Offset 0x30) . . . 792
Table 715. PERS_data register (Offset 0x34) . . . 792
Table 716. TDM_timeselot_NBR register (Offset 0x38) . . . 793
Table 717. TDM_Frame_NBR register (Offset 0x3C) . . . 793
Table 718. TDM_SYNC_GEN register (Offset 0x40) . . . 793
Table 719. SPI_I2C_usage register (Offset 0x44) . . . 798
Table 720. SPI_I2C_active (Offset 0x48) . . . 798
Table 721. I2S_CONF register (Offset 0x4C) . . . 799
Table 722. I2S_CONF2 register (Offset 0x6C) . . . 801
Table 723. I2S_CLK_CONF register (Offset 0x50) . . . 802
Table 724. Interrupt mask register (Offset 0x54) . . . 804
Table 725. Dummy access address . . . 805
Table 726. Interrupt status register (Offset 0x58) . . . 805
Table 727. Action memory . . . 806

Table 728. External signals . . . 811
Table 729. Register map . . . 812
Table 730. MDCTRLREG register bit assignments . . . 812
Table 731. GPIODIRREG register bit assignments . . . 813
Table 732. .GPIODATAREG register bit assignments . . . 813
Table 733. STATUSREG register bit assignments . . . 813
Table 734. KBREG register bit assignments . . . 814
Table 735. Key-code table (hex values) . . . 814
Table 736. Power state for synchronous DRAM system (DRAM clocked by PLL1) . . . 817
Table 737. Power state for asynchronous DRAM system (DRAM clocked by PLL2) . . . 818
Table 738. Techniques applicable in NORMAL state . . . 820
Table 739. Modules supporting DCS technique . . . 821
Table 740. Power and current consumption for modules . . . 824
Table 741. Delta power consumption for modules . . . 824
Table 742. IP voltage usage . . . 825
Table 743. Booting types . . . 826
Table 744. Command format . . . 834
Table 745. Device descriptors . . . 835
Table 746. Configuration registers . . . 835
Table 747. Interface registers . . . 836
Table 748. Bulk OUT endpoint . . . 836
Table 749. Bulk In endpoint . . . 836
Table 750. String descriptors . . . 837
Table 751. Document revision history . . . 843

List of figures

Figure 1. SPEAr300 top view . . . 56
Figure 2. SPEAr300 - Core architecture overview . . . 59
Figure 3. Multiplexing scheme . . . 72
Figure 4. ARM926EJ-S block diagram . . . 86
Figure 5. VIC block diagram . . . 90
Figure 6. ICM block diagram . . . 104
Figure 7. MPMC block diagram . . . 106
Figure 8. Memory controller architecture . . . 109
Figure 9. WRAPx effective transaction . . . 114
Figure 10. Weighted round-robin priority group structure . . . 123
Figure 11. Memory map: Maximum . . . 143
Figure 12. Alternate memory map . . . 143
Figure 13. Clock generation scheme . . . 202
Figure 14. Processor clock . . . 203
Figure 15. DDR Controller Clock . . . 203
Figure 16. RAS block diagram . . . 204
Figure 17. I2S clock schematic . . . 205
Figure 18. Telecom clock schematic . . . 205
Figure 19. Main Crystal Connection . . . 207
Figure 20. RTC crystal connection . . . 207
Figure 21. Top view of miscellaneous registers . . . 208
Figure 22. SPI block diagram . . . 270
Figure 23. System controller block diagram . . . 287
Figure 24. System mode control state machine . . . 290
Figure 25. SMI block diagram . . . 299
Figure 26. External SPI memory map in AHB address space . . . 301
Figure 27. Watchdog module block diagram . . . 313
Figure 28. GPT block diagram . . . 319
Figure 29. GPIO block diagram . . . 325
Figure 30. GPIO signal interfaces diagram . . . 326
Figure 31. GPIO interrupt triggering logic . . . 328
Figure 32. DMAC block diagram . . . 334
Figure 33. DMAC signal interface diagram . . . 334
Figure 34. DMAC-to-interrupt controller connection . . . 337
Figure 35. C3 block diagram . . . 359
Figure 36. UHC block diagram . . . 412
Figure 37. USB Host controller (UHOSTC) block diagram . . . 415
Figure 38. UDC-AHB subsystem block diagram within the USB 2.0 device . . . 464
Figure 39. UDC_Device block diagram . . . 464
Figure 40. RxFIFO implementation . . . 467
Figure 41. Linked-list memory structure in DMA mode . . . 471
Figure 42. In transaction flow in DMA mode . . . 473
Figure 43. Out transaction flow in DMA mode . . . 474
Figure 44. In transaction flow in slave-only mode . . . 476
Figure 45. Out transaction flow in slave-only mode . . . 477
Figure 46. SETUP data memory . . . 478
Figure 47. Out data memory . . . 479
Figure 48. In data memory . . . 482

Figure 49. UDC-AHB subsystem memory map . . . 490
Figure 50. MAC-UNIV (MAC-AHB configuration) system-level block diagram . . . 506
Figure 51. DMA descriptor list: ring structure (left) and chain structure (right). . . . 509
Figure 52. DMA descriptor format (Transmit Descriptor, 32 bit) . . . 510
Figure 53. DMA descriptor format (Receive Descriptor, 32 bit) . . . 513
Figure 54. Interrupt management: sbd_intr_o and pmt_intr_o generation. . . . 516
Figure 55. Wake-up frame filter registers. . . . 544
Figure 56. Clocking scheme for MAC-AHB . . . 554
Figure 57. JPGC signal interfaces diagram . . . 556
Figure 58. JPGC block diagram. . . . 556
Figure 59. Dataflow block diagram of the FIrDA controller . . . 572
Figure 60. UART block diagram. . . . 590
Figure 61. I^2C controller functional block diagram . . . 608
Figure 62. START and STOP conditions [from I^2C-bus specification] . . . 609
Figure 63. START byte procedure [from I^2C-bus specification] . . . 610
Figure 64. Multiple master arbitration . . . 616
Figure 65. Clock synchronization . . . 617
Figure 66. ADC block diagram . . . 643
Figure 67. FSMC block diagram . . . 663
Figure 68. SDIO controller pin diagram . . . 679
Figure 69. Architectural block diagram . . . 683
Figure 70. SD card detection . . . 684
Figure 71. SD clock supply sequence . . . 685
Figure 72. Command issue sequence . . . 686
Figure 73. Command completion sequence . . . 687
Figure 74. Data transaction sequence without DMA . . . 689
Figure 75. Data transaction sequence with SDMA . . . 691
Figure 76. Data transaction sequence with ADMA . . . 693
Figure 77. Abort transaction sequence . . . 694
Figure 78. CLCD block diagram. . . . 737
Figure 79. Powering up and down sequences. . . . 748
Figure 80. Power up & power down sequences . . . 763
Figure 81. CLCD clock muxing scheme. . . . 764
Figure 82. Telecom block diagram. . . . 766
Figure 83. TDM clock cell block diagram . . . 767
Figure 84. SYNC0 (slave/master) and SYNC1 to SYNC3 possible shaping . . . 768
Figure 85. SYNC4 to SYNC7 generation. . . . 769
Figure 86. TDM cell . . . 769
Figure 87. Illustration for TDM switching . . . 771
Figure 88. Storage in memory during an odd switched frame . . . 771
Figure 89. Storage in memory during an even switched frame . . . 772
Figure 90. Various type of data carried by the TDM bus . . . 773
Figure 91. Address generation and bank switching . . . 773
Figure 92. Memory filling after 3 frames according narrowband cases . . . 774
Figure 93. Memory filling after 3 consecutive frames for two wideband cases . . . 775
Figure 94. Sample management on odd frame . . . 775
Figure 95. Sample management on even frame . . . 776
Figure 96. Read/write sequence during frame N of a buffer for a given channel. . . . 776
Figure 97. Buffer address generation . . . 777
Figure 98. I2S data reception and transmission (8 bits) . . . 778
Figure 99. I2S data flow on 2*32 bit data. . . . 779
Figure 100. DAC application . . . 780

Figure 101. DAC used with TDM at 8 kHz . . . 780
Figure 102. Camera interface block diagram . . . 781
Figure 103. IT bus change and persistency supervision . . . 782
Figure 104. TDM CLK_GEN bits . . . 787
Figure 105. I2S clock tree . . . 804
Figure 106. Interrupt management . . . 809
Figure 107. Keyboard controller block diagram . . . 810
Figure 108. Operative system control states . . . 817
Figure 109. Clock supply . . . 822
Figure 110. Typical power consumption with DDR2 @ 333 MHz . . . 823
Figure 111. Typical power consumption with DDR2 @ 166 MHz and mobile DDR . . . 823
Figure 112. Boot stages . . . 827
Figure 113. Hardware memory . . . 828
Figure 114. Boot flows . . . 831
Figure 115. Serial NOR Flash boot . . . 832
Figure 116. NAND Flash boot . . . 833
Figure 117. USB boot . . . 838
Figure 118. Serial boot . . . 840
Figure 119. Ethernet boot . . . 842

1 Acronyms

The below table contains acronyms and abbreviations that are used in this document.

Table 1. Acronyms

Terms	Expansion
ADC	Analog to Digital Convertor
AES	Advanced Encryption Standard
AFE	Analog Front End
ALU	Arithmetic Logic Unit
ASIC	Application Specific Integrated Circuit
AS	Application Subsystem
AMBA	Advanced Micro controller Bus Architecture
AHB	AMBA High speed Bus
APB	Advanced Peripheral Bus
BIST	Built-In Self Test
BS	Basic Subsystem
CBC	Cipher Block Chaining
CTR	Counter
CMOS	Complimentary Metal-Oxide Semiconductor.
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Convertor
DES	Data Encryption Standard
DLL	Data Link Layer
DMA	Direct Memory Access
DDR	Double Data Rate
ECB	Electronic Code Book
EHCI	Enhanced Host Controller Interface
FIFO	First-In-First-Out
FPGA	Field Programmable Gate Array
FSMC	Flexible Static Memory Controller
GPIO	General Purpose Input Output
HS	High speed Subsystem
IEEE	Institute of Electrical and Electronics Engineers
ISO	Isochronous
JPEG	Joint Photographic Experts Group
JTAG	Joint Test Action Group

Table 1. Acronyms (continued)

Terms	Expansion
LIFO	Last-In-First-Out
LS	Low speed Subsystem
LSB	Least Significant Bit
MAC	Media Access Control
MCU	Micro-Controller Unit
MD-5	Message Digest 5
MMU	Memory Management Unit
MSB	Most Significant Bit
OHCI	Open Host Controller Interface
PHY	Physical layer
RAM	Random Access Memory
RAS	Reconfigurable Array Subsystem
RF	Radio Frequency
RFU	Reserved for Future Use
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RTC	Real Time Clock
RX	Receive
SDIO	Secure Digital Input Output
SHA-1	Secure Hash Algorithm
SoC	System-on-Chip
SPEAr	Structured Processor Enhanced Architecture
SMI	Serial Memory Interface
SSP	Synchronous Serial Peripheral
TCM	Tightly Coupled Memory
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VIC	Vectored Interrupt Controller
WDT	Watchdog Timer

2 Preface

2.1 Terms & conditions

The following terms are used hereafter in this document.

- Reserved - All reserved or unused bits of registers must be written as zero, and ignored on read unless otherwise stated in the relevant text.

2.2 Conventions

2.2.1 Numbering

The following convention of stating constant numbers is used in this document:

- [<size in bits>]'<base><number>

where:

- **<size in bits>**: (optional) width of bits field associated to <number>
- **<base>**: “b” for binary, ‘h” for hexadecimal, “o” for octal, “d” for decimal
- **<number>**: value of constant number, according to <base>

Examples:

‘d19	unsized decimal value
8’h2C	8 bit wide hexadecimal value of 0x2C, corresponding to b00101100
8’b011001	8 bit wide binary value of b00011001
32’hFFFF	32 bit wide hexa decimal value of 0xFFFF, corresponding to b00000000000000001111111111111111

2.2.2 Bits

The conventions below apply to description of both bit and registry field hereafter in this document:

- a bit is defined as “set” when its value is set to ‘b1.
- a bit is defined as “cleared” when its value is set to ‘b0.

2.2.3 Typographical

The following typographical conventions are used hereafter in this document:

Table 2. Typographical conventions

Typographic format	Meaning
italic	Highlights notes
bold	Highlights important terms, definitions and names of registry field.
MONOSPACE CAPITAL BOLD	Indicates signal names.

3 Reference documentation

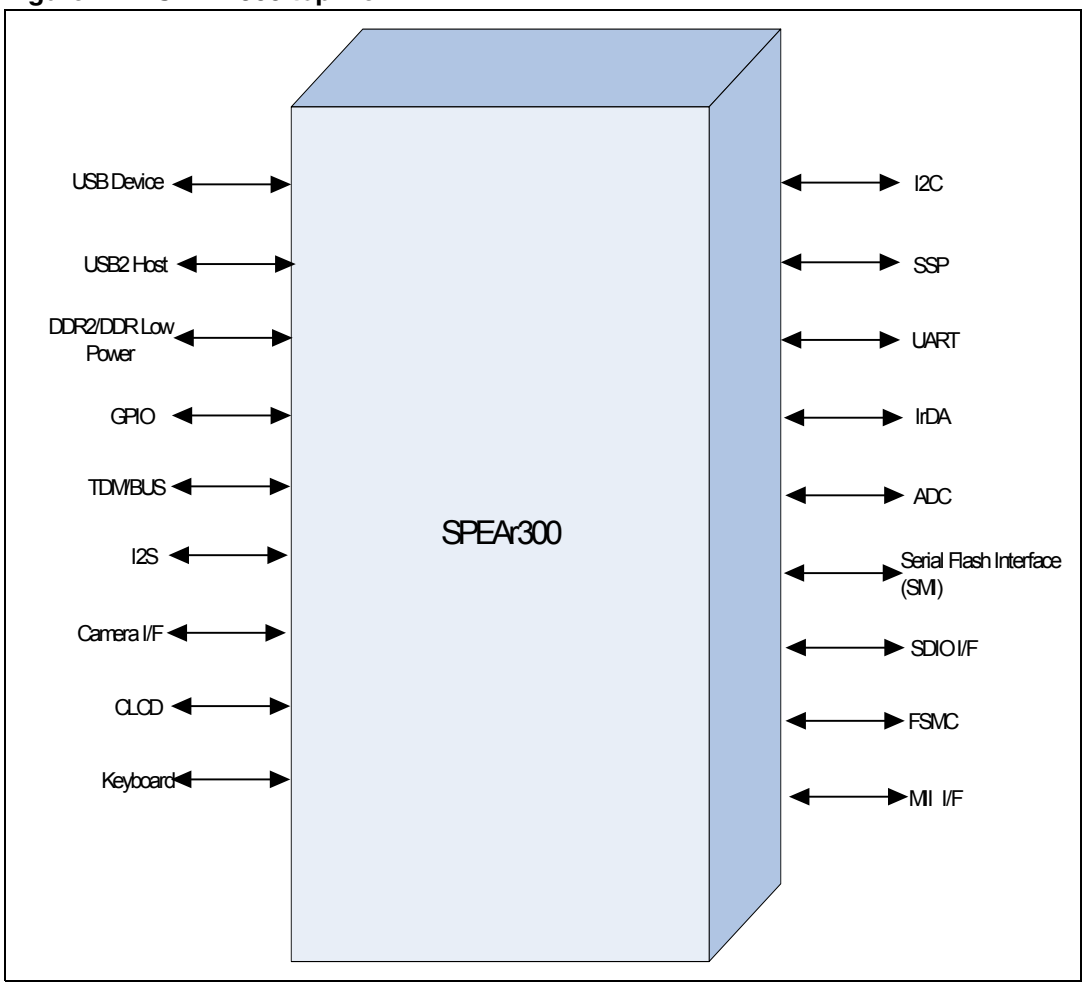
1. ARM926EJ-S – Technical reference manual
2. AMBA specification (ARM IHI 0011A), rev. 2.0
3. USB 2.0 specification
4. OHCI specification
5. EHCI specification
6. ISO/IEC 10918-1 (International Organization for Standardization)

4 Product overview

4.1 Device overview

An outline picture of the main SPEAr300 functional interfaces is shown in *Figure 1*.

Figure 1. SPEAr300 top view



4.1.1 Main features

The following main functionalities are implemented in SPEAr300 embedded MPU device:

- ARM926EJ-S core @333 MHz, 16+16 KB-I/D cache, configurable TCM-I/D size, MMU, TLB, JTAG and ETM trace module (multiplexed interfaces).
- Dynamic power saving features.
- High performance linked list 8-channel DMA.
- Multi-port memory controller: 8/16 bit mobileDDR@166 MHz or DDR2@333 MHz.
- USB2.0 Host (High-Full-Low speed); integrated PHY transceiver.
- USB2.0 Device (High-Full speed); integrated PHY transceiver.
- Ethernet 10/100 MAC with MII Interface (IEEE-802.3)
- I2C (High-Fast-Low speed) Master/Slave.
- Cryptographic co-processor (C3).
- IrDA controller with a data rate from 9.6 Kbps to 4 Mbps.
- Touchscreen support (using ADC).
- RTC - WDT - SYSCTR - MISC internal control registers.
- ADC (1us/1MSPS) 8 analog input channels; 10 bit resolution.
- JPEG codec accelerator single clock per pixel encoding decoding.
- 6 x 16 bits general purpose Timers with programmable prescaler (only 4 timers with capture mode).
- 32KB ROM & up to 56 KB internal SRAM
- Flexible static memory controller (FSMC) up to 16 bit data bus width, supporting external asynchronous SRAM, NAND/NOR Flash.
- 3 x SPI Master/Slave (Motorola-Texas_National) Master/Slave up to 50 Mbps.
- SDIO interface supporting SPI, SD1, SD4 and SD8 mode with card detect, write protect, LED.
- I2S interface, full duplex with data buffer for left and right channels allowing up to 64 ms of voice buffer (for 32 bit samples). *The I2S and SDIO interfaces share the same RAM resources.*
- UART with HW flow control (speed rate up to 3 Mbps)
- Up to 512 timeslots, master or slave TDM. Any input timeslot can be switched to any output timeslot, and/or can be buffered for computation (up to 16 channels of 1 to 4 timeslots buffered during 32 ms). Up to 16 buffers can be played in output timeslots.
- Up to 8 additional I2C/SPI chip selects.
- Camera interface ITU-601 with external or embedded synchronization (ITU-656 or CSI2). Picture limit is given by the line length that must be stored in a 2048*32 buffer.
- Color LCD Controller, supports up to 1024X768 resolution, 24 bpp true colour, STN/TFT display panels.
- 9 x 9 keyboard controller.
- 18 GPIOs for CODEC (up to 8 quad CODECS) & SLIC management.
- 1 bit DAC
- Up to 62 GPIOs (multiplexed with peripheral I/Os), up to 22 with interrupt capability.
- Possible NAND / NOR Flash booting.
- 8/16 bits parallel Flash interface allowing connection of NOR or NAND Flash.

4.2 Architecture properties

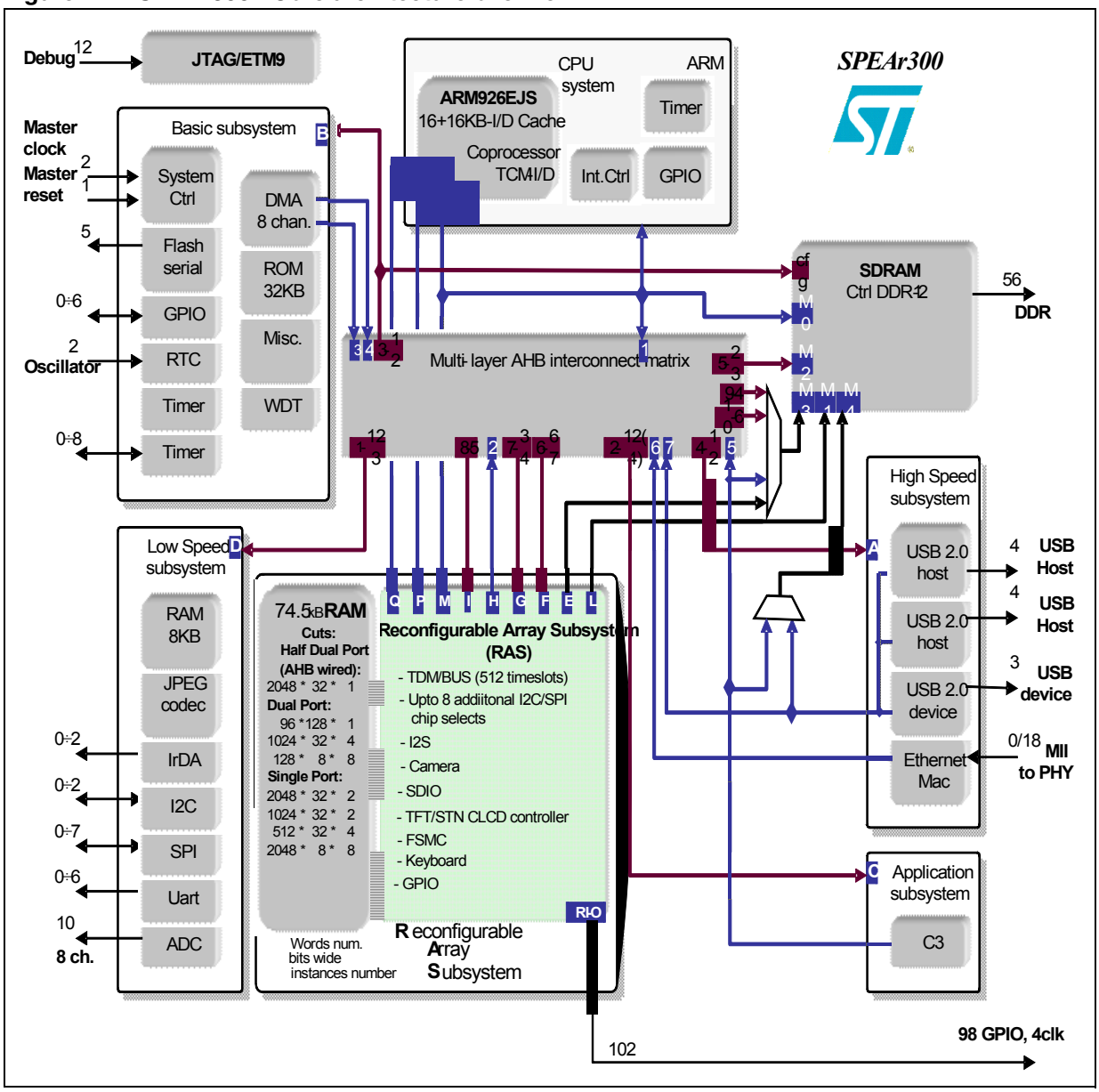
- Power save features:
  - Operating frequency SW programmable.
  - Clock gating functionality.
  - Low frequency operating mode.
  - Automatic power saving controlled from application activity demands.
- Architecture easily extensible.
- External memory bandwidth of each master tuneable to meet the target performances of different applications.

4.3 System architecture overview

4.3.1 Core architecture

The internal architecture is based on several-shared subsystem logics interconnected through a multilayer interconnection matrix as shown in the *Figure 2*.

Figure 2. SPEAr300 - Core architecture overview



The switch matrix structure allows different subsystem dataflows to be executed in parallel improving the core platform efficiency.

High performance master agents are directly interconnected with the memory controller reducing the memory access latency. Three different memory paths (two of them shared with other masters) are reserved for the programmable logic to enhance the user application

throughput. The overall memory bandwidth assigned to each master port can be programmed and optimized through an internal efficient weighted round-robin arbitration mechanism.

4.4 CPU subsystem

- ARM926EJ-S running at 333 MHz with:
  - MMU
  - 16 Kbyte of instruction cache
  - 16 Kbyte of data cache
  - AMBA bus interface
  - JTAG
  - ETM9 (embedded trace macro-cell) for debug, large size version.
- Local timer (two channels)
- Interrupt controller managing sources which are prioritized and vectorized.

4.5 Multilayer bus matrix

The bus matrix has seven master inputs: two DMA inputs, and one input each for the Reconfigurable Array Subsystem (RAS) block, Ethernet controller, USB device, USB host controller and C3. There are ten slave outputs connected to almost all the system blocks: Low Speed Subsystem, Application Subsystem, Basic Subsystem, High Speed Subsystem, DDR Controller Port-2, RAS-F, RAS-G, RAS-I and two DDR Controller Port - 3.

4.6 Dynamic memory controller

This is a multi-port memory controller able to manage DDR mobile up to 166 MHz and DDR2 up to 333 MHz external memory. Internally, it handles 5 ports supporting the following masters:

- CPU
- Reconfigurable Array Subsystem (RAS)
- RAS block and one DMA channel through the multilayer bus matrix
- Ethernet MAC muxed with USB 2.0 device, C3 accelerator and the RAS block.
- USB 2.0 Host controller muxed with C3 accelerator

as well as the configuration port that can be accessed by the CPU or by the RAS logic through the multilayer bus matrix.

The multi-port memory controller block has a programmable arbitration scheme and the transactions happen on a different layer from the main bus. It also offers a local FIFO to increase the throughput and reduce the latency.

4.7 Basic subsystem

- Eight high performance DMA channels with two AHB interfaces to parallelize the activity when two channels are working at the same time.
- 32 Kbyte of ROM.
- Serial Flash interface capable of working up to 50 Mbps.
- Four timers with programmable prescaler.
- Watchdog timer.
- RTC with separate power supply allowing battery connection
- Upto 6 GPIOs bidirectional signals with interrupt capability
- System controller and miscellaneous registers array allowing a full configurability of the system.

4.8 High speed connectivity subsystem

- Ethernet MAC controller that can support 10/100 Mbps with external PHY.
- One USB host controller compatible with USB 2.0 high-speed specification managing two ports. The peripheral has dedicated channel to the multi-port memory controller and two slave ports for CPU programming (OHCI and EHCI). The PHYs are embedded. One host controller at a time can perform high speed transfer.
- One USB device compatible with USB 2.0 high-speed specifications.
  A dedicated channel connects the peripheral with the multi-port memory controller and registers and internal FIFO are accessible from the CPU through the main AHB bus.
  An USB-Plug detector block is also available to detect the presence of the VBUS voltage.
  The port is provided with sixteen physical endpoints and proper configurations to achieve logical endpoints.

4.9 Low speed connectivity subsystem

- One UART with a data rate up to 3 Mbps.
- IrDA controller with a data rate from 9.6 Kbps to 4 Mbps.
- One Synchronous Serial Peripheral (SSP) controller capable of operating in master and slave (Motorola-Texas-National) with a data rate up to 40 Mbps.
- One I^2C controller capable of operating in master and slave mode and covering all the possible speeds (data rates) (high, fast and low).
- JPEG CODEC accelerator (1clk per pixel).
- 57 Kbyte of static RAM.
- ADC converter (1 µs/1 MSPS) with 8 analog input channels, 10 bit approximation.

4.10 Application subsystem

Channel **C**ontrol **C**oprocessor (**C3**) that offers the following main features:

- High performance DMA based co-processor enabling the acceleration of data-driven computationally expensive functions, such as: Cryptography, Pattern matching, Signal Processing, etc.
- Highly programmable (instruction driven) controller.
- Four Instruction Dispatchers.
- Four hardware accelerators channels with in/out FIFO inside.
- 64 KB of internal RAM for low latency accesses.
- Coupling/Chaining Module (internal cross- bar) for inter channel direct high-speed communications, up to 8 paths.
- Cryptographic channels library that includes:
- AES with ECB, CBC, CTR modes.
- Mode Programmable AES.
- DES/3DES with ECB, CBC modes.
- MD5, SHA-1, SHA256 with HMAC.

4.11 Reconfigurable logic array subsystem

The SPEAr300 also includes certain specific functions:

- 8/16 bits parallel Flash interface allowing connection of NOR or NAND Flash and asynchronous SRAM.
- Possible NAND Flash or parallel NOR Flash booting.
- Color LCD Controller, supports upto 1024 x 768 resolution, 24 bpp true colour, STN/TFT display panels.
- SDIO interface supporting SPI, SD1, SD4 and SD8 mode with card detect, write protect, LED.
- 9 x 9 keyboard controller.
- Upto 62 GPIOs (multiplexed with peripheral I/Os), up to 22 with interrupt capability.
- Up to 512 timeslots, master or slave TDM. Any input timeslot can be switched to any output timeslot, and/or can be buffered for computation (up to 16 channels of 1 to 4 timeslots buffered during 32 ms). Up to 16 buffers can be played in output timeslots.
- 18 GPIOs for CODEC (up to 8 quad CODECS) & SLIC management.
- 1 bit DAC.
- I2S interface, full duplex with data buffer for left and right channels allowing up to 64ms of voice buffer (for 32 bit samples).

Note: The I2S and SDIO interfaces share the same RAM resources.

- Camera interface ITU-601 with external or embedded synchronization (ITU-656 or CSI2). Picture limit is given by the line length that must be stored in a 2048*32 buffer.
- Upto 8 additional I^2C/SPI chip selects.

4.12 Clock and reset system

- The system clocks are generated by three PLLs:
  - Two of them are fully programmable (the first one generates the clock for CPU and AMBA system; instead the second one generates the clock for the RAS block and for the DDR Memory interface. Both the PLLs offer an EMI reduction mode (Dithering) than can replace all traditional drop methods for Electro-Magnetic Interference.
  - The third PLL generates the clock for USB controllers.
- Several synthesizers provide different frequencies for the various IPs.
- Fully programmable control of clock and reset signals for all the slave blocks allowing sophisticated power management.

5 Pin description

The following tables describe the pinout of the SPEAr300 listed by functional block.

List of abbreviations:

PU = Pull Up

PD = Pull Down

5.1 Required external components

- DDR_COMP_1V8: place an external 121 kΩ resister between ball P4 and ball R4
- USB_TX_RTUNE: connect an external 43.2Ω pull-down resistor to ball k5
- DIGITAL_REXT: place an external 121 kΩ resistor between ball G4 and ball F4

5.2 Dedicated pins

Table 3. Master clock, RTC, Reset and 3.3 V comparator pin descriptions

Group	Signal name	Ball	Direction	Function	Pin type
Master Clock	MCLK_XI	P1	Input	24 MHz (typical) crystal in	Oscillator 2.5 V capable
	MCLK_XO	P2	Output	24 MHz (typical) crystal out	
RTC	RTC_XI	E2	Input	32 kHz crystal in	Oscillator 1.5 V capable
	RTC_XO	E1	Output	32 kHz crystal out	
Reset	MRESET#(1)	M17	Input	Main Reset	TTL Schmitt trigger input buffer, 3.3 V tolerant
3.3 V Comp.	DIGITAL_REXT	G4	Output	Configuration	Analog, 3.3 V capable
	DIGITAL_GND_REX	F4	Power	Power	Power

1. The minimum time for which MRESET should be low to reset the device is 1 millisecond.

Table 4. Power supply pin description

Group	Signal name	Ball	Value
DIGITAL GROUND	GND	G6, G7, G8, G9, G10, G11, H6, H7, H8, H9, H10, H11, J6, J7, J8, J9, J10, J11, K6, K7, K8, K9, K10, K11, L6, L7, L8, L9, L10, M8, M9, M10	0 V
ANALOG GROUND	AGND	F2, G1, J2, L1, L3, L5, N2, N4, P3, R3,N12	0 V
I/O	VDD3	F5, F6, F7, F10, F11, F12, G5, J12, K12, L12, M12	3.3 V
CORE	VDD	F8, F9, G12, H5, H12, J5, L11, M6, M7, M11	1.2 V

Table 4. Power supply pin description (continued)

Group	Signal name	Ball	Value
USB HOST0 PHY	HOST0_VDDbc	L2	2.5 V
	HOST0_VDDb3	K4	3.3 V
	HOST0_VDDbs	M3	1.2 V
USB HOST1 PHY	HOST1_VDDbc	K3	2.5 V
	HOST1_VDDb3	J1	3.3 V
	HOST1_VDDbs	M3	1.2 V
USB DEVICE PHY	DEVICE_VDDbc	N1	2.5 V
	DEVICE_VDDb3	N3	3.3 V
	HOST0_VDDbs	M3	1.2V
OSCI (master clock)	MCLK_VDD	R1	1.2V
	MCLK_VDD2v5	R2	2.5 V
PLL1	DITH1_AVDD	G2	2.5 V
PLL2	DITH2_AVDD	M4	2.5 V
DDR I/O	SSTL_VDDe	M5, N5, N6, N7, N8, N9, N10, N11	1.8 V
ADC	ADC_AVDD	N13	2.5 V
OSCI RTC	RTC VDD	F1	1.5 V

Table 5. Debug pin descriptions

Group	Signal name	Ball	Direction	Function	Pin type
DEBUG	TEST_0	K16	Input	Test[4:0] configuration ports. For functional mode, they have to be set to 00110.	TTL input buffer, 3.3 V tolerant, PD
	TEST_1	K15			
	TEST_2	K14			
	TEST_3	K13			
	TEST_4	J15			
	BOOT_SEL	J14		Reserved, to be fixed at high level	
	nTRST	L16	Input	Test reset input	TTL Schmitt trigger input buffer, 3.3 V tolerant, PU
	TDO	L15	Output	Test data output	TTL output buffer, 3.3 V capable 4 mA
	TCK	L17	Input	Test clock	TTL Schmitt trigger input buffer, 3.3 V tolerant, PU
	TDI	L14	Input	Test data input	
	TMS	L13	Input	Test mode select	

Table 6. Serial memory interface (SMI) pin description

Group	Signal name	Ball	Direction	Function	Pin type
SMI	SMI_DATAIN	M13	Input	Serial Flash input data	TTL Input Buffer 3.3 V tolerant, PU
	SMI_DATAOUT	M14	Output	Serial Flash output data	TTL output buffer 3.3 V capable 4 mA
	SMI_CLK	N17	I/O	Serial Flash clock	
	SMI_CS_0	M15	Output	Serial Flash chip select	
	SMI_CS_1	M16			

Table 7. USB pin descriptions

Group	Signal name	Ball	Direction	Function	Pin type
USB DEV	DEV_DP	M1	I/O	USB Device D+	Bidirectional analog buffer 5 V tolerant
	DEV_DM	M2		USB Device D-	
	DEV_VBUS	G3	Input	USB Device VBUS	TTL input buffer 3.3 V tolerant, PD
	HOST1_DP	H1	I/O	USB HOST1 D+	Bidirectional analog buffer 5 V tolerant
	HOST1_DM	H2		USB HOST1 D-	
USB HOST	HOST1_VBUS	H3	Output	USB Host1 VBUS	TTL output buffer 3.3 V capable, 4 mA
	HOST1_OVRC	J4	Input	USB Host1 Over-Current	TTL input buffer 3.3 V tolerant, PD
	HOST0_DP	K1	I/O	USB Host0 D+	Bidirectional analog buffer 5 V tolerant
	HOST0_DM	K2		USB Host0 D-	
	HOST0_VBUS	J3	Output	USB Host0 VBUS	TTL output buffer 3.3 V capable, 4 mA
	HOST0_OVRC	H4	Input	USB Host0 Over-current	TTL Input Buffer 3.3 V tolerant, PD
	USB_TXRTUNE	K5	Output	Reference resistor	Analog
	USB_ANALOG_TEST	L4	Output	Analog Test Output	Analog

Table 8. ADC pin description

Group	Signal name	Ball	Direction	Function	Pin type
ADC	AIN_0	N16	Input	ADC analog input channel	Analog buffer 2.5 V tolerant
	AIN_1	N15			
	AIN_2	P17			
	AIN_3	P16			
	AIN_4	P15			
	AIN_5	R17			
	AIN_6	R16			
	AIN_7	R15			
	ADC_VREFN	N14		ADC negative voltage reference	
	ADC_VREFP	P14		ADC positive voltage reference	

Table 9. DDR pin description

Group	Signal name	Ball	Direction	Function	Pin type
DDR	DDR_ADD_0	T2	Output	Address Line	SSTL_2/SSTL_18
	DDR_ADD_1	T1			
	DDR_ADD_2	U1			
	DDR_ADD_3	U2			
	DDR_ADD_4	U3			
	DDR_ADD_5	U4			
	DDR_ADD_6	U5			
	DDR_ADD_7	T5			
	DDR_ADD_8	R5			
	DDR_ADD_9	P5			
	DDR_ADD_10	P6			
	DDR_ADD_11	R6			
	DDR_ADD_12	T6			
	DDR_ADD_13	U6			
	DDR_ADD_14	R7			
	DDR_BA_0	P7	Output	Bank select	
	DDR_BA_1	P8			
	DDR_BA_2	R8			
	DDR_RAS	U8	Output	Row Add. Strobe	
	DDR_CAS	T8	Output	Col. Add. Strobe	
	DDR_WE	T7	Output	Write enable	
	DDR_CLKEN	U7	Output	Clock enable	
	DDR_CLK_P	T9	Output	Differential clock	Differential SSTL_2/SSTL_18
	DDR_CLK_N	U9			

Table 9. DDR pin description (continued)

Group	Signal name	Ball	Direction	Function	Pin type
DDR	DDR_CS_0	P9	Output	Chip Select	SSTL_2/SSTL_18
	DDR_CS_1	R9			
	DDR_ODT_0	T3	I/O	On-Die Termination Enable lines	
	DDR_ODT_1	T4			
	DDR_DATA_0	P11	I/O	Data Lines (Lower byte)	
	DDR_DATA_1	R11			
	DDR_DATA_2	T11			
	DDR_DATA_3	U11			
	DDR_DATA_4	T12			
	DDR_DATA_5	R12			
	DDR_DATA_6	P12			
	DDR_DATA_7	P13			
	DDR_DQS_0	U10	Output	Lower Data Strobe	Differential SSTL_2/SSTL_18
	DDR_nDQS_0	T10			
	DDR_DM_0	U12	Output	Lower Data Mask	SSTL_2/SSTL_18
	DDR_GATE_0	R10	I/O	Lower Gate Open	
	DDR_DATA_8	T17	I/O	Data Lines (Upper byte)	
	DDR_DATA_9	T16			
	DDR_DATA_10	U17			
	DDR_DATA_11	U16			
	DDR_DATA_12	U14			
	DDR_DATA_13	U13			
	DDR_DATA_14	T13			
	DDR_DATA_15	R13			
	DDR_DQS_1	U15	I/O	Upper Data Strobe	Differential SSTL_2/SSTL_18
	DDR_nDQS_1	T15			
	DDR_DM_1	T14	I/O	Upper Data Mask	SSTL_2/SSTL_18
	DDR_GATE_1	R14		Upper Gate Open	
	DDR_VREF	P10	Input	Reference Voltage	Analog
	DDR_MEM_COMP_GND	R4	Power	Return for Ext. Resistors	Power
	DDR_MEM_COMP_REXT	P4	Power	Ext. Resistor	Analog
	DDR2_EN	J13	Input	Configuration	TTL Input Buffer 3.3 V Tolerant, PU

5.3 Shared I/O pins (PL_GPIOs)

SPEAr300 devices feature, in the Reconfigurable Array Subsystem (RAS), specific sets of IPs as well as groups of software controllable GPIOs (that can be used alternatively). In the SPEAr300 the following IPs are implemented in the RAS:

- FSMC NAND/NOR Flash interface
- GPIO/Keyboard controller
- 8-bit camera interface
- CLCD controller interface
- Digital-to-analog converter (DAC)
- I2S
- 4 SPI/I2C control signals
- TDM block
- SDIO interface
- GPIOs

The 98 PL_GPIO and 4 PL_CLK pins have the following characteristics:

- Output buffer: TTL 3.3 V capable up to 10 mA
- Input buffer: TTL, 3.3 V tolerant, selectable internal pull up/pull down (PU/PD)

The PL_GPIOs can be configured in 13 different modes. This allows SPEAr300 to be tailored for use in various applications, see *Section 30.1*

5.3.1 PL_GPIO pin description

Table 10. PL_GPIO pin description

Group	Signal name	Ball	Direction	Function	Pin type
PL_GPIOs	PL_GPIO_97... PL_GPIO_0	(see the section *Table 13*)	I/O	General purpose I/O or multiplexed pins (see the section *Table 13*)	(see the introduction of the *Section 5.3* here above)
	PL_CLK1... PL_CLK4			programmable logic external clocks	

5.3.2 Alternate functions

Other peripheral functions are listed in the Alternate Functions column of *Table 13: PL_GPIO multiplexing scheme* and can be individually enabled/disabled via RAS register 1.

5.3.3 Boot pins

The status of the boot pins is read at startup by the BootROM.

Table 11. Booting pins

B3-B0	Boot device
0000	USB Device
0001	ETH - MAC address in I2C
0010	ETH - MAC address in SPI
0011	Serial NOR
0100	Parallel NOR 8bit (FSMC)
0101	Parallel NOR 16bit (FSMC)
0110	Nand 8bit
0111	Nand 16bit
1010	UART

Note: Other combinations are reserved.

5.3.4 GPIOs

Some PL_GPIO pins can be used as software controlled general purpose I/Os (GPIOs).

- 6 base GPIOs can be enabled as alternate functions on PL_GPIO.
- 18 GPIO are provided by the RAS IPs G8 and G10 on PL_GPIO.
- 18 GPIOs are available if the GPIO/ keyboard controller is configured in GPIO mode

5.3.5 Multiplexing scheme

The two multiplexers shown in *Figure 3.* are controlled by different registers. The first multiplexer selects the I/O functions of the RAS IPs in one of 13 modes shown in “Configuration mode” columns in *Table 13*). This selection is programmable via 4 bits in RAS register 2.

The second multiplexer is controlled by RAS register 1 and allows you to enable the I/O functions shown in alternate functions column of *Table 13*.

Figure 3. Multiplexing scheme

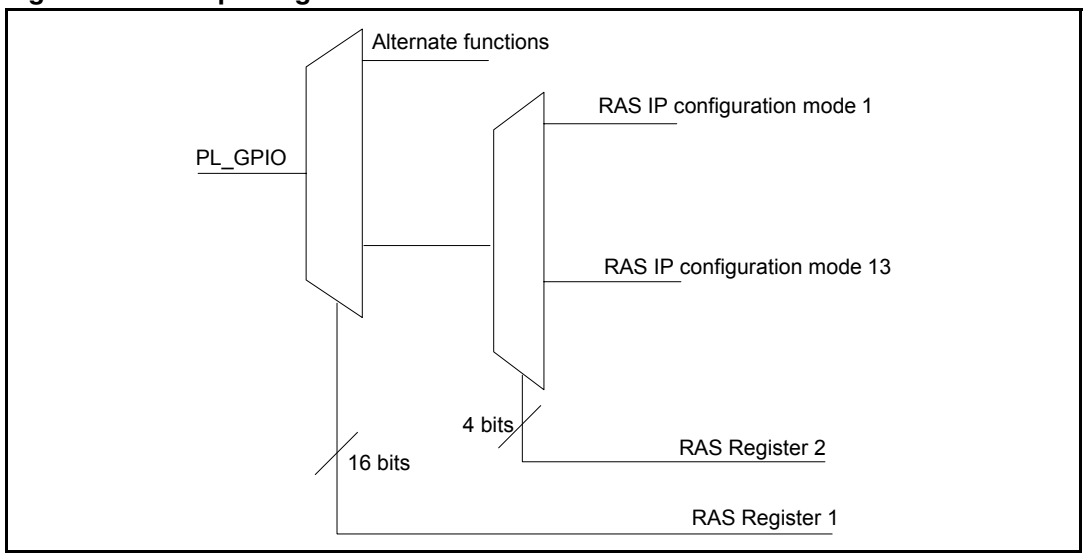


Table 12. Available peripherals in each configuration mode

Modes	FSMC	Boot pins	SPI/I2C Multi slave control	I2S	CLCD	DAC	Camera interface	TDM No of voice devices	SDIO/MMC data lines	Keyboard keys	GPIOs					
											Max. no. of I/Os	Bidirectional	Input only	Output only	Special outputs (sync)	Max. no. of I/Os with Interrupt
1	16-bit NAND	4									18	6	12			6
2	16-bit NOR	4									18	6	12			6
3	16-bit NAND				1			1	8		28	28				22
4			8	1		1		8	8	9*9	62	38	8	12	4	14
5			4	1	1	1		2	8	9*9	42	38		4		14
6			8	1		1		8	8	9*9	58	38	8	8	4	14
7			4	1	1	1		2	8	9*9	42	38		4		14
8	8-bit NOR		8					8	4		44	24	8	8	4	14
9	8-bit NAND /NOR		8	1		1		4	4		42	24	8	8	2	14
10			4	1		1	8-bit	2	8	9*9	36	32		4		10
11				1	1	1	14-bit	2	8	7*5	26	26				10
12				1		1	14-bit	2	8	7*5	26	26				10
13			4	1	1	1	8-bit	2	8	9*9	32	28		4		6

TDM interfacing using GPIOs

In some configuration modes where less than 8 TDM devices are indicated in *Table 12*, additional TDM devices can be supported by using GPIO pins. The TDM needs a dedicated interrupt line, an SPI and an independent frame sync signal to interface each device. When enough SPI chip selects signals are not available (SPI_I2C signals), the chip select can be performed by a GPIO. In this case the number of possible TDM devices supported is:

- Modes 5, 7, 8 and 9: up to 8 devices
- Modes 3 and 10: up to 6 devices
- Modes 11 and 12: up to 4 devices

Table 13. PL_GPIO multiplexing scheme

PL / pin number	Alternate function (enabled by RAS register 1)	Configuration mode (enabled by RAS register 2)												
		1	2	3	4	5	6	7	8	9	10	11	12	13
97/H16		/E1	/E1	/E1	1	1	1	1	/E1	/E1	1	1	1	1
96/H15		D0	DQ0	D0	COL0	COL0	COL0	COL0	D0	D0	COL0	COL0	COL0	COL0
95/H14		D1	DQ1	D1	COL1	COL1	COL1	COL1	D1	D1	COL1	COL1	COL1	COL1
94/H13		D2	DQ2	D2	COL2	COL2	COL2	COL2	D2	D2	COL2	COL2	COL2	COL2
93/G17		D3	DQ3	D3	COL3	COL3	COL3	COL3	D3	D3	COL3	COL3	COL3	COL3
92/G16		D4	DQ4	D4	COL4	COL4	COL4	COL4	D4	D4	COL4	COL4	COL4	COL4
91/G15		D5	DQ5	D5	COL5	COL5	COL5	COL5	D5	D5	COL5	DIO0_1	DIO0_1	COL5
90/G14		D6	DQ6	D6	COL6	COL6	COL6	COL6	D6	D6	COL6	DIO1_1	DIO1_1	COL6
89/F17		D7	DQ7	D7	COL7	COL7	COL7	COL7	D7	D7	COL7	DIO2_1	DIO2_1	COL7
88/F16		D8	DQ8	D8	COL8	COL8	COL8	COL8	G8_0	G8_0	COL8	DIO3_1	DIO3_1	COL8
87/G13		D9	DQ9	D9	ROW0	ROW0	ROW0	ROW0	G8_1	G8_1	ROW0	ROW0	ROW0	ROW0
86/E17		D10	DQ10	D10	ROW1	ROW1	ROW1	ROW1	G8_2	G8_2	ROW1	ROW1	ROW1	ROW1
85/F15		D11	DQ11	D11	ROW2	ROW2	ROW2	ROW2	G8_3	G8_3	ROW2	ROW2	ROW2	ROW2
84/D17		D12	DQ12	D12	ROW3	ROW3	ROW3	ROW3	G8_4	G8_4	ROW3	ROW3	ROW3	ROW3
83/E16		D13	DQ13	D13	ROW4	ROW4	ROW4	ROW4	G8_5	G8_5	ROW4	ROW4	ROW4	ROW4
82/E15		D14	DQ14	D14	ROW5	ROW5	ROW5	ROW5	G8_6	G8_6	ROW5	ROW5	ROW5	ROW5
81/C17		D15	DQ15 A-1	D15	ROW6	ROW6	ROW6	ROW6	G8_7	G8_7	ROW6	ROW6	ROW6	ROW6
80/D16		0	A0	CLD0	G8_0 (out)	CLD0	0	CLD0	A0	A0	Reserved	CLD0	Reserved	CLD0
79/F14		0	A1	CLD1	G8_1 (out)	CLD1	0	CLD1	A1	A1	Reserved	CLD1	Reserved	CLD1
78/D15		0	A2	CLD2	G8_2 (out)	CLD2	0	CLD2	A2	A2	Reserved	CLD2	Reserved	CLD2
77/B17		0	A3	CLD3	G8_3 (out)	CLD3	0	CLD3	A3	A3	Reserved	CLD3	Reserved	CLD3
76/F13		0	A4	CLD4	G8_4(out)	CLD4	0	CLD4	A4	A4	Reserved	CLD4	Reserved	CLD4
75/E14		0	A5	CLD5	G8_5 (out)	CLD5	0	CLD5	A5	A5	Reserved	CLD5	Reserved	CLD5
74/C16		0	A6	CLD6	G8_6 (out)	CLD6	0	CLD6	A6	A6	Reserved	CLD6	Reserved	CLD6

Table 13. PL_GPIO multiplexing scheme (continued)

PL / pin number	Alternate function (enabled by RAS register 1)	Configuration mode (enabled by RAS register 2)												
		1	2	3	4	5	6	7	8	9	10	11	12	13
73/A17		0	A7	CLD7	G8_7 (out)	CLD7	0	CLD7	A7	A7	Reserved	CLD7	Reserved	CLD7
72/B16		0	A8	CLD8	IT0	CLD8	IT0	CLD8	IT0	IT0	Reserved	CLD8	Reserved	CLD8
71/D14		0	A9	CLD9	IT1	CLD9	IT1	CLD9	IT1	IT1	Reserved	CLD9	Reserved	CLD9
70/C15		0	A10	CLD10	IT2	CLD10	IT2	CLD10	IT2	IT2	Reserved	CLD10	Reserved	CLD10
69/A16		0	A11	CLD11	IT3	CLD11	IT3	CLD11	IT3	IT3	Reserved	CLD11	Reserved	CLD11
68/B15		0	A12	CLD12	IT4	CLD12	IT4	CLD12	IT4	IT4	Reserved	CLD12	Reserved	CLD12
67/C14		0	A13	CLD13	IT5	CLD13	IT5	CLD13	IT5	IT5	Reserved	CLD13	Reserved	CLD13
66/E13		0	A14	CLD14	IT6	CLD14	IT6	CLD14	IT6	IT6	Reserved	CLD14	Reserved	CLD14
65/B14		0	A15	CLD15	IT7	CLD15	IT7	CLD15	IT7	IT7	Reserved	CLD15	Reserved	CLD15
64/D13		0	A16	CLD16	SPI_I2C4	CLD16	SPI_I2C4	CLD16	SPI_I2C4	SPI_I2C4	Reserved	CLD16	Reserved	CLD16
63/C13		0	A17	CLD17	SPI_I2C5	CLD17	SPI_I2C5	CLD17	SPI_I2C5	SPI_I2C5	Reserved	CLD17	Reserved	CLD17
62/A15		0	A18	CLD18	SPI_I2C6	CLD18	SPI_I2C6	CLD18	SPI_I2C6	SPI_I2C6	Reserved	CLD18	Reserved	CLD18
61/E12		0	A19	CLD19	SPI_I2C7	CLD19	SPI_I2C7	CLD19	SPI_I2C7 /DOUT	SPI_I2C7 /DOUT	Reserved	CLD19	Reserved	CLD19
60/A14		0	A20	CLD20	TDM_ SYNC4	CLD20	TDM_ SYNC4	CLD20	TDM_ SYNC4	TDM_ SYNC4	Reserved	CLD20	Reserved	CLD20
59/B13		0	A21	CLD21	TDM_ SYNC5	CLD21	TDM_ SYNC5	CLD21	TDM_ SYNC5	TDM_ SYNC5	Reserved	CLD21	Reserved	CLD21
58/D12		CL	A22	CL	TDM_ SYNC6	CLD22	TDM_ SYNC6	CLD22	TDM_ SYNC6	CL	Reserved	CLD22	Reserved	CLD22
57/E11		AL	A23	AL	TDM_ SYNC7	CLD23	TDM_ SYNC7	CLD23	TDM_ SYNC7	AL	Reserved	CLD23	Reserved	CLD23
56/C12		/W	/W	/W	ROW7	ROW7	ROW7	ROW7	/W	/W	ROW7	VSYNC_1	VSYNC_1	ROW7
55/A13		/R	/G	/R	ROW8	ROW8	ROW8	ROW8	/R	/R	ROW8	HSYNC_1	HSYNC_1	ROW8
54/E10		0	0	CLAC	G10_9	CLAC	G10_9	CLAC	G10_9	G10_9	G10_9	CLAC	G10_9	CLAC
53/D11		0	0	CLCP	G10_8	CLCP	G10_8	CLCP	G10_8	G10_8	G10_8	CLCP	G10_8	CLCP
52/B12		0	0	CLFP	G10_7	CLFP	G10_7	CLFP	G10_7	G10_7	G10_7	CLFP	G10_7	CLFP

Table 13. PL_GPIO multiplexing scheme (continued)

PL / pin number	Alternate function (enabled by RAS register 1)	Configuration mode (enabled by RAS register 2)												
		1	2	3	4	5	6	7	8	9	10	11	12	13
51/D10		0	0	CLLP	G10_6	CLLP	G10_6	CLLP	G10_6	G10_6	G10_6	CLLP	G10_6	CLLP
50/A12	TMR_CPTR4	0	0	CLLE	G10_5	CLLE	G10_5	CLLE	G10_5	G10_5	G10_5	CLLE	G10_5	CLLE
49/C11	TMR_CPTR3	0	0	CLPP	G10_4	CLPP	G10_4	CLPP	G10_4	G10_4	G10_4	CLPP	G10_4	CLPP
48/B11	TMR_CPTR2	B0	B0	CLD22	SPI_I2C0	SPI_I2C0	SPI_I2C0	SPI_I2C0	SPI_I2C0	SPI_I2C0	SPI_I2C0	DIO4_1	DIO4_1	SPI_I2C0
47/C10	TMR_CPTR1	B1	B1	CLD23	SPI_I2C1	SPI_I2C1	SPI_I2C1	SPI_I2C1	SPI_I2C1	SPI_I2C1	SPI_I2C1	DIO5_1	DIO5_1	SPI_I2C1
46/A11	TMR_CLK4	B2	B2	GPIO7	SPI_I2C2	SPI_I2C2	SPI_I2C2	SPI_I2C2	SPI_I2C2	SPI_I2C2	SPI_I2C2	DIO6_1	DIO6_1	SPI_I2C2
45/B10	TMR_CLK3	B3	B3	GPIO6	SPI_I2C3	SPI_I2C3	SPI_I2C3	SPI_I2C3	SPI_I2C3	SPI_I2C3	SPI_I2C3	DIO7_1	DIO7_1	SPI_I2C3
44/A10	TMR_CLK2	H0	H0	GPIO5	G10_3/O0	G10_3/O0	G10_3/O0	G10_3/O0	G10_3	DAC_O0	DAC_O0	DAC_O0	DAC_O0	DAC_O0
43/E9	TMR_CLK1	H1	H1	GPIO4	G10_2/O1	G10_2/O1	G10_2/O1	G10_2/O1	G10_2	DAC_O1	DAC_O1	DAC_O1	DAC_O1	DAC_O1
42/D9	UART_DTR	H2	H2	GPIO3	I2S_DIN	I2S_DIN	I2S_DIN	I2S_DIN	G10_1	I2S_DIN	I2S_DIN	I2S_DIN	I2S_DIN	I2S_DIN
41/C9	UART_RI	H3	H3	GPIO2	I2S_ LRCK	I2S_ LRCK	I2S_ LRCK	I2S_ LRCK	G10_0	I2S_LRCK	I2S_LRCK	I2S_LRCK	I2S_LRCK	I2S_LRCK
40/B9	UART_DSR	H4	H4	GPIO1	I2S_CLK	I2S_CLK	I2S_CLK	I2S_CLK	TDM_SYNC3	I2S_CLK	I2S_CLK	I2S_CLK	I2S_CLK	I2S_CLK
39/A9	UART_DCD	H5	H5	GPIO0	I2S_ DOUT	I2S_ DOUT	I2S_ DOUT	I2S_ DOUT	TDM_SYNC2	DOUT	I2S_DOUT	I2S_DOUT	I2S_DOUT	I2S_DOUT
38/A8	UART_CTS	H6	H6	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1	TDM_ SYNC1
37/B8	UART_RTS	H7	H7	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT	TDM_ DOUT
36/C8	SSP_CS4	0	0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0	TDM_ SYNC0
35/D8	SSP_CS3	Reserved	Reserved	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK	TDM_CLK
34/E8	SSP_CS2	0	0	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN	TDM_DIN
33/E7	BasGPIO5	0	0	SD_CMD	SD_CMD	SD_CMD	SD_CMD	SD_CMD	SD_CMD	SD_CMD	SD_CMD	SD_CMD	SD_CMD	SD_CMD
32/D7	BasGPIO4	0	0	SD_CLK	SD_CLK	SD_CLK	SD_CLK	SD_CLK	SD_CLK	SD_CLK	SD_CLK	SD_CLK	SD_CLK	SD_CLK
31/C7	BasGPIO3	0	0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0	SD_DAT0
30/B7	BasGPIO2	0	0	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1	SD_DAT1

Table 13. PL_GPIO multiplexing scheme (continued)

PL / pin number	Alternate function (enabled by RAS register 1)	Configuration mode (enabled by RAS register 2)												
		1	2	3	4	5	6	7	8	9	10	11	12	13
29/A7	BasGPIO1	0	0	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2	SD_DAT2
28/A6	BasGPIO0	0	0	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3	SD_SDAT3
27/B6	MII_TX_CLK	0	0	SD_SDAT4	SD_SDAT4	SD_SDAT4	SD_SDAT4	SD_SDAT4	G8_0	G8_0	SD_SDAT4	SD_SDAT4	SD_SDAT4	SD_SDAT4
26/A5	MII_TXD0	0	0	SD_SDAT5	SD_SDAT5	SD_SDAT5	SD_SDAT5	SD_SDAT5	G8_1	G8_1	SD_SDAT5	SD_SDAT5	SD_SDAT5	SD_SDAT5
25/C6	MII_TXD1	0	0	SD_SDAT6	SD_SDAT6	SD_SDAT6	SD_SDAT6	SD_SDAT6	G8_2	G8_2	SD_SDAT6	SD_SDAT6	SD_SDAT6	SD_SDAT6
24/B5	MII_TXD2	0	0	SD_SDAT7	SD_SDAT7	SD_SDAT7	SD_SDAT7	SD_SDAT7	G8_3	G8_3	SD_SDAT7	SD_SDAT7	SD_SDAT7	SD_SDAT7
23/A4	MII_TXD3	0	0	G8_4	G8_4	G8_4	G8_4	G8_4	G8_4	G8_4	G8_4	G8_4	G8_4	G8_4
22/D6	MII_TX_EN	0	0	G8_5	G8_5	G8_5	G8_5	G8_5	G8_5	G8_5	G8_5	G8_5	G8_5	G8_5
21/C5	MII_TX_ER	0	0	G8_6	G8_6	G8_6	G8_6	G8_6	G8_6	G8_6	DIO7	DIO8_1	DIO8_1	DIO7
20/B4	MII_RX_CLK	0	0	G8_7	G8_7	G8_7	G8_7	G8_7	G8_7	G8_7	DIO6	DIO9_1	DIO9_1	DIO6
19/A3	MII_RX_DV	0	0	G10_0	G10_0	G10_0	G10_0	G10_0	G10_0	G10_0	DIO5	DIO10_1	DIO10_1	DIO5
18/D5	MII_RX_ERR	0	0	G10_1	G10_1	G10_1	G10_1	G10_1	G10_1	G10_1	DIO4	DIO11_1	DIO11_1	DIO4
17/C4	MII_RXD0	0	0	G10_2	G10_2	G10_2	G10_2	G10_2	G10_2	G10_2	DIO3	DIO12_1	DIO12_1	DIO3
16/E6	MII_RXD1	0	0	G10_3	G10_3	G10_3	G10_3	G10_3	G10_3	G10_3	DIO2	DIO13_1	DIO13_1	DIO2
15/B3	MII_RXD2	0	0	G10_4	G10_4	G10_4	G10_4	G10_4	G10_4	G10_4	DIO1	G10_4	G10_4	DIO1
14/A2	MII_RXD3	0	0	G10_5	G10_5	G10_5	G10_5	G10_5	G10_5	G10_5	DIO0	G10_5	G10_5	DIO0
13/A1	MII_COL	0	0	G10_6	G10_6	G10_6	G10_6	G10_6	G10_6	G10_6	VSYNC	G10_6	G10_6	VSYNC
12/D4	MII_CRS	0	0	G10_7	G10_7	G10_7	G10_7	G10_7	G10_7	G10_7	HSYNC	G10_7	G10_7	HSYNC
11/E5	MII_MDC	0	0	G10_8	G10_8	G10_8	G10_8	G10_8	G10_8	G10_8	G10_8	G10_8	G10_8	G10_8
10/C3	MII_MDIO	0	0	G10_9	G10_9	G10_9	G10_9	G10_9	G10_9	G10_9	G10_9	G10_9	G10_9	G10_9
9/B2	SSP_MOSI	0	0	SD_CD	SD_CD	SD_CD	SD_CD	SD_CD	SD_CD	SD_CD	SD_CD	SD_CD	SD_CD	SD_CD
8/C2	SSP_SCLK	0	0	SD_WP	SD_WP	SD_WP	SD_WP	SD_WP	SD_WP	SD_WP	SD_WP	SD_WP	SD_WP	SD_WP
7/D3	SSP_SS	0	0	0	0	0	0	0	0	0	0	0	0	0
6/B1	SSP_MISO	0	0	SD_LED	SD_LED	SD_LED	SD_LED	SD_LED	SD_LED	SD_LED	SD_LED	SD_LED	SD_LED	SD_LED
5/D2	I2C_SDA	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13. PL_GPIO multiplexing scheme (continued)

PL / pin number	Alternate function (enabled by RAS register 1)	Configuration mode (enabled by RAS register 2)												
		1	2	3	4	5	6	7	8	9	10	11	12	13
4/C1	I2C_SCL	0	0	0	0	0	0	0	0	0	0	0	0	0
3/D1	UART_RX	/E4	/E4	/E4	1	1	1	1	1	1	1	1	1	1
2/E4	UART_TX	/E3	/E3	/E3	1	1	1	1	1	1	1	1	1	1
1/E3	IrDA_RX	/E2	/E2	/E2	1	1	1	1	1	1	1	1	1	1
0/F3	IrDA_TX	R/B	R/B	R/B	1	1	1	1	R/B	R/B	1	1	1	1
CK1/K17	PL_CLK1	TCLK*	TCLK*	CCLK/ TCLK*	TCLK*	CCLK/TCLK*	TCLK*	CCLK/ TCLK*	TCLK*	TCLK*	TCLK*	CCLK/ TCLK*	TCLK*	CCLK/ TCLK*
CK2/J17	PL_CLK2	Reserved	Reserved	int_CLK	int_CLK	int_CLK	int_CLK	int_CLK	int_CLK	int_CLK	int_CLK	int_CLK	int_CLK	int_CLK
CK3/J16	PL_CLK3	Reserved	Reserved	\int_CLK	\int_CLK	\int_CLK	\int_CLK	\int_CLK	\int_CLK	\int_CLK	CLK	CLK	CLK	CLK
CK4/H17	PL_CLK4	Reserved	Reserved	2.048 MHz	2.048 MHz	2.048 MHz	2.048 MHz	2.048 MHz	2.048 MHz	2.048 MHz	PCLK	PCLK	PCLK	PCLK

Notes/legend for *Table 13*:

GPIO (General purpose I/O):

- basGPIO: Base GPIOs in the basic subsystem (enabled as alternate functions)
- G10 and G8: GPIOs in the telecom subsystem
- GPIOx: GPIOs in the independent GPIO block in the RAS subsystem

TDM_: DM interface signals

SD_: SDIO interface

IT pins: interrupts

Table cells filled with '0' or '1' are unused and unless otherwise configured as Alternate function or GPIO, the corresponding pin is held at low or high level respectively by the internal logic.

Table cells filled with 'Reserved' denote pins that must be left unconnected.

Table 14. Table shading

Shading	Pin group
FSMC	FSMC pins: NAND or NOR Flash
Keyboard	Keyboard pins ROWs are outputs, COLs are inputs
CLCD	Colur LCD controller pins
CAMERA	Camera pins
UART	UART pins
Ethernet MAC	MII/SMII Ethernet MAC Pins
SDIO/MMC	SD card controller pins
GPT	Timer pins
IrDA	IrDA pins
SSP	SSP pins
I2C	I2C pins

5.4 PL_GPIO pin sharing for debug modes

In some cases the PL_GPIO pins may be used in different ways for debugging purposes. There are three different cases (see also *Table 15*):

1. Case 1 - All the PL_GPIO get values from Boundary scan registers during Ex-test instruction of JTAG. Typically this configuration is used to verify correctness of the soldering process during the production flow.
2. Case 2 - All the PL_GPIO maintain their original meaning but the JTAG Interface is connected to the processor. This configuration is useful during the development phase but offers only "static" debug.
3. Case 3 - Some PL_GPIO, as shown in *Table 15: Ball sharing during debug*, are used to connect the ETM9 lines to an external box. This configuration is typically used only during the development phase. It offers a very powerful debug capability. When the processor reaches a breakpoint it is possible, by analyzing the trace buffer, to understand the reason why the processor has reached the break.

Table 15. Ball sharing during debug

Signal	Case 1 - Board debug	Case 2 - Static debug	Case 3 - Full debug
Test [0]	0	1	0
Test [1]	0	0	1
Test [2]	0	0/1	0/1
Test [3]	0	0/1	0/1
Test [4]	1	0	0
nTRST	nTRST_bscan	nTRST_ARM	nTRST_ARM
TCK	TCK_bscan	TCK_ARM	TCK_ARM
TMS	TSM_bscan	TMS_ARM	TSM_ARM
TDI	TDI_bscan	TDI_ARM	TDI_ARM
TDO	TDO_bscan	TDO_ARM	TDO_ARM
PL_GPIO [97]	BSR Value	Functional I/O	ARM_TRACE_CLK
PL_GPIO [96]	BSR Value	Functional I/O	ARM_TRACE_PKTA[0]
PL_GPIO [95]	BSR Value	Functional I/O	ARM_TRACE_PKTA[1]
PL_GPIO [94]	BSR Value	Functional I/O	ARM_TRACE_PKTA[2]
PL_GPIO [93]	BSR Value	Functional I/O	ARM_TRACE_PKTA[3]
PL_GPIO [92]	BSR Value	Functional I/O	ARM_TRACE_PKTB[0]
PL_GPIO [91]	BSR Value	Functional I/O	ARM_TRACE_PKTB[1]
PL_GPIO [90]	BSR Value	Functional I/O	ARM_TRACE_PKTB[2]
PL_GPIO [89]	BSR Value	Functional I/O	ARM_TRACE_PKTB[3]
PL_GPIO [88]	BSR Value	Functional I/O	ARM_TRACE_SYNCA
PL_GPIO [87]	BSR Value	Functional I/O	ARM_TRACE_SYNCB
PL_GPIO [86]	BSR Value	Functional I/O	ARM_PIPESTATA[0]
PL_GPIO [85]	BSR Value	Functional I/O	ARM_PIPESTATA[1]
PL_GPIO [84]	BSR Value	Functional I/O	ARM_PIPESTATA[2]
PL_GPIO [83]	BSR Value	Functional I/O	ARM_PIPESTATB[0]
PL_GPIO [82]	BSR Value	Functional I/O	ARM_PIPESTATB[1]
PL_GPIO [81]	BSR Value	Functional I/O	ARM_PIPESTATB[2]
PL_GPIO [80]	BSR Value	Functional I/O	ARM_TRACE_PKTA[4]
PL_GPIO [79]	BSR Value	Functional I/O	ARM_TRACE_PKTA[5]
PL_GPIO [78]	BSR Value	Functional I/O	ARM_TRACE_PKTA[6]
PL_GPIO [77]	BSR Value	Functional I/O	ARM_TRACE_PKTA[7]
PL_GPIO [76]	BSR Value	Functional I/O	ARM_TRACE_PKTB[4]
PL_GPIO [75]	BSR Value	Functional I/O	ARM_TRACE_PKTB[5]
PL_GPIO [74]	BSR Value	Functional I/O	ARM_TRACE_PKTB[6]

Table 15. Ball sharing during debug (continued)

Signal	Case 1 - Board debug	Case 2 - Static debug	Case 3 - Full debug
PL_GPIO [73]	BSR Value	Functional I/O	ARM_TRACE_PKTB[7]
PL_GPIO [72:0]			

6 Memory map

Table 16. Main memory map

Start address	End address	Notes
0x0000.0000	0x3FFF.FFFF	DDR2 or Low Power DDR
0x4000.0000	0xBFFF.FFFF	*Table 22: Reconfigurable array subsystem*
0xC000.0000	0xCFFF.FFFF	Reserved
0xD000.0000	0xD7FF.FFFF	*Table 18: Low speed subsystem*
0xD800.0000	0xDFFF.FFFF	*Table 21: Application subsystem*
0xE000.0000	0xE7FF.FFFF	*Table 20: High speed subsystem*
0xE800.0000	0xEFFF.FFFF	Reserved
0xF000.0000	0xF7FF.FFFF	*Table 17: Multi layer CPU subsystem*
0xF800.0000	0xFFFF.FFFF	*Table 19: Basic subsystem*

Table 17. Multi layer CPU subsystem

Start address	End address	Peripheral	Notes	Bus
0xF000.0000	0xF00F.FFFF	Timer		APB
0xF010.0000	0xF10F.FFFF	-	Reserved	-
0xF110.0000	0xF11F.FFFF	VIC		AHB
0xF120.0000	0xF7FF.FFFF	-	Reserved	-

Table 18. Low speed subsystem

Start address	End address	Peripheral	Notes	Bus
0xD000.0000	0xD007.FFFF	UART		APB
0xD008.0000	0xD00F.FFFF	ADC		APB
0xD010.0000	0xD017.FFFF	SSP		APB
0xD018.0000	0xD01F.FFFF	I2C		APB
0xD020.0000	0xD07F.FFFF	-	Reserved	-
0xD080.0000	0xD0FF.FFFF	JPEG codec		AHB
0xD100.0000	0xD17F.FFFF	IrDA		AHB
0xD180.0000	0xD1FF.FFFF	-	Reserved	-
0xD280.0000	0xD7FF.FFFF	SRAM	Static ram shared memory (56 Kbyte)	AHB

Table 19. Basic subsystem

Start address	End address	Peripheral	Notes
0xF800.0000	0xFBFF.FFFF	Serial Flash memory	
0xFC00.0000	0xFC1F.FFFF	Serial Flash controller	
0xFC20.0000	0xFC3F.FFFF	-	reserved
0xFC40.0000	0xFC5F.FFFF	DMA controller	
0xFC60.0000	0xFC7F.FFFF	SDRAM controller	
0xFC80.0000	0xFC87.FFFF	Timer 1	
0xFC88.0000	0xFC8F.FFFF	Watch dog timer	
0xFC90.0000	0xFC97.FFFF	Real time clock	
0xFC98.0000	0xFC9F.FFFF	General purpose I/O	
0xFCA0.0000	0xFCA7.FFFF	System controller	
0xFCA8.0000	0xFCAF.FFFF	Miscellaneous registers	
0xFCB0.0000	0xFCB7.0000	Timer 2	
0xFCB8.0000	0xFEFF.FFFF	-	Reserved
0xFF00.0000	0xFFFF.FFFF	Internal Rom	Boot ROM

Table 20. High speed subsystem

Start address	End address	Peripheral	Notes	Bus
0xE000.0000	0xE07F.FFFF	-	Reserved	APB
0xE080.0000	0xE0FF.FFFF	Ethernet ctrl	MAC	AHB
0xE100.0000	0xE10F.FFFF	USB2.0 device	FIFO	AHB
0xE110.0000	0xE11F.FFFF	USB2.0 device	Configuration registers	AHB
0xE120.0000	0xE12F.FFFF	USB2.0 device	Plug detect	AHB
0xE130.0000	0xE17F.FFFF	-	Reserved	AHB
0xE180.0000	0xE18F.FFFF	USB2.0 EHCI 0/1		AHB
0xE190.0000	0xE19F.FFFF	USB2.0 OHCI 0		AHB
0xE1A0.0000	0xE20F.FFFF	-	Reserved	AHB
0xE210.0000	0xE21F.FFFF	USB2.0 OHCI 1		AHB
0xE220.0000	0xE27F.FFFF	-	Reserved	AHB
0xE280.0000	0xE28F.FFFF	ML USB ARB	Configuration register	AHB
0xE281.0000	0xE7FF.FFFF	-	Reserved	AHB

Table 21. Application subsystem

Start address	End address	Peripheral	Notes	Bus
0xD800.0000	0xD8FF.FFFF		Reserved	
0xD900.0000	0xD97F.FFFF		C3 coprocessor	AHB
0xD980.0000	0xDFFF.FFFF		Reserved	

Table 22. Reconfigurable array subsystem

Base Address	Address Space	IP
0x50000000	0x5000_0000 - 0x5FFF_FFFF	TELECOM
0x60000000	0x6000_0000 - 0x6FFF_FFFF	CLCD
0x70000000	0x7000_0000 - 0x7FFF_FFFF	SDIO
0x80000000	0x8000_0000 - 0x98FF_FFFF	FSMC
0x80000000	0x8000_0000 - 0x83FF_FFFF	FSMC NAND on PCBank0
0x84000000	0x8400_0000 - 0x87FF_FFFF	FSMC NAND on PCBank1
0x88000000	0x8800_0000 - 0x8BFF_FFFF	FSMC NAND on PCBank2
0x8C000000	0x8C00_0000 - 0x8FFF_FFFF	FSMC NAND on PCBank3
0x90000000	0x9000_0000 - 0x90FF_FFFF	FSMC NOR 0
0x91000000	0x9100_0000 - 0x91FF_FFFF	FSMC NOR 1
0x92000000	0x9200_0000 - 0x92FF_FFFF	FSMC NOR 2
0x93000000	0x9300_0000 - 0x93FF_FFFF	FSMC NOR 3
0x94000000	0x9400_0000 - 0x98FF_FFFF	FSMC Registers
0x99000000	0x9900_0000 - 0x9FFF_FFFF	RAS Registers
0xA0000000	0xA000_0000 - 0xA8FF_FFFF	Keyboard
0xA9000000	0xA900_0000 - 0xAFFF_FFFF	GPIO

7 CPU subsystem_ARM926EJ-S

7.1 Overview

The ARM926EJ-S is a powerful processor, targeted for multi-tasking applications.

Belonging to ARM9 general purposes family microprocessor, its main outstanding feature is its Memory management unit, which provides virtual memory features, making it also compliant with advanced operating system, like Linux.

The ARM926EJ-S supports the 32 bit ARM and 16 bit thumb instruction sets, enabling the user to trade off between high performance and high code density and includes features for efficient execution of byte code Java mode.

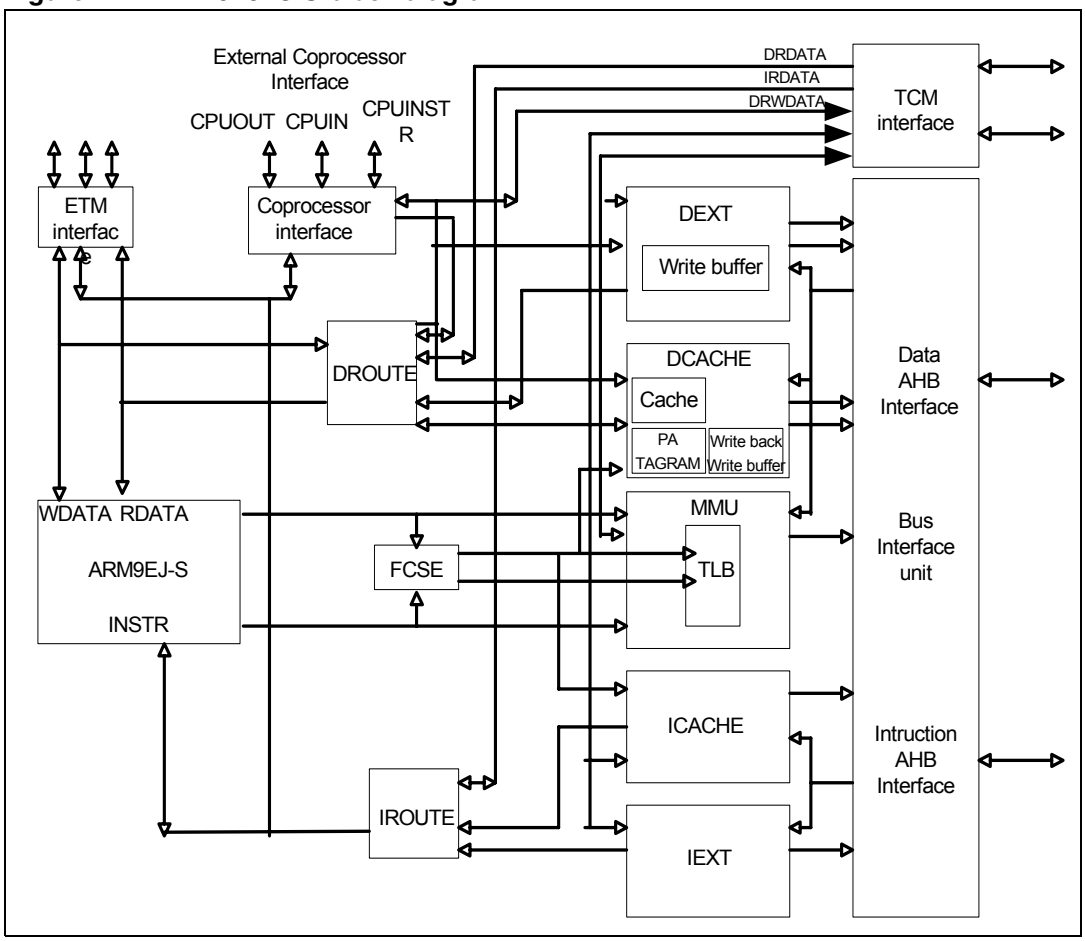
Additionally, it has the ARM debug architecture and includes logic to assist in software debug.

Main features are:

- Max CPU frequency 333 MHz (downward scalable).
- MMU.
- 16 Kbyte of instruction cache + 16 Kbyte of data cache.
- AMBA bus interface.
- JTAG and ETM9 (embedded trace macro-cell) for debug, large size version.

7.2 Functional description

Figure 4. ARM926EJ-S block diagram



Note: Co-processor interface and TCM interface are not used.

7.3 Main function description

7.3.1 Memory management unit

A single set of two-level page tables stored in main memory is used to control the address translation, permission checks, and memory region attributes for both data and instruction accesses.

The Memory Management Unit (MMU) uses a single unified Translation Look aside Buffer (TLB) to cache the information held in the page tables. To support both sections and pages, there are two levels of address translation, and the MMU puts the translated physical addresses into the MMU TLB.

The MMU TLB consists of two parts:

- The main TLB, which is a two-way, set-associative cache for page table information. It has 32 entries per way for a total of 64 entries.
- The lockdown TLB, which is an eight-entry fully-associative cache that contains locked TLB entries. Locking TLB entries can ensure that a memory access to a given region never incurs the penalty of a page table walk.

The MMU features are:

- Standard ARM architecture v4 and v5 MMU mapping sizes, domains, and access protection scheme,
- Mapping sizes are 1 MB (sections), 64 KB (large pages), 4 KB (small pages), and 1 KB (tiny pages),
- Access permissions for large pages and small pages can be specified separately for each quarter of the page (subpage permissions),
- Hardware page table walks,
- Invalidate entire TLB using CP15 c8,
- Invalidate TLB entry selected by MVA, using CP15 c8,
- Lockdown of TLB entries using CP15 c10.

7.3.2 Caches and write buffer

The ARM926EJ-S processor includes an Instruction Cache (ICache), a Data Cache (DCache) and a write buffer of 16 KB each. The size of the caches are 16 KB Instruction Cache and 16 KB Data Cache.

The caches have the following features:

- Virtual index, virtual tag, addressed using the Modified Virtual Address (MVA),
- Four-way set associative, with a cache line length of 32 bytes per line, and with two dirty bits in the DCache,
- DCache supports write-through and write-back (or copyback) cache operations,
- Allocate on read-miss is supported. The caches perform critical-word first cache refilling,
- Pseudo-random or round-robin replacement selectable,
- Cache lockdown registers enable control over which cache ways are used for allocation on a linefill, providing a mechanism for both lockdown and controlling cache pollution,
- The DCache stores the Physical Address (PA) tag,
- PLD data preload instruction does not cause data cache linefills,
- Maintenance operations to provide efficient invalidation of the entire DCache or ICache, regions of the two caches or region of virtual memory,
- Provide operations for efficient cleaning and invalidation of entire DCache, regions of it and regions of virtual memory.

The latter allows DCache coherency to be efficiently maintained when small code changes occur, for example for self-modifying code and changes to exception vectors.

7.3.3 Bus interface unit

The ARM926EJ-S Bus Interface Unit (BIU) arbitrates and schedules the AHB requests.

The BIU contains separate masters for both instruction and data access, enabling multi-layer AHB and multi-AHB systems to be implemented, giving the benefit of increased overall bus bandwidth and a more flexible system architecture.

To increase system performance, write buffers are used to prevent AHB writes stalling the ARM926EJ-S system.

8 CPU subsystem_Vectored interrupt controller (VIC)

8.1 Overview

Acting as an interrupt controller, the VIC determines the source that is requesting service and where its interrupt service routine (ISR) is loaded, doing that in hardware. In particular, the VIC supplies the starting address, or vector address, of the ISR corresponding to the highest priority requesting interrupt source.

Main features of the VIC are:

- Support for 32 standard interrupt sources.
- Generation of both Fast Interrupt reQuest (FIQ) and Interrupt ReQuest (IRQ), according to ARM system operation. IRQ is used for general interrupts, whereas FIQ is intended for fast, low-latency interrupt handling. In particular, using a single FIQ source at a time in the system provides interrupt latency reduction, because the ISR can be directly executed without determining the source of the interrupt.
- Support for 16 vectored interrupts (IRQ only).
- Hardware interrupt priority, where FIQ interrupt has the highest priority, followed by vectored IRQ interrupts (from vector 0 to vector 15), and then non-vectored IRQ interrupts with the lowest priority.
- Interrupt masking.
- Interrupts request status and raw interrupt status (prior to masking).
- Software interrupt generation.
- An AHB slave to connect to the CPU.

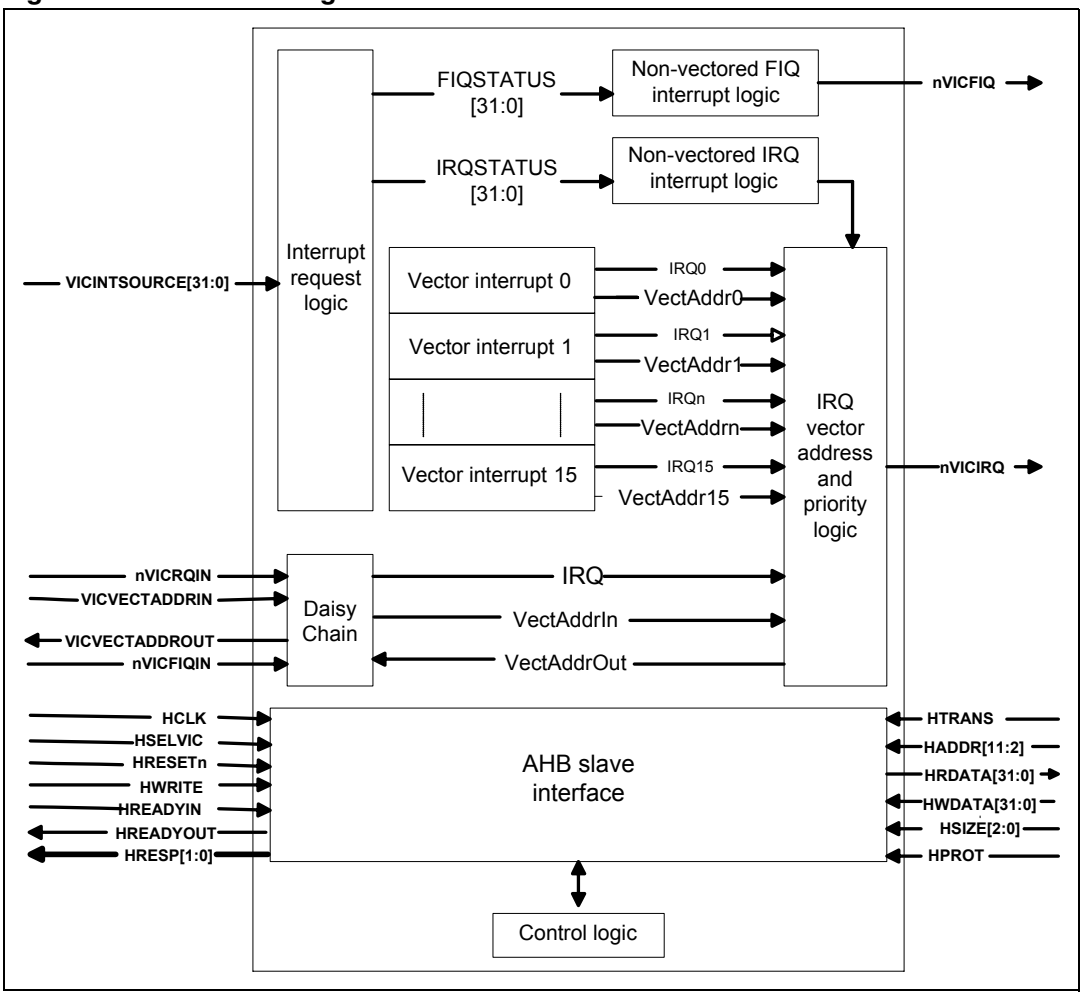
The interrupt inputs must be level sensitive, active HIGH, and held asserted until the interrupt service routine clears the interrupt. Edge-triggered interrupts are not compatible. The interrupt inputs do not have to be synchronous to HCLK.

Note: The VIC does not handle interrupt sources with transient behaviour.

8.2 Block diagram

Figure 5 shows the block diagram of VIC.

Figure 5. VIC block diagram



8.3 Main functions description

8.3.1 Interrupt request logic

The interrupt request logic block receive the interrupt requests from peripheral and combines them with the software interrupt requests (see *Section 8.3.6*). After that, it masks out the requests that are not enabled (through VICINTENABLE register, *Section 8.6.7: VICINTENABLE register*) and roots the others to either IRQSTATUS or FIQSTATUS depending on VICINTSELECT (see *Section 8.6.6: VICINTSELECT register*).

FIQ interrupts have the highest priority, followed by vectored interrupts (0-15) and, finally, non vectored interrupts.

8.3.2 Non-vectored FIQ interrupt logic

The non-vectored FIQ interrupt logic block generates the FIQ interrupt signal by combining the FIQ interrupt requests coming from the interrupt request logic and any requests from daisy-chained interrupt controllers.

8.3.3 Non-vectored IRQ interrupt logic

The non-vectored IRQ Interrupt Logic block generates the non-vectored IRQ interrupt signal by combining the non-vectored IRQ interrupt requests coming from the interrupt request logic. This signal is then sent to the IRQ vector address and priority logic.

8.3.4 Vectored interrupts

As depicted in *Figure 5*, there are 16 vectored interrupt blocks within the VIC. Each vectored interrupt block receives the IRQ interrupt requests from the interrupt request logic block and it generates a vectored IRQ interrupt.

In particular, a vectored IRQ interrupt is generated only if:

- It is enabled in the VICINTENABLE register (*Section 8.6.7: VICINTENABLE register*),
- It is set to generate an IRQ interrupt in the VICINTSELECT register (*Section 8.6.6: VICINTSELECT register*),
- It is enabled in the relevant VICVECTCNTL register (*Section 8.6.15: VICVECTCNTL register*),
- It is currently the highest requesting interrupt (vector 0 to vector 15, highest to lowest).

Besides, each vectored interrupt block is associated to the 32 bit address of the ISR to be executed. These ISR addresses are mapped in the VICVECTADDRi (with i = 0...15) registers (*Section 8.6.14: VICVECTADDR register*). The VICVECTADDR register (*Section 8.6.12: VICVECTADDR register*) contains the ISR address for the currently active IRQ interrupt.

8.3.5 Interrupt priority logic

The interrupt priority logic block organizes the following requests according to their priority:

- Non-vectored interrupt requests,
- Vectored interrupt requests,
- External interrupt requests.

If the interrupt is not currently being serviced, the highest priority request generates an IRQ interrupt.

Note: External interrupt is not being used in SPEAr300.

8.3.6 Software interrupts

The software can control the source interrupt lines to generate software interrupts (VICSOFTINT register, *Section 8.6.9: VICSOFTINT register*). These interrupts are generated before interrupt masking within the Interrupt Request Logic block, in the same way as external source interrupts.

It is possible to clear software interrupts by writing to the VICSOFTINTCLEAR register (*Section 8.6.10: VICSOFTINTCLEAR register*). This is normally done at the end of the ISR.

8.3.7 AHB slave interface

The AHB Slave Interface block connects the VIC to the CPU through the AHB bus.

8.4 Interrupt connection table

Table 23. Interrupt sources

Interrupt sources	IRQ #
Reserved	0
Generic Interrupt #3 from RAS	1
CPU Subsystem Timer 1_1	2
CPU Subsystem Timer 1_2	3
Basic Subsystem Timer 1_1	4
Basic Subsystem Timer 1_2	5
Basic Subsystem Timer 2_1	6
Basic Subsystem Timer 2_2	7
Basic Subsystem DMA (**D**irect **M**emory **A**ccess)	8
Basic Subsystem SMI (**S**erial **M**emory **I**nterface)	9
Basic Subsystem RTC (**R**eal **T**ime **C**lock)	10
Basic Subsystem GPIO (**G**eneral **P**urpose **I**nput **O**utput)	11
Basic Subsystem WD (**W**atch**D**og **T**imer)	12
DDR Controller	13
System Error	14
Wake-up FIQ from RCG	15
Low Speed Subsystem JPEG (**J**oint **P**hotographic **E**xperts **G**roup)	16
Low Speed Subsystem IrDA (**I**nfra**r**ed **D**ata **A**ssociation)	17
Low Speed Subsystem ADC (**A**nalog to **D**igital **C**onverter)	18
Low Speed Subsystem UART (**U**niversal **A**synchronous **R**eceiver and **T**ransmitter)	19
Low Speed Subsystem SPI (**S**erial **P**eripheral **I**nterface bus)	20
Low Speed Subsystem I^2C (**I**nter-**I**ntegrated **C**ircuit)	21
High Speed Subsystem Ethernet MAC (**M**edium **A**ccess **C**ontrol) Power Management Event	22
High Speed Subsystem Ethernet MAC (**M**edium **A**ccess **C**ontrol)	23
High Speed Subsystem USB Device (**U**niversal **S**erial **B**us)	24
High Speed Subsystem USB Host 1 OHCI (**O**pen **H**ost **C**ontroller **I**nterface)	25
High Speed Subsystem USB Host EHCI (**Enhanced Host C**ontroller **I**nterface)	26
High Speed Subsystem USB Host 2 OHCI (**O**pen **H**ost **C**ontroller **I**nterface)	27
Generic Interrupt #0 from RAS	28
Generic Interrupt #1 from RAS	29
Generic Interrupt #2 from RAS	30
Application Subsystem C3 (**C**hannel **C**ontrol **C**oprocessor)	31

8.5 How to reduce interrupt latency

The interrupt latency depends on the type of interrupt, see *Table 24*.

Table 24. Interrupt latency for different types of interrupts

Event	Worst case (cycles)		
	FIQ	IRQ	IRQ (reduced latency)
Interrupt synchronization	3	3	3
Worst case interrupt disable period	7	10	10
Entry to first instruction	2	2	2
Nesting, assuming single-state AHB	-	10	-
Load IRQ vector to PC	-	-	5
Total	12	25	20

To reduce interrupt latency, you can re-enable the IRQ interrupts in the processor after the ISR is entered, so the current ISR is interrupted, and the higher-priority ISR is executed. The VIC then only enables a higher priority interrupt than the interrupt currently being serviced. If a higher priority interrupt goes active, the current ISR is interrupted and the higher-priority ISR is executed. Before the interrupt enable bits in the processor can be re-enabled, the LR and SPSR must be saved, preferably on a software stack. When the ISR is exited, you must disable the interrupts, reload the LR and SPSR, and write to the vector address register, VICVECTADDR.

8.6 Programming model

8.6.1 Register map

The VIC can be fully configured by programming its 32 bit wide registers which can be accessed at the base addresses 0xF110_0000.

VIC registers can be logically divided in four main groups:

- Interrupt control and status registers (listed in *Table 25*), for interrupt configuration.
- Vector address registers (listed in *Table 26*), containing the address of the ISRs.
- Vector control registers (listed in *Table 27*), which select the interrupt source for the vectored interrupt.
- Identification registers (listed in *Table 28*), namely eight 8 bit RO registers reporting VIC-specific information (part number, revision number and so on). Refer to ARM technical documentation for further details.

Note: Offset addresses from 0x300 to 0x310 are reserved for test purposes.

Table 25. VIC interrupt control registers summary

Name	Offset	Type	Reset value	Description
VICIRQSTATUS	0x000	RO	32'h0	IRQ status
VICFIQSTATUS	0x004	RO	32'h0	FIQ status

Table 25. VIC interrupt control registers summary (continued)

Name	Offset	Type	Reset value	Description
VICRAWINTR	0x008	RO	-	Raw interrupt status
VICINTSELECT	0x00C	RW	32'h0	Interrupt select
VICINTENABLE	0x010	RW	32'h0	Interrupt enable
VICINTENCLEAR	0x014	WO	-	Interrupt enable clear
VICSOFTINT	0x018	RW	32'h0	Software interrupt
VICSOFTINTCLEAR	0x01C	WO	-	Software interrupt clear
VICPROTECTION	0x020	RW	32'h0	Protection enable

Table 26. VIC vector address registers summary

Name	Offset	Type	Reset value	Description
VICVECTADDR	0x030	RW	32'h0	Vector address
VICDEFVECTADDR	0x034	RW	32'h0	Default vector address
VICVECTADDR0	0x100	RW	32'h0	Vector address registers
VICVECTADDR1	0x104	RW	32'h0	
VICVECTADDR2	0x108	RW	32'h0	
VICVECTADDR3	0x10C	RW	32'h0	
VICVECTADDR4	0x110	RW	32'h0	
VICVECTADDR5	0x114	RW	32'h0	
VICVECTADDR6	0x118	RW	32'h0	
VICVECTADDR7	0x11C	RW	32'h0	
VICVECTADDR8	0x120	RW	32'h0	
VICVECTADDR9	0x124	RW	32'h0	
VICVECTADDR10	0x128	RW	32'h0	
VICVECTADDR11	0x12C	RW	32'h0	
VICVECTADDR12	0x130	RW	32'h0	
VICVECTADDR13	0x134	RW	32'h0	
VICVECTADDR14	0x138	RW	32'h0	
VICVECTADDR15	0x13C	RW	32'h0	

Table 27. VIC interrupt vector control registers summary

Name	Offset	Type	Reset value	Description
VICVECTCNTL0	0x200	RW	32'h0	Vector Control.
VICVECTCNTL1	0x204	RW	32'h0	
VICVECTCNTL2	0x208	RW	32'h0	
VICVECTCNTL3	0x20C	RW	32'h0	
VICVECTCNTL4	0x210	RW	32'h0	
VICVECTCNTL5	0x214	RW	32'h0	
VICVECTCNTL6	0x218	RW	32'h0	
VICVECTCNTL7	0x21C	RW	32'h0	
VICVECTCNTL8	0x220	RW	32'h0	
VICVECTCNTL9	0x224	RW	32'h0	
VICVECTCNTL10	0x228	RW	32'h0	
VICVECTCNTL11	0x22C	RW	32'h0	
VICVECTCNTL12	0x230	RW	32'h0	
VICVECTCNTL13	0x234	RW	32'h0	
VICVECTCNTL14	0x238	RW	32'h0	
VICVECTCNTL15	0x23C	RW	32'h0	

Table 28. VIC identification registers summary

Name	Offset	Type	Reset value	Description
VICPERIPHID0	0xFE0	RO	8'h90	Peripheral Identification.
VICPERIPHID1	0xFE4	RO	8'h11	
VICPERIPHID2	0xFE8	RO	8'h04	
VICPERIPHID3	0xFEC	RO	8'h00	
VICPCELLID0	0xFF0	RO	8'h0D	Identification Registers
VICPCELLID1	0xFF4	RO	8'hF0	
VICPCELLID2	0xFF8	RO	8'h05	
VICPCELLID3	0xFFC	RO	8'hB1	

8.6.2 Register description

8.6.3 VICIRQSTATUS register

The VICIRQSTATUS is the RO register which provides the status of interrupts after IRQ masking (through VICINTENABLE and VICINTSELECT registers, *Section 8.6.7: VICINTENABLE register* and *Section 8.6.6: VICINTSELECT register* respectively), at the output of the Interrupt Request Logic block (*Section 8.3.1*). The VICIRQSTATUS bit assignments are given in *Table 29*.

Table 29. VICIRQSTATUS register bit assignments

Bit	Name	Reset value	Description
[31:00]	IRQStatus	32'h0	Each bit is associated to an interrupt. If a bit is set, it indicates that the relevant interrupt is active, and generates an interrupt to the processor.

8.6.4 VICFIQSTATUS register

The VICFIQSTATUS is the RO register which provides the status of the interrupts after FIQ masking (through VICINTENABLE and VICINTSELECT, *Section 8.6.7: VICINTENABLE register* and *Section 8.6.6: VICINTSELECT register* respectively), at the output of the interrupt request logic block (*Section 8.3.1*). The VICFIQSTATUS bit assignments are given in *Table 30*.

Table 30. VICFIQSTATUS register bit assignments

Bit	Name	Reset value	Description
[31:00]	FIQStatus	32'h0	Each bit is associated to an interrupt. If a bit is set, it indicates that the relevant interrupt is active, and generates an interrupt to the processor.

8.6.5 VICRAWINTR register

The VICRAWINTR is a RO register, which provides the raw status of both interrupt sources and software interrupts (before masking through enable registers, VICINTENABLE and VICINTSELECT). The VICRAWINTR bit assignments are given in *Table 31*.

Table 31. VICRAWINTR register bit assignments

Bit	Name	Reset value	Description
[31:00]	Raw Interrupt	-	Each bit is associated to an interrupt. If a bit is set, it indicates that the relevant interrupt request is active before masking.

8.6.6 VICINTSELECT register

The VICINTSELECT is a RW register which allows to select whether the corresponding interrupt generates an FIQ or an IRQ interrupt. The VICINTSELECT bit assignments are given in *Table 32*.

Table 32. VICINTSELECT register bit assignments

Bit	Name	Reset value	Description
[31:00]	IntSelect	32'h0	Each bit is associated to an interrupt line. Each bit allows to select the type of interrupt for relevant interrupt requests, according to encoding: 1'b0 = IRQ interrupt 1'b1 = FIQ interrupt

8.6.7 VICINTENABLE register

The VICINTENABLE is a RW register which allows to enable the interrupt request lines by masking the interrupt sources for the IRQ interrupt. The VICINTENABLE bit assignments are given in *Table 33*.

Table 33. VICINTENABLE register bit assignments

Bit	Name	Reset value	Description
[31:00]	IntEnable	32'h0	Each bit is associated to an interrupt line. If a bit is set, the relevant interrupt request to the processor is enabled. A HIGH bit sets the corresponding bit in the VICINTENABLE Register. A LOW bit has no effect.

8.6.8 VICINTENCLEAR register

The VICINTENCLEAR is a WO register which allows to clear bits in the VICINTENABLE register (*Section 8.6.7: VICINTENABLE register*). The VICINTENCLEAR bit assignments are given in *Table 34*.

Table 34. VICINTENCLEAR register bit assignments

Bit	Name	Reset value	Description
[31:00]	IntEnableClear	-	Each bit is associated to an interrupt line. Writing a 1'b1 in a bit, the corresponding bit in the VICINTENABLE register is cleared. Writing a 1'b0 has no effect.

8.6.9 VICSOFTINT register

The VICSOFTINT (software interrupt) is a RW register which generates software interrupts. The VICSOFTINT bit assignments are given in *Table 35*.

Table 35. VICSOFTINT register bit assignments

Bit	Name	Reset value	Description
[31:00]	SoftInt	32'h0	Each bit is associated to a source interrupt. Setting a bit, a software interrupt for the specific source interrupt is generated before interrupt masking.

8.6.10 VICSOFTINTCLEAR register

The VICSOFTINTCLEAR is a WO register which allows to clear bits in the VICSOFTINT register (*Section 8.6.9: VICSOFTINT register*). The VICSOFTINTCLEAR bit assignments are given in *Table 36*.

Table 36. VICSOFTINTCLEAR register bit assignments

Bit	Name	Reset value	Description
[31:00]	SoftIntClear	-	Each bit is associated to an interrupt line. Writing a 1'b1 in a bit, the corresponding bit in the VICSOFTINT register is cleared. Writing a 1'b0 has no effect.

8.6.11 VICPROTECTION register

The VICPROTECTION is a RW register which allows to enable or disable protected register access. The VICPROTECTION bit assignments are given in *Table 37*.

Table 37. VICPROTECTION register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read: undefined. Write: should be zero.
[00]	Protection	1'h0	Enable/disable protected register access. Setting this bit, protected register access is enabled ensuring that only privileged mode accesses, reads and writes, can access the interrupt controller registers. Clearing this bit, protected register access is disabled allowing both user mode and privileged mode to access the registers.

Note: This register is cleared on reset, and it can only be accessed in privileged mode. If the AHB master cannot generate accurate protection information, this register shall leaved in its reset state (protection disabled) in order to enable User mode access.

8.6.12 VICVECTADDR register

The VICVECTADDR (vector address) is a RW register which contains the ISR address of the currently active interrupt. The VICVECTADDR bit assignments are given in *Table 38*.

Table 38. VICVECTADDR register bit assignments

Bit	Name	Reset value	Description
[31:00]	Vector Addr	32'h0	Reading from this register provides the address of the currently active ISR, indicating that the interrupt is being serviced. Writing to this register indicates that the interrupt has been serviced and the interrupt is cleared.

Note: The ISR reads the VICVECTADDR register when an IRQ interrupt is generated. At the end of the ISR, the VICVECTADDR register is written to, to update the priority hardware. Reading or writing to this register at other times can cause incorrect operation.

8.6.13 VICDEFVECTADDR register

The VICDEFVECTADDR (default vector address) is a RW register which contains the default ISR address.

8.6.14 VICVECTADDR register

Each VICVECTADDRi (with i = 0...15) is a RW register which contains the ISR address for the relevant vectored interrupt.

8.6.15 VICVECTCNTL register

Each VICVECTCNTLi (with i = 0...15) is a RW registers which allows to select the interrupt source for the i-th vectored interrupt. The bit assignments of VICVECTCNTLi are given in *Table 39*.

Table 39. VICVECTCNTL registers bit assignments

Bit	Name	Reset value	Description
[31:06]	Reserved	-	Read: undefined. Write: should be zero.
[05]	E	1'h0	If set, it enables vector interrupt.
[04:00]	IntSource	5'h0	It allows to select any of the 32 interrupt sources (IRQ only).

Note: Vectored interrupts are only generated if the interrupt is enabled. The specific interrupt is enabled in the VICINTENABLE register (Section 8.6.7: VICINTENABLE register), and the interrupt is set to generate an IRQ interrupt in the VICINTSELECT register (Section 8.6.6: VICINTSELECT register). This prevents multiple interrupts being generated from a single request if the controller is incorrectly programmed.

8.6.16 Peripheral identification registers

The read-only VICPeriphID0-3 registers are four 8 bit registers, that span address locations 0xFE0-0xFEC. You can treat the registers conceptually as a single 32 bit register. The read-only registers provide the following options for the peripheral. *Table 40*.

Table 40. Peripheral identification registers bit assignments

Bit	Name	Description
[31:24]	Configuration	This is the configuration option of the peripheral. the configuration value is 0.
[23:20]	Revision number	This is the revision number of the peripheral. The revision number starts from 0.
[19:12]	Designer	This is the identification of the designer 12 -15 Designer 0; 16 -19 Designer 1
[11:00]	Part number	This identifies the peripheral. The VIC uses the three-digit product code 0x90. 0 - 7 Part number 0; 8-11 Part number 1

8.6.17 VICPERIPHID0 register

The read-only VICPERIPHID0 register, with address offset of 0xFE0, is hard-coded, and the fields within the register determine the reset value. *Table 41* shows the bit assignments for this register.

Table 41. VICPERIPHID0 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:00]	Partnumber0	These bits read back as 0x90

8.6.18 VICPERIPHID1 register

The read-only VICPERIPHID1 register, with address offset of 0xFE4, is hard-coded, and the fields within the register determine the reset value. *Table 42* shows the bit assignments for this register.

Table 42. VICPERIPHID1 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:04]	Designer0	These bits read back as 0x1
[03:00]	Partnumber1	These bits read back as 0x1

8.6.19 VICPERIPHID2 register

The read-only VICPERIPHID2 register, with address offset of 0xFE8, is hard-coded, and the fields within the register determine the reset value. *Table 43* shows the bit assignments for this register.

Table 43. VICPERIPHID2 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:04]	Revision	These bits read back as 0x1
[03:00]	Designer1	These bits read back as 0x0

8.6.20 VICPERIPHID3 register

The read-only VICPERIPHID3 register, with address offset of 0xFEC, is hard-coded, and the fields within the register determine the reset value. *Table 44* shows the bit assignments for this register.

Table 44. VICPERIPHID3 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:00]	Configuration	These bits read back as 0x0

8.6.21 Identification registers

The read-only VICPCELLID0-3 registers are four 8 bit registers, that span address locations 0xFF0-0xFFC. You can treat the registers conceptually as a single 32 bit register. Use the register as a standard cross-peripheral identification system.

8.6.22 VICPCELLID0 register

The read-only VICPCELLID0 register, with address offset 0xFF0, is hard-coded and the fields within the register determine the reset value. *Table 45* shows the bit assignments for this register

Table 45. VICPCELLID0 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:00]	VICPCellID0	These bits read back as 0x0D

8.6.23 VICPCELLID1 register

The read-only VICPCELLID1 register, with address offset 0xFF4, is hard-coded and the fields within the register determine the reset value. *Table 46* shows the bit assignments for this register

Table 46. VICPCELLID1 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:00]	VICPCellID1	These bits read back as 0xF0

8.6.24 VICPCELLID2 register

The read-only VICPCELLID2 register, with address offset 0xFF8, is hard-coded and the fields within the register determine the reset value. *Table 47* shows the bit assignments for this register

Table 47. VICPCELLID2 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:00]	VICPCellID2	These bits read back as 0x05

8.6.25 VICPCELLID3 register

The read-only VICPCELLID3 register, with address offset 0xFFC, is hard-coded and the fields within the register determine the reset value. *Table 48* shows the bit assignments for this register

Table 48. VICPCELLID3 register bit assignments

Bit	Name	Description
[31:08]		Read undefined
[07:00]	VICPCellID3	These bits read back as 0xB1

9 Bus interconnection matrix

The SoC interconnection matrix scheme is given *Table 49* and *Table 50*.

Table 49. SoC interconnection matrix scheme

			Ethernet MAC	C3	USB (Host and Device)	RAS_E	RAS L	Processor	DMA#1	DMA#2	RAS H
Targets	MemCtr#0							REQ			
	MemCtr#1						REQ				
	MemCtr#2	lcm5							REQ1		REQ2
	MemCtr#3	lcm7	REQ1		REQ2	REQ3				REQ4	
	MemCtr#4	lcm8		REQ1	REQ2						
	Ras_I				REQ						
	Ras_M							REQ			
	Ras_G1								REQ		
	Ras_G2									REQ	
	Ras_F	lcm6	REQ1	REQ2							
	Sbs_LowSpeed	lcm1						REQ 1	REQ2		REQ3
	Sbs_High Speed	lcm4						REQ 1			REQ2
	Sbs_Basic	lcm3						REQ 1			REQ2
	Sbs_Application	lcm2						REQ 1		REQ3	REQ2

Note: 1 RAS_E,F, G,G2,H,M designate the internal logic ports connecting to the RAS subsystem.

2 RAS_G1 and RAS_G2 are internal logic ports assigned to the interconnection implemented between the two masters of the DMA controller with the IPs in the RAS subsystem.

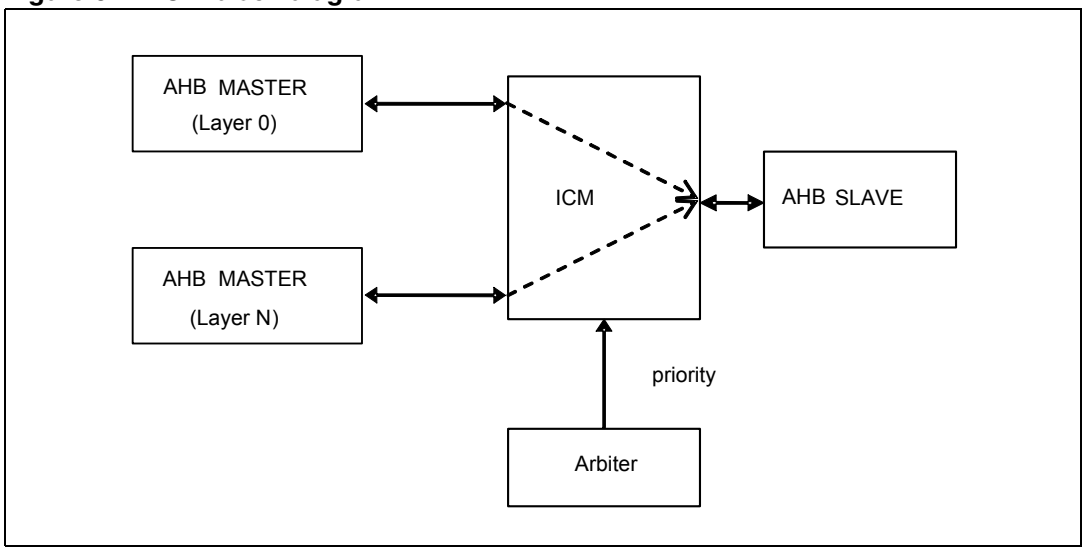
Table 50. SoC interconnection matrix

Legend	Description
A	Grey box: No connection exists between target and initiator
B	White box: A connection exists between target and initiator
C	‘Req’: A connection that is required between target and initiator

9.1 ICM

The AMBA system has ten programmable Multi-layer Interconnection Matrix (ICM). The ICM allows multiple master layers to access a slave (See *Figure 6*)

Figure 6. ICM block diagram



A layer is referred to as one or more masters that complete together with one master winning ownership of the slave.

When there is more that one layer looking for access to the slave at the sametime, this is referred to as a clash of requests. Whenever a clash is detected, only one layer can gain access to the slave. The layer that do not gain access to the slave need to have their address and control signals stored into their input stage. When address and control signals are stored into an input stage, then the stored transfer controls the request and lock generation circuitry. When a lower priority layer is in the middle of a burst transfer and a higher priority layer issues a transfer, the higher priority layer is stored and then held off until the lower priority layer completes the transfer.

Table 51 shows the Master Layer on each ICM input stages while *Table 52* lists the ICM slaves.

Table 51. ICM master layers (Initiator)

ICM	L0	L1	L2	L3
1	Processor	RAS_H	DMA#1	
2	Processor	RAS_H	DMA#1	
3	Processor	RAS_H		
4	Processor	RAS_H		
5	RAS_H	DMA#1		
6	Ethernet MAC	USB(Hosts - Device		
7	Ras_E	DMA#2	Ethernet MAC	C3
8	C3	USB(Hosts - Device)		

Table 52. ICM slaves (Targets)

ICM	M1
1	Sbs_LowSpeed
2	Sbs_Application
3	Sbs_Basic
4	Sbs_Highspeed
5	MemCtr#2 (MPMC)
6	Ras_F
7	MemCtr#3 (MPMC)
8	MemCtr#4 (MPMC)

In the Miscellaneous register bank are allocated eight register (ICM_x_ARB_CFG) one for each ICM.

These registers have all the same layout:

- The 31th bit is in charge of choosing the arbitration scheme: fixed priority or round robin
- Bit [30:28] specifies the priority starting level in case of round robin arbitration protocol
- Then 3 bit are allocated to each layer to set the priority level in case of fixed priority scheme: bit [2:0] for Layer0, [5:3] for Layer1 and so on. Refer to table 4 for layer list.

10 DDR memory controller (MPMC)

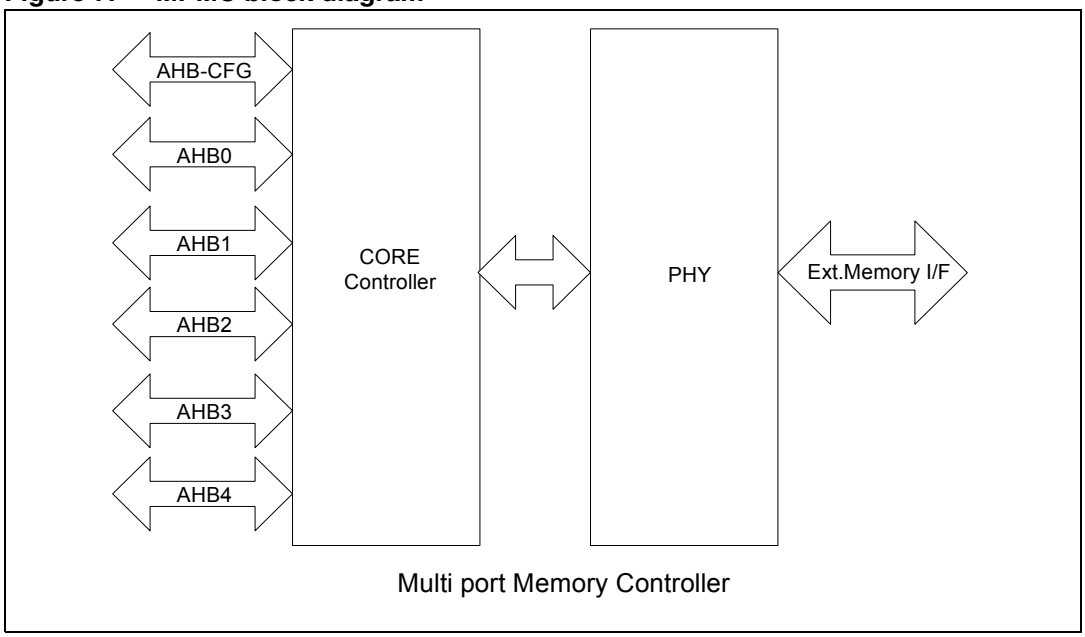
10.1 Overview

SPEAr300 integrates a high performances multi-port memory controller able to support DDR-Mobile and DDR2 double data rate memory devices. The multi-port architecture ensures memory is shared efficiently among different high-bandwidth client modules.

It offers 6 internal ports. One of them is reserved to registers access during the controller initialization while the other five are used to access the external memory.

It also include the physical layer (PHY) and some DLL that allows a fine tuning of all the timing parameter to maximize the data valid windows at every frequency in the allowed range.

Figure 7. MPMC block diagram



10.2 Signal description

The following *Table 53* and *Table 54* describe the signals either for the internal connections that the external ones.

Table 53. External memory Interface signals

Signal name	Direction	Description
DDR_CLK_P	Out	Differential memory clock. Positive line
DDR_CLK_N	Out	Differential memory clock. Negative line.
DDR_DQS_(1:0)	Bidir	Differential memory data strobe positive line. Drove during write transaction and received from memory device during read transfer.

Table 53. External memory Interface signals (continued)

Signal name	Direction	Description
DDR_nDQS_(1:0)	Bidir.	Differential memory data strobe negative line. Drove during write transaction and received from memory device during read transfer.
DDR_CLKEN	Out	Memory clock enable (active high)
DDR_CS_(1:0)	Out	Memory chip select (active low)
DDR_RAS	Out	Memory row address select
DDR_CAS	Out	Memory column address select
DDR_WE	Out	Memory write enable
DDR_ADD_(14:0)	Out	Memory address bus
DDR_BA_(2:0)	Out	Memory bank address
DDR_DM_(1:0)	Out	Memory data mask (active high)
DDR_DATA_(15:0)	Bidir.	Memory data bus
DDR_ODT_(1:0)	Out	Memory On-die termination enable signals (active high)
DDR_GATE_(1:0)	Bidir.	Memory gate open (DQS delay tune considering board propagation delay and internal pad propagation delay)

Table 54. Internal signals

Signal name	Description
AHB-CFG	Through this bus, connected to the multilayer interconnection matrix output port #3, the CPU or any other logic block with master capability can configure the memory controller registers.
AHB0	Through this bus the CPU can access the external memory.
AHB1	Through this bus the master port L can access the external memory.
AHB2	This bus, through the multilayer interconnection matrix output port 5, give access to the external memory to the following masters: DMA1 Master port H
AHB3	This bus, thanks to an external multiplexer, give access to the external memory to the following masters: DMA2 Ethernet controller Master port E Channel controller coprocessor (C3)
AHB4	This bus, thanks to an external multiplexer, give access to the external memory to the following masters: USB2 host and device controllers Channel controller coprocessor (C3)
IRQ	The memory controller interrupt request line is connected to the physical request number 13

10.3 Features overview

The memory controller includes the following main features:

- Multi channel AHB interfaces:
  - Five independent AHB ports.
  - Separate AHB memory controller programming interface.
  - Support all AHB burst types.
  - Lock transaction are not supported.
  - Port queue post multiple AHB transactions.
- Internal efficient port arbitration scheme to ensure high memory bandwidth utilization.
- Fully pipelined read - write commands.
- Advanced bank look-ahead features for high memory throughput.
- Programmable register interface to control memory device parameters and protocols including the following main functionalities:
  - Auto pre-charge.
  - Read/write grouping.
  - Bank splitting.
  - Bank grouping.
  - Swapping.
  - Aging.
- Full initialization of memory on memory controller reset.
- DRAM controller supports both DDR-Mobile and DDR2 memory devices:
  - DDR- Mobile up to 166 MHz (333 MT/sec).
  - DDR2 up to 333 MHz (666 MT/sec).
- Memory frequency with DLL enable configured within range from 100 MHz to 333 MHz.
- Controller supports:
  - Wide range of memory device: 128 MBit, 256 MBit, 512 MBit, 1 MBit, 2 MBit.
  - Two chip selects.
  - Memory data with 8 or 16 bit.
  - Configurable memory parameters:
  - Row address from 8 to 15 bit.
  - Column address from 7 to 14 bit.
  - Memory internal banks 4 or 8.
- Programmable DQS0/1 signals per byte configurable single ended and differential mode.
- Built-in adjustable delay compensation circuitry (DCC) for reliable data sends and captures timing.
- Dynamic memory self refresh power reduction automatically activated from SoC power management unit.

10.4 Main block description

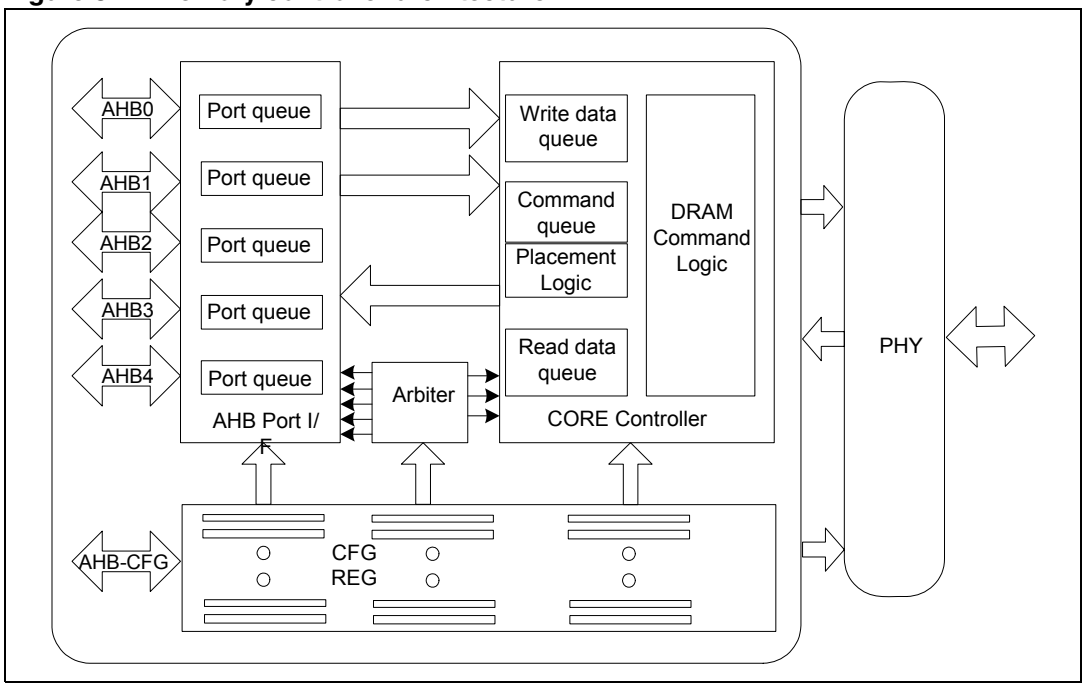
The multi-port memory controller supports high memory bandwidth utilization and an efficient arbitration scheme for high priority agent requests.

The memory controller architecture consists of the following main sub-blocks:

- AHB port interfaces.
- Arbiter.
- Command queue with placement logic.
- Write data queue.
- DRAM command processing.

The core controller interfaces with standard AHB ports through the interface blocks. All requests are processed through the internal arbiter which feeds single commands to the command queue of core controller. Write and read data is routed independently of the arbiter through the data interfaces. There are multiple write data interfaces to the write data queue of core controller, and a single read data interface back from the core controller to the port interface blocks. The architecture of the multi-port system is shown in *Figure 8*.

Figure 8. Memory controller architecture



The interface blocks contain FIFOs for commands, READ and WRITE data, handling any clock domain crossings as required. From the port interface blocks, commands are processed by an Arbiter which feeds single commands to the command queue of the Memory Controller core. WRITE and READ data are routed directly to the WRITE and READ data queues of the Memory Controller core, Arbiter independently.

Each port has a distinct WRITE data interface to the WRITE data queue of the Memory Controller core. However, for READ data, all ports share a single READ data interface back to the port interface blocks.

10.4.1 AHB-Memory controller interfaces

The Memory Controller core interfaces with 5 AHB data ports and 1 AHB register port. The AHB data ports function as AHB slaves to external AHB masters such as CPUs, DMAs, DSPs, and other peripherals. The port implementation is restricted to support the AHB-lite protocol and is designed for Multi-Layer AHB architectures. This implies that an AHB slave port will never respond with a SPLIT or a RETRY response type. Early termination of AHB bursts is fully supported.

Note: An AHB slave port can be used in a system with masters that support the full AHB protocol as long as the system is Multi-Layer AHB.

AHB-Memory Controller interfaces handle all communication between the AHB bus and the Memory Controller core. An incoming AHB transaction is first synchronized from the AHB clock domain to the Memory Controller core clock domain, then mapped into a Memory Controller core-level transaction, and finally stored in the AHB port FIFOs. From the AHB FIFOs, the transaction is presented to the Arbiter which arbitrates requests from all ports and forwards a single transaction to the Memory Controller core.

Configured options

Each AHB port in the Memory Controller has been defined for the requirements of the intended system. The configured options are:

Type of interface to the memory controller core clock

All ports of this Memory Controller are clock domain programmable relative to the main Memory Controller core clock. These ports initialize in asynchronous operation, but can be changed by programming the associated ahbY_fifo_type_reg parameter. For more information on the setting of this parameter, refer to Port clocking.

Asynchronous FIFOs handle the clock domain crossing when operating in any of the non synchronous modes. Refer to *Table 55: Configured AHB settings*, for the clock relativity for each port.

Note: When switching between asynchronous and synchronous mode of operation, ensure that there are no outstanding transactions on the port whose behaviour is being changed. If transactions are waiting, there may be unexpected loss of data or a lockout condition on that AHB port.

Datapath width

Each port has a data interface width of 32 bits.

READ and WRITE command Lengths for INCR Operations

AHB ports handle sequential requests of unspecified length (INCR) by issuing block data requests of the size programmed in the associated ahbX_wrcnt or ahbX_rdcnt parameter (where X is the port number). If the request is larger than the value programmed in that parameter, the request will be divided into multiple requests. Subsequent read commands will be issued when the last word of data has been delivered back to the requesting AHB port. Subsequent WRITE commands will be issued after the last data word of the previous request has been transferred from the AHB interface to the Memory Controller core. The value defined in each parameter should be a multiple of the number of bytes in the AHB bus width. For this Memory Controller, since the AHB bus width is 32 bits, these parameters should be programmed to 4, 8, 12, 16, etc. up to 1024 bytes.

Port FIFO depths

Each data port contains a read, a write and a command FIFO. The depth of each buffer in each port is listed in *Table 55: Configured AHB settings*.

Error detection

When an illegal operational condition is detected on a new AHB transaction entering the port, the port responds with an ERROR.

Port clocking

There are four user-selectable modes of operation for each of the AHB-Memory Controller port interfaces.

The mode is set by programming the corresponding ahbY_fifo_type_reg parameter. The four settings are:

Synchronous ('b11)

The AHB port clock and the Memory Controller core clock must be aligned in frequency in phase. The AHB-Memory Controller port interface block will not be required to perform any clock synchronization in any of the FIFOs.

- **Port: Core Pseudo-Synchronous ('b10)**

  The port operates at half of the frequency of the Memory Controller core frequency, with clocks that are aligned in phase. One stage of the two-stage synchronization logic of the FIFOs will be utilized to synchronize commands, WRITE data and READ data to the appropriate clock domain.

- **Port: Core Pseudo-Synchronous ('b01**)

  The port frequency is twice the Memory Controller core frequency, although the clocks are aligned in phase. One stage of the two-stage synchronization logic of the FIFOs will be utilized to synchronize commands, WRITE data and READ data to the appropriate clock domain.

Asynchronous ('b00)

The AHB bus and the Memory Controller core operate on clocks that are mismatched in frequency and phase. The AHB port FIFOs use two stages of synchronization logic to synchronize commands, WRITE data and READ data to the appropriate clock domain.

AHB Port FIFOs

Incoming transactions on the AHB bus are processed by the interface logic and mapped into equivalent transactions on the Memory Controller core bus. These transactions are queued in the port FIFOs.

There are three separate AHB port FIFOs for commands, READ data and WRITE data. The depths of the FIFOs are configured by the user and generally dependent on system requirements.

Command FIFO

The command FIFO holds the AHB command address, burst type and size. Typically, the command FIFO is fairly small in depth due to the single-threaded, pipelined nature of the AHB protocol. The protocol does not allow more than one outstanding transaction on any

AHB port. A deeper command FIFO is only useful in situations when the master issues several very short WRITE bursts.

In this case, the commands and the associated data will be completely captured in the command and write FIFOs and the bus will be free to start other operations.

Read FIFO

The read FIFO holds the ahbX_HRDATA signals sent back from the Memory Controller. There is only one streaming READ data interface out from the memory for all AHB ports. The Memory Controller steers this data stream to the port that requested the data. As a result, the AHB ports must be ready to accept the READ data as soon as it is available on the internal Memory Controller core bus to avoid stalling the Memory Controller itself.

Write FIFO

The write FIFO holds the ahbX_HWDATA signals sent into the Memory Controller. The depth of the write FIFO depends on the typical length of a burst write transaction.

Note: There is a WRITE data queue inside the Memory Controller core as well. Therefore, the write FIFOs allow the AHB bus to off load its WRITE command completely before it is fully transferred to the Memory Controller core buffers.

Settings

Table 55. Configured AHB settings

Port number	Data width	Clock domain type	Command FIFO depth	Write FIFO depth	Read FIFO depth
0	32	Async or Sync	8	8	8
1	32	Async or Sync	8	8	8
2	32	Async or Sync	8	16	16
3	32	Async or Sync	8	8	8
4	32	Async or Sync	8	8	8

Reset

There are two sets of reset logic inside the Memory Controller: the reset for the Memory Controller core and the reset for the AHB ports.

The reset signal for the Memory Controller core is the asynchronous active-low reset rst_n signal that resets all critical flip-flops in the system to ensure the Memory Controller core exits from reset in a known state.

When the Memory Controller core is reset all parameters are reset too, so any command inside the Memory Controller core are lost. Resetting Memory Controller core does not automatically reset the AHB ports: resetting the Memory Controller core without AHB port reset will generate unknown behavior.

The AHB port reset is the active-low asynchronous signal ahbX_HRESETn. When this reset is asserted, the associated AHB port will be reset. The port FIFOs will be cleared and the pointers will be reset. To prevent corruption within the Memory Controller, AHB ports should only be reset at initialization and while the port is idle at the interface and with no commands within the Memory Controller core.

During initialization, it is recommended that both resets be asserted simultaneously. The Memory Controller core reset should be removed first, followed by the port reset. The reset should be asserted for at least 5 cycles.

AHB register port

The AHB-Memory Controller Interface contains a special register port for converting AHB register addresses to Memory Controller core register addresses. This port operates asynchronously. However, unlike the data ports, the register port has no allocated storage in the form of FIFOs.

The register port only supports the AHB SINGLE transaction burst types for all transactions with a bytes-per-beat (2ahbX_HSIZE) equal to or less than the width of the AHB register bus. There is no support for WRAP transactions or early burst termination for the register port. BUSY cycles can be inserted between the incrementing burst operations.

Similar to the data ports, the AHB register port responds to illegal conditions with the ERROR response. However, a register port ERROR will only occur when the transaction address is not aligned to the size of the transaction.

All parameters related to the AHB port operation are located in the Memory Controller core register map.

These parameters are programmed during the Memory Controller initialization sequence along with all of the other device parameters. A typical boot-up sequence includes a reset of the AHB ports as well as of the Memory Controller core, followed by Memory Controller core setting for AHB operation by AHB register port.

AHB transactions

The AHB-Memory Controller Interfaces support the complete set of AHB transactions. The list includes: SINGLE, INCR4, INCR8, INCR16, WRAP4, WRAP8, WRAP16 and INCR.

For documentation purposes, INCRx will refer to INCR4, INCR8 or INCR16 commands. WRAPx will refer to WRAP4, WRAP8 or WRAP16 commands. Command handling is defined for full-size transfers, in which the bytes-per-beat is equal to the bus width, or narrow transfers, where the bytes-per-beat is less than the bus width.

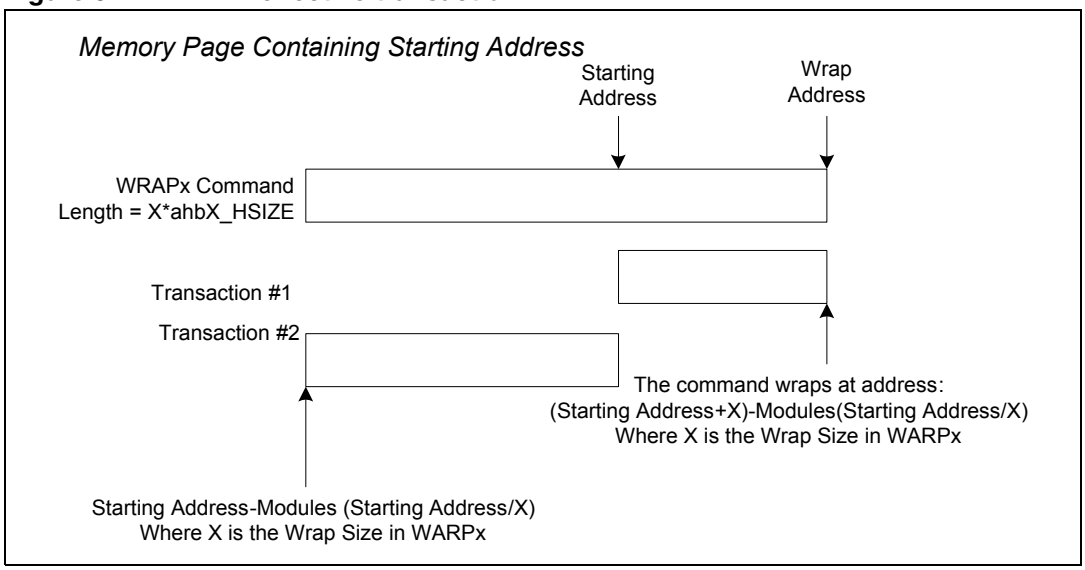
Full-Size SINGLE, INCRx or WRAPx transactions

SINGLE and INCRx transactions with a bytes-per-beat value (2ahbX_HSIZE) equal to the width of the AHB bus are issued as single Memory Controller commands to the Memory Controller core. The starting address of the Memory Controller transaction is the same as the address specified during the first NONSEQ beat of the corresponding AHB transaction. The Memory Controller core accepts these commands in a pipelined fashion such that, for read transactions, the READ data can be delivered in contiguous data words back to the AHB interface. These words may not be contiguous if other AHB requests with higher priority get placed ahead of some of these commands, or if the read queue inside the Memory Controller core is not deep enough to absorb a full burst.

WRAPx transactions with a starting address that is not aligned to the total number of bytes in the burst (bytes-per-beat times the number of beats) are issued as two separate Memory Controller core transactions as shown in *Figure 9*. The first transaction has a starting address which is the same as the WRAP transaction starting address and its byte count is the number of bytes from the starting address to the wrap boundary. The second transaction has a starting address which is aligned to the byte count in the WRAPx transfer and its byte count is the number of bytes from the wrapped address back to the starting address. This

means that an AHB WRAP instruction is divided into two Memory Controller core transactions, one for the non-wrapped portion of the transaction and one for the wrapped portion. The starting address determines whether the WRAP transaction will actually wrap or not. A wrap transaction with an aligned address is equivalent to an INCRx command and is treated as such by the AHB port.

Figure 9. WRAPx effective transaction



Full-Size INCR transactions

Since the Memory Controller core bus protocol does not support transactions of an unspecified length, AHB INCR transactions require special handling by the AHB port logic. Each AHB port contains a pair of programmable parameters for determining the length of a command that will be issued to the Memory Controller core when an INCR command is issued to the AHB port. These parameters are ahbX_wrcnt and ahbX_rdcnt and they hold the programmed size used for an unspecified length WRITE and a READ command length in bytes for port X when this type of command is issued to the Memory Controller core.

The value defined in these parameters should be a multiple of the number of bytes in a data word for the Memory Controller core. If that is not the case, the parameter values will automatically be truncated to the data word boundary. Clearing these parameters will cause the port to issue commands of 0 length to the Memory Controller core, which the core interprets as the pre-configured value of 1024.

The values for the ahbX_wrcnt and ahbX_rdcnt parameters should be chosen carefully and should be based on the average length of an INCR transaction expected from the AHB master. If the values are programmed too low, the AHB port must issue a large number of small Memory Controller core transactions that may fill up the command queue and reduce system performance. A full queue may also inhibit higher priority requests from meeting their latency requirements. On the other hand, if the values for the parameters are programmed too high, then write transactions may force the AHB port to issue masked WRITE commands (WRITE commands that do not alter the contents of memory) to the Memory Controller core to complete the transactions. Similarly, for READ transactions, programming the length too large will cause extraneous data to be gathered, wasting cycles. An optimal value for the READ and WRITE length is important.

Note: The values in the ahbX_wrcnt and ahbX_rdcnt parameters should never exceed 1024.

The first command is issued when the AHB request is decoded. Subsequent read commands are issued when the last word of data has been delivered from the Memory Controller core to the AHB port.

Subsequent WRITE commands are issued after the last WRITE data word of the last command has been transferred from the AHB bus to the Memory Controller core.

Narrow SINGLE or INCRx transactions

For INCRx transactions with a bytes-per-beat less than the width of the AHB data bus, the port gathers data from the AHB master until it collects all the data needed to form a complete data word, or until the completion of the transaction.

The data is then issued to the Memory Controller core as multiples of the complete data word. If the start or end of a WRITE transaction is not aligned to the data word boundary, the appropriate bytes are masked off when the data is sent to the Memory Controller core. This provides a significant optimization in the port, preventing the Memory Controller core's Write Data Queue from filling with a large number of small data chunks. Performance is also improved significantly by allowing the Memory Controller core to write a complete word of data in a single burst instead of small sections of the word in several bursts. The reverse operation is done for INCRx READ transactions. The AHB port issues a READ of the complete word and then divides the READ data into smaller components depending on the size specified with the READ request. The alignment of data bytes within the AHB bus for transaction with a size less than the bus data width depends on the address of the written or read bytes. The AHB master is responsible for aligning the WRITE data bytes in the appropriate byte lanes of the AHB bus for these transactions. Similarly, the AHB master should mask out any data that is not in the appropriate byte lanes for the READ transaction as don't-care bytes.

Narrow WRAPx or INCR transactions

This optimization is NOT performed for WRAPx transactions with a bytes-per-beat less than width or INCR transactions of unspecified lengths. Each beat of these kinds of transactions is issued to the Memory Controller core as a separate transaction, thereby occupying a separate queue entry. Therefore, these transactions are inefficient and should only be used if necessary.

Data Alignment for Narrow transactions

Examples of the alignment of READ/WRITE data for a 64 bit wide AHB bus with BYTE, HALFWORD and WORD transactions at different addresses is shown in *Table 56* and *Table 57*.

Table 56. READ/WRITE data alignment - Little Endian

Transaction type	Address	Data alignment - little endian
BYTE	0x0	0x----------Aa
BYTE	0x1	0x----------Aa--
BYTE	0x2	0x----------Aa----
BYTE	0x3	0x--------Aa------
BYTE	0x4	0x------Aa--------
BYTE	0x5	0x----Aa----------

Table 56. READ/WRITE data alignment - Little Endian (continued)

Transaction type	Address	Data alignment - little endian
BYTE	0x6	0x--Aa----------
BYTE	0x7	0xAa----------
HALF WORD	0x0	0x----------BbAa
HALF WORD	0x2	0x--------BbAa----
HALF WORD	0x4	0x----BbAa--------
HALF WORD	0x6	0xBbAa----------
WORD	0x0	0x--------DdCcBbAa
WORD	0x4	0xDdCcBbAa--------

Table 57. READ/WRITE data alignment - Big Endian

Transaction type	Address	Data alignment - big endian
BYTE	0x0	0xAa----------
BYTE	0x1	0x--Aa----------
BYTE	0x2	0x----Aa----------
BYTE	0x3	0x------Aa--------
BYTE	0x4	0x--------Aa------
BYTE	0x5	0x----------Aa----
BYTE	0x6	0x----------Aa--
BYTE	0x7	0x----------Aa
HALF WORD	0x0	0xAaBb----------
HALF WORD	0x2	0x----AaBb--------
HALF WORD	0x4	0x--------AaBb----
HALF WORD	0x6	0x----------AaBb
WORD	0x0	0xAaBbCcDd--------
WORD	0x4	0x--------AaBbCcDd

AHB-memory controller transaction mapping equations

For AHB ports, a WORD is defined as 32 bits (4 bytes) and is the maximum size supported for a 32 bit AHB port. *Table 58* shows examples of the mapping of various AHB transaction bursts and sizes to the corresponding Memory Controller core transaction lengths for a 32 bit wide AHB port interface.

Table 58. AHB-Memory controller translation example

AHBx transaction			Memory controller transaction	
Address	**Burst size AHBx HBURST**	**Bytes per beat (AHBx HSIZE)**	**Address**	**Length**
AHBx Address	SINGLE	Byte, half-word or word	AHBx Address	1 x (AHBx HSIZE)
AHBx Address	INCR4	Byte, half-word or word	AHBx Address	2 x (AHBx HSIZE)
AHBx Address	INCR8	Byte, half-word or word	AHBx Address	8 x (AHBx HSIZE)
AHBx Address	INCR16	Byte, half-word or word	AHBx Address	16 x (AHBx HSIZE)
AHBx Address	WRAP4 (offset n= 0-3)	Byte, half-word or word	Transaction#1: AHBx Address Transaction #2: Wrapped AHBx Address	Transaction #1: (4 - n) x (AHBx HSIZE) Transaction #2: n x (AHBx HSIZE)
AHBx Address	WRAP8 (offset n= 0-7)	Byte, half-word or word	Transaction#1: AHBx Address Transaction #2: Wrapped AHBx Address	Transaction #1: (8 - n) x (AHBx HSIZE) Transaction #2: n x (AHBx HSIZE)
AHBx Address	WRAP16 (offset n= 0-15)	Byte, half-word or word	Transaction#1: AHBx Address Transaction #2: Wrapped AHBx Address	Transaction #1: (16 - n) x (AHBx HSIZE) Transaction #2: n x (AHBx HSIZE)
AHBx Address	INCR (for each beat)	Byte or half-word	AHBx Address	1 x (AHBx HSIZE)
AHBx Address	INCR	Word	Transaction 1: AHBx HADDR Transaction 2: AHBx HADDR+blk size Transaction 3: 2 x (AHBx HADDR+blk size)(1)	(AHBx read_cnt) or (AHBx write_cnt)(2)

1. blk size is the value in ahbX_rdnct or abhX_wrcnt.
2. Programmable value

Early burst termination

Unlike the AHB bus protocol, the Memory Controller core bus protocol does not allow for early burst termination. As a result, the Memory Controller core requires the master to complete the whole READ/WRITE transaction of the length specified to the Memory

Controller core. The length is a defined quantity that is specified at the beginning of the transaction. For WRITE transactions, the AHB port forwards the data from the AHBx HWDATA signals provided by the AHB master to the Memory Controller core. If the WRITE transaction is early burst-terminated, the port will continue the data stream, but issue masked WRITE data for the duration of the transaction instead. This allows the whole transaction to be completed without corrupting data in memory.

For READ transactions, the AHB port returns the expected number of bytes of data to the AHB master. If a READ transaction is early burst-terminated, the Memory Controller continues to send the READ data from the Memory Controller core, but the data is not forwarded back to the AHB master.

Errors

- **Error types**

When an illegal operational condition is detected on a new AHB transaction entering the port (i.e. during a NONSEQ burst), the port responds with an ERROR. The ERROR is a two-cycle response as dictated by the AHB protocol. In the first cycle, the ahbX_HREADY signal is low and in the second cycle, the ahbX_HREADY signal is high. The illegal conditions which generate this error are:

1. Transactions with a bytes-per-beat greater than the width of the AHB bus.
2. The transaction address is not aligned to the size of the transaction. This is a requirement of the AHB protocol.

- **Error handling**

Once a port error is detected in the Memory Controller core, the following actions occur:

1. The internal interrupt signal controller_int is asserted.
2. A bit in the status parameter int_status will be set to 1'b1 to indicate the type of error.

The interrupt is cleared by writing to the interrupt acknowledge parameter int_ack. Setting 1'b1 a bit in the int_ack parameter will trigger the same bit in the int_status parameter to be cleared to 1'b0. Please refer to *Section 10.13* for more information on the interrupt parameters.

10.4.2 Arbiter

From the port interface blocks, commands are presented to the Arbiter, which is responsible for arbitrating between the port requests and sending a single command to the Memory Controller core.

10.4.3 Write data queue

The WRITE data queue is a WRITE data storage array for transactions. The queue consists of multiple buffers holding WRITE data for the write requests of a particular port. Write data is stored in these buffers for commands in the command queue until the command is processed in the placement logic and needed by the DRAM command arbitration logic. The buffers can accept data until space is no longer available. The buffers are defined to hold 16 entries.

The size of the WRITE data buffers and the burst length programmed into the memory devices affect the overall performance of a single port during the WRITE operation. Each buffer must have a depth of at least twice the number of data words for a memory burst to ensure that the Memory Controller can continuously burst WRITE data for a port. The buffer

should also be so large to hold enough words to consume data for a single write transaction related to the bus. The data may actually be written into the buffers from the bus at a later time, depending on the priority of the request and the number of transactions in the Command Queue.

10.4.4 DRAM command processing

The DRAM command processing logic is used to process the commands in the Command Queue.

The logic organizes the commands to the memory devices in such a way that data throughput is maximized. Bank opening and closing cycles are used for data transfers.

The logic uses a variety of factors to determine when to issue bank open and close commands. The logic reviews the entire Command Queue to look-ahead which banks are to be accessed in the future. The timing is then set to meet the trc and tras_min timing parameters of the memory devices, values which were programmed into the Memory Controller on initialization. This flexibility allows the Memory Controller to be tuned to extract the maximum performance out of memory devices. The parameters that relate to DRAM device protocol are listed in Chapter “Register Interface”.

10.4.5 Latency

By using the placement logic of the command queue in the Memory Controller core, a new request by any port can be immediately placed at the top of the command queue or can interrupt an ongoing request. This scheme allows a high priority request to be processed in the shortest possible time.

However, since there are many factors that determine the placement into the command queue, there are also many factors that affect the actual latency of the command. These factors include:

The coherency status of the transactions already in the command queue: If there is a data coherency conflict with a transaction already in the command queue, the new transaction will be placed after the transaction that produced the conflict. The position of the conflicting transaction determines the latency of the high priority READ or WRITE command.

- The priority status of the transactions already in the command queue: If the new command has a higher priority than those already in the command queue, the new request will be serviced ahead of the lower priority command. As a result, the latency of the new command will be lower than the latency of the older command.
- The READ, WRITE, and bank information of the transactions already in the command queue: In general, READs will be placed ahead of WRITEs when both are of the same priority level. READ commands are grouped with other READ commands of similar priorities and WRITE commands are grouped with other WRITE commands of similar priorities. Among these groupings, transactions with similar bank and different row destinations are separated as much as possible.

If every placement conditions are met, a new command would be placed at the top of the command queue. However, if the new command is of a higher priority than the transaction executing, the current command will be interrupted and the new command will be processed first. The interruption will occur at a natural burst boundary of the DRAMs. The interrupted transaction will be placed at the top of the placement queue and it will be recovered after the new request is completed. The page status of the new transaction determines when the current transaction is interrupted.

If the page for the new transaction is already open, the current transaction will be interrupted at the next natural burst boundary of the DRAM device. If the page is not currently open instead, the new request will be placed at the top of the command queue while its page is prepared.

There are a fixed number of latency cycles in the memory controller, based on the pipeline through the memory controller logic. These steps are:

- Command passing through the port interface. (fixed)
- Arbitration through the Arbiter. (fixed)
- Placement into the Command Queue. (fixed)
- Memory Command Generation. (variable)
- Sending of control signals from the core logic to flip-flops near the I/O drivers. (fixed)
- Flight time to the DRAM device. (variable)
- Flight time from the DRAM device. (variable)
- For READs, synchronization of READ data from the data strobe domain. (fixed)
- For READs, data pass through the port interface. (fixed)

For asynchronous AHB interfaces, additional 4-5 cycles are included for the round-trip transaction to synchronize to the Database core clock. Some typical scenarios for a Multi-Port AHB interface and their effects on latency are:

- Read latency with a page hit and empty queue.
- (9) + Cas Latency
- Read latency with a page miss to a closed page and an empty queue.
- Trcd + Cas latency + (9)
- Read latency with a page miss to an open page and an empty queue.
- Trp+Trcd+Cas latency+9
- Read latency with a page hit and a currently executing transaction.
- TBurst end+Cas Latency+9
- Read latency with a page miss to a closed bank and a currently executing transaction. The page open command will be executed while the current burst is completing if possible.
- MAX(Trcd, TBurst end)+ Cas Latency + 9
- Read latency with a page miss to an open page and an empty queue. The page close command will be executed while the current burst is completing if possible.
- MAX(Trp+Trcd,TBurst end)+Cas latency+9

TBurst end equals the time to complete the current burst in progress. The maximum value here is 3 for a READ command and 3 + twr for a WRITE command if the burst count is configured to 8 for DDR DRAM devices. The minimum value is 0.

10.5 Multi-port arbiter

The Arbiter manages arbitrating requests from the ports and sending requests to the Memory Controller core. Each transaction received by the Arbiter logic has an associated priority, which works with each port's arbitration logic to determine how ports issue requests to the Memory Controller core. The Memory Controller supports the Weighted Round-Robin arbitration scheme.

The Arbiter logic routes READ data from the Memory Controller core to the appropriate port. The requesting port is assumed to be able to receive the data. WRITE data from each port is connected directly to its own WRITE data interface inside the Memory Controller core, allowing the ports to independently pass them to the Memory Controller core buffers.

10.5.1 Arbitration overview

The weighted round-robin arbitration scheme is a three-step arbitration system. All commands are routed into priority groups based on the priority of the requests. Inside each priority group, requests are processed according to the “weight” (relative priority) of each port. Finally, each priority group sends a single command to the priority select module, which passes the highest priority command on to the Memory Controller core.

This arbitration scheme also supports two additional features. First, for situations where the priority (port and relative) for multiple commands are identical, a port ordering system whereby the user may adjust the order in which the ports are considered is provided. Beside, for situations where two ports may be linked, a mechanism allowing the pair of ports to share arbitration bandwidth for a better bandwidth efficiency, is also supplied.

Weighted round-robin arbitration is a complex arbitration scheme. To understand the operation, each concept must be first understood individually. This will be matter of the following Sections.

10.5.2 Understanding round-robin operation

Round-robin operation is the simplest form of arbitration and is the best one for systems not requiring requests to be treated preferentially to maintain bandwidth or minimize latency. This scheme uses a counter that rotates through the port numbers, incrementing every time a port request is granted.

If the port that the counter is checking has an active request and the Memory Controller core command queue is not full, the request will be sent to the Memory Controller core.

If there is not an active request for that port, the port itself will be skipped and the next port will be checked.

In any case, the counter is incremented one-by-one as any request has been processed, regardless of which port's request was arbitrated.

Round-robin arbitration guarantees that each port's requests can be successfully arbitrated into the Memory Controller core every N cycles, where N is the number of ports in the component itself.

No port will ever be locked out, and any port can have its requests processed on every cycle as long as all other ports are idle and the command queue is not full.

An example of the round-robin scheme is shown in *Table 59*.

Cycles 0, 2 and 6 show the system behaviour when the command queue is full. Cycle 8 shows the system behaviour when the port addressed by the arbitration counter does not have an active request.

All other cycles show normal behaviour.

Table 59. Round-Robin operation example

Cycle	Port addressed by an arbitration counter	Port requesting				Command queue full?	Winner of arbitration	Value of counter at the next cycle
		P0	P1	P2	P3			
0	0	Y	Y	Y	Y	Yes	None	0
1	0	Y	Y	Y	Y	No	P0	1
2	1		Y	Y	Y	Yes	None	1
3	1	Y	Y	Y	Y	No	P1	2
4	2	Y		Y	Y	No	P2	3
5	3	Y			Y	No	P3	0
6	0	Y		Y		Yes	None	0
7	0	Y		Y		No	P0	1
8	1			Y		No	P2	2
9	2			Y	Y	No	P2	3
10	3	Y			Y	No	P3	0

10.5.3 Understanding port priority

For AHB ports, the priority is associated with a port and each port has separate priority parameter for READ and WRITE operations. These values are stored into the programmable parameters ahbX_r_priority and ahbX_w_priority (where X is the port number) at startup. Internally, the ports are organized into priority groups based on their priority setting.

The priority value is also used by the placement logic inside the Memory Controller core when filling the command queue, with “0” meaning highest priority and “7” as lowest priority.

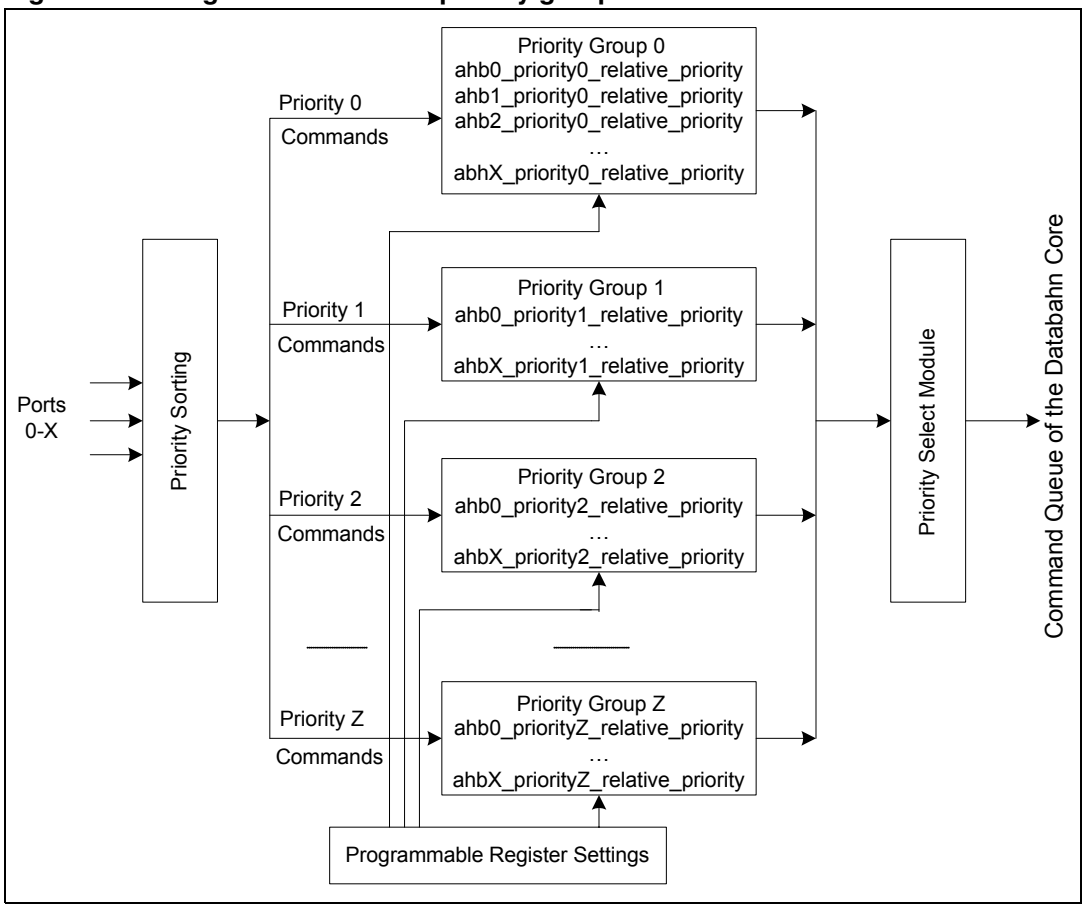
Even if the user is allowed to set a “0” priority level, it would be better to avoid this specific value so the placement queue can reach this level by aging.

10.5.4 Understanding relative priority

Inside each priority group, the relative priority is used to set arbitration. The Memory Controller contains 8 identical priority groups with control logic selecting among the requests from all ports at that given priority level. The relative priority parameters ahbX_priorityY_relative_priority (where X being the port number and Y the priority group) “weight” the ports for each level and determine how the priority group will be arbitrated.

Figure 10 shows this type of arbitration system.

Figure 10. Weighted round-robin priority group structure



Using relative priority concept, the arbitration is skewed in favor of certain ports following user setting.

Note: The relative priority parameters have a minimum acceptable value of 1 to prevent port lockout: A 0 value triggers an error condition.

If the relative priorities are all programmed to the same value within any priority group, the arbitration will mimic a version of simple round-robin scheme within that specific priority group. Instead of incrementing as any request is processed, the simple round-robin counter will only increment to the next port after the ahbX_priorityY_relative_priority number of requests are processed.

To each port X owning priority level Y, it will be allocated resources corresponding to the ratio of that port's relative priority (ahbX_priorityY_relative_priority) to the sum of all requesting port's relative priority values. If a particular port is not requesting, it will be not included in the sum calculation, i.e. the arbitration will be proportionately split among the requesting ports.

For instance, let us consider a system with 4 ports where all requests have priority 0. This system is described in *Table 60*.

Table 60. Relative priority example

Parameter	System A
ahb0_priority0_relative_priority	1
ahb1_priority0_relative_priority	2
ahb2_priority0_relative_priority	3
ahb3_priority0_relative_priority	4

For this system, port 0 will be processed 1/(1+2+3+4) = 1/10 of the time and Port 3 will be processed 4/ (1+2+3+4) = 4/10 of the time. However, if Port 2 is not actively requesting, then port 0 will be processed 1/(1+2+4) = 1/7 of the time and port 3 will be processed 4/(1+2+4) = 4/7 of the time.

In order to warrant that relative priorities are maintained, there is a weight counter for each port within each priority group. These counters track the number of transactions accepted for that port in that priority group. When any counter value reaches the programmed relative port priority, the scan order for that priority group will be internally modified. The port matching its relative priority will be dynamically positioned to the bottom of the scan order and its counter reset, allowing other ports to get a preferential position.

For ports that are not expected to issue requests at a given priority level, the associated relative priority parameter should be programmed to 0x1. This allows to minimum allocation avoiding the risk of lock out in case a command appears.

10.5.5 Understanding port ordering

With simple round-robin arbitration, the ports are scanned following their port number in incrementing order inside the system. Assuming that the command queue is not full, the port referenced by the counter is examined for valid incoming transactions. If there is an active request, it will be accepted. Otherwise, the next port in the scan order will be checked.

For Memory Controller performing weighted round-robin arbitration, the user can sort the order in which the ports are scanned. This is a useful feature either requests from some ports are more critical or specific order could reduce contention among ports.

The three bit ahbX_port_ordering parameters are used to set this new scan order. A value of 3'b000 sets the highest listing in the scan order as a value of 3'b111 is the lowest as well.

If the 5 ahbX_port_ordering parameters are set with unique values, the scan order will be modified to proceed sequentially following this new order.

If any of the port ordering parameters have the same value, those ports will still be equal in the arbitration test. In this case, the port number will select between these ports, with the lower-numbered port automatically being selected first.

For instance, let us consider a system with 8 ports and two port orders as shown in *Table 61*.

For System B, the port ordering parameters contain different values, so the resulting order is entirely based on the values of the parameters.

For System C, three ports have the same value set as port order. For these three ports, the port number settles the order. Remaining ports follow the port ordering parameters.

Table 61. Port ordering example

Parameter	System B	System C
ahb0_port_ordering	3	3
ahb1_port_ordering	4	0
ahb2_port_ordering	5	5
ahb3_port_ordering	6	6
ahb4_port_ordering	7	7
ahb5_port_ordering	0	1
ahb6_port_ordering	2	0
ahb7_port_ordering	1	0
Port Scan Order	P5-P7-P6-P0-P1-P2-P3-P4	P1-P6-P7-P5-P0-P2-P3-P4

If every port ordering parameters are set to the same value, the scan order will default to the numbered port order.

10.5.6 Weighted round-robin arbitration summary

The Memory Controller weighted round-robin arbitration system merges the concepts of round-robin operation, priority, relative priority and port ordering. The incoming commands are separated into priority groups based on the priority of the associated port for that type of command. Inside each priority group, the relative priority values are examined to settle the arbitration winner. If relative priority values are the same and no individual command can be selected, the scan order is used to select among the requests.

Finally the highest priority command incoming from the highest relative priority port, having the highest location in the scan order, will be selected and sent to the Memory Controller core.

For instance, let us consider the system described in *Table 62*. The counters refer to the ones inside each port priority group to guarantee that relative priorities are maintained. To simplify, on assumes the command queue never be full and commands are only received at priority level 0.

Table 62. System D specifications

Parameter	Port 0	Port 1	Port 2	Port 3
ahbX_priority0_relative_priority	4	3	2	1
ahbX_port_ordering	0	1	2	3

The behaviour is shown in *Table 63*. The highest requesting port in the scan order always wins arbitration and the scan order is dynamically modified whenever any port counter reaches its allocated relative priority value.

Note: If the command queue was considered, cycles where the command queue was full would not have any arbitration winner and therefore the counter values and scan order would not change on that cycle.

Table 63. System D operation

Cycle	Ports requesting				Arbitration winner	Next counter				Next scan order
	P0	P1	P2	P3		P0	P1	P2	P3	
										P0-P1-P2-P3
0	Y		Y	Y	P0	1	0	0	0	P0-P1-P2-P3
1	Y		Y	Y	P0	2	0	0	0	P0-P1-P2-P3
2	Y	Y	Y	Y	P0	3	0	0	0	P0-P1-P2-P3
3	Y	Y	Y	Y	P0	4	0	0	0	P1-P2-P3-P0
4	Y	Y	Y	Y	P1	0	1	0	0	P1-P2-P3-P0
5	Y	Y	Y	Y	P1	0	2	0	0	P1-P2-P3-P0
6	Y	Y	Y	Y	P1	0	3	0	0	P2-P3-P0-P1
7	Y		Y	Y	P2	0	0	1	0	P2-P3-P0-P1
8	Y		Y	Y	P2	0	0	2	0	P3-P0-P1-P2
9	Y			Y	P3	0	0	0	1	P0-P1-P2-P3
10	Y		Y	Y	P0	1	0	0	0	P0-P1-P2-P3
11			Y	Y	P2	1	0	1	0	P0-P1-P2-P3
12			Y	Y	P2	1	0	2	0	P0-P1-P3-P2

If the same system also contains two ports that only can request at priority level 1, the system behavior will be slightly altered. Adding these 2 ports leads to the second priority group structure increasing the arbitration depth. *Table 64* describes this system. The text in ***bold-italic (grey background)*** highlights the priority level changes for P4 and P5.

Table 64. System E specifications

Parameter	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
ahbX_priority0_relative_priority	4	3	2	1	1	1
ahbX_priority1_relative_priority	1	1	1	1	3	2
ahbX_port_ordering	0	1	2	3	4	5

To simplify, the command queue is again considered to never be full and it is assumed that commands from ports 0, 1, 2 and 3 are only received at priority level 0. The behavior is shown in *Table 65*.

Note: If any priority 0 port (P0, P1, P2, P3) is requesting, the system will behave as one only priority group is acting as shown in Table 63. Ports 4 and 5 can only win arbitration when no higher-priority commands exist.

Table 65. System E operation

Cycle	Port Requesting						Arbitration Winner	Next Counter						Next Scan Order
	P0	P1	P2	P3	P4	P5		P0	P1	P2	P3	P4	P5	
														Priority 0: P0-P1-P2-P3 Priority 1: P4-P5
0			Y			Y	P2	0	0	1	0	0	0	P0-P1-P2-P3-P4-P5
1	Y		Y			Y	P0	1	0	1	0	0	0	P0-P1-P2-P3-P4-P5
2			Y			Y	P2	1	0	2	0	0	0	P0-P1-P3-P2-P4-P5
3	Y		Y		Y	Y	P0	2	0	0	0	0	0	P0-P1-P3-P2-P4-P5
4			Y		Y	Y	P2	2	0	1	0	0	0	P0-P1-P3-P2-P4-P5
5					Y	Y	P4	2	0	1	0	1	0	P0-P1-P3-P2-P4-P5
6		Y			Y	Y	P1	2	1	1	0	1	0	P0-P1-P3-P2-P4-P5
7					Y	Y	P4	2	1	1	0	2	0	P0-P1-P3-P2-P4-P5
8					Y	Y	P4	2	1	1	0	3	0	P0-P1-P3-P2-P5-P4
9					Y	Y	P5	2	1	1	0	0	1	P0-P1-P3-P2-P5-P4
10					Y		P4	2	0	1	0	1	1	P0-P1-P3-P2-P5-P4

10.5.7 Priority relaxing

With reference to *Table 65*, it is evident that ports at lower priority levels will not win arbitration in weighted round-robin arbitration unless there are no higher priority requests. This could mean that, in a situation where high priority requests are being received continuously, lower priority requests could be locked out indefinitely. To avoid this scenario and control the arbitration latency for lower-priority ports, it is possible to disable priority groups temporarily. This is known as priority relaxing, and it is a time-controlled function.

Each higher priority group will be temporarily disabled when the pre-set counter value for the lower priority group has been reached and a request is waiting. The ahbX_priority_relax parameters set the counter value for port X at which the priority relax condition will be triggered.

The timing counters inside each port are controlled by the weighted_round_robin_latency_control parameter. When the latency control bit is set to 1'b1, the timing counters are free-running. Any timing counter may hit its ahbX_priority_relax value at any point. Whenever this happens, higher-priority groups are disabled to allow a waiting request for this port to be processed. This brings a random latency for each port, but the maximum latency is fixed at the ahbX_priority_relax value.

If the current port does not have any commands waiting when the timing counter hits the relax value, the counter will be reset and the Arbiter will works normally.

When the weighted_round_robin_latency_control parameter is cleared to 1'b0, the timing counters only count while that port has a waiting request that is not being processed. In this case, when the port's ahbX_priority_relax parameter value is reached, all priority groups at priority levels higher than the waiting request are disabled. This port's command is granted arbitration and is moved through to the Memory Controller core. Since the priority relax parameters and counters are associated with individual ports, it is possible that multiple

priority relax counters could reach their specified value simultaneously. In this case, the lower priority command will be arbitrated first and the higher priority command afterward. This situation could modify the arbitration latency slightly, causing it to be longer than the expected value in the priority relax parameter.

Consider the System F as described in *Table 66*. The same conditions apply as for the previous example. The command queue is considered to never be full, commands from ports 0, 1, 2 and 3 are only received at priority level 0, and commands from ports 4 and 5 are only received at priority 1.

Table 66. System F specifications

Parameter	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
ahbX_priority0_relative_priority	4	3	2	1	1	1
ahbX_priority1_relative_priority	1	1	1	1	2	1
ahbX_port_ordering	0	1	2	3	5	4

Table 67 shows the system behavior. The exact settings of the latency control and priority relax parameters are not displayed. Relaxed Ports column indicates instead which ports, if any, have hit their priority relax values. The following cycles are important to observe:

- **Cycles 1 and 7**: A port relaxes while a higher priority request and a higher scan order request are both present. The relaxed port still wins arbitration.
- **Cycle 4**: Two ports of the same priority relax. The higher scan order request wins arbitration.
- **Cycle 5**: Two ports of different priorities relax. The lower priority port that relaxed wins arbitration. The higher priority port that relaxed will maintain its relax condition, and win arbitration in the next cycle.

Table 67. System F operation with priority relaxing

Cycle	Ports Requesting						Relaxed Ports	Arbitration Winner	Next Counter						Next Scan Order
	P0	P1	P2	P3	P4	P5			P0	P1	P2	P3	P4	P5	
															Priority 0: P0-P1-P2-P3 Priority 1: P5-P4
0			Y		Y	Y		P2	0	0	1	0	0	0	P0-P1-P2-P3 P5-P4
1			Y		Y	Y	P5	P5	0	0	1	0	0	1	P0-P1-P2-P3 P5-P4
2			Y		Y	Y		P2	0	0	2	0	0	0	P0-P1-P3-P2 P4-P5
3		Y			Y	Y		P1	0	1	0	0	0	0	P0-P1-P3-P2 P4-P5
4	Y				Y	Y	P4, P5	P4	0	1	0	0	1	0	P0-P1-P3-P2 P4-P5

Table 67. System F operation with priority relaxing (continued)

Cycle	Ports Requesting						Relaxed Ports	Arbitration Winner	Next Counter						Next Scan Order
	P0	P1	P2	P3	P4	P5			P0	P1	P2	P3	P4	P5	
5	Y				Y	Y	P0, P5	P5	0	1	0	0	1	1	P0-P1-P3-P2 P4-P5
6	Y				Y		P0	P0	1	1	0	0	1	0	P0-P1-P3-P2 P4-P5
7	Y				Y	Y	P4	P4	1	1	0	0	2	0	P0-P1-P3-P2 P5-P4
8	Y	Y	Y			Y		P0	2	1	0	0	0	0	P0-P1-P3-P2 P5-P4
9		Y	Y	Y		Y		P1	2	2	0	0	0	0	P0-P1-P3-P2 P5-P4
10		Y	Y	Y		Y	P2	P2	2	2	1	0	0	0	P0-P1-P3-P2 P5-P4

Priority relaxing allows low priority commands to be able to move through the Arbiter to the Memory Controller core. This will ensure that the system can meet maximum latency requirements.

10.5.8 Port pairing

The Memory Controller Arbiter embeds a feature which allows adjacent ports to be grouped together and considered jointly for arbitration. The weighted_round_robin_weight_sharing parameter controls this function, with one bit per pair of ports in the Memory Controller. Bit 0 handles ports 0 and 1, Bit 1 handles ports 2 and 3 and so on. Where Memory Controller interfaces to an odd number of ports, the highest numbered port is excluded from the port pairing system.

Since the ports are grouped together, their relative priorities are not considered separately. Referring to *Section 10.5.4*, the general formula for port priority allocation is the ratio of that port's relative priority (ahbX_priorityY_relative_priority) to the sum of all requesting port's relative priority values. In this case, the relative priority value of only one of the paired ports is used for the sum calculation. This means that the bandwidth will be divided differently among the ports.

Let us consider the port pair at the top of the scan order: if one only port is requesting it will win arbitration. If the both are requesting, port ordering is used to determine which port wins arbitration.

When the ports are paired, their scan order can never be altered and they will always remain together in the scan order. Their counters increment together, so when they reach their relative priority value, the port pair will dynamically be placed at the bottom of the scan order for that priority group.

In order for port weight sharing to be used, the relative priority parameters for the port pair must be programmed to the same value and the port order of the paired ports should be sequential. If either condition is not followed, an error bit will be set to1'b1.

Let us consider System G as described in *Table 68*. Again, to simplify the command queue is considered to never be full, commands from ports 0, 1, 2 and 3 are only received at priority level 0 and commands from ports 4 and 5 are always at priority 1.

However, now ports P0 - P1, and ports P4 - P5 are paired.

Table 68. System G specifications

Parameter	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
ahbX_priority0_relative_priority	3	3	2	2	1	1
ahbX_priority0_relative_priority	3	3	2	2	1	1
ahbX_priority0_relative_priority	3	3	2	2	1	1
weighted_round_robin_weight_sharing	1 (Paired)		0 (Not Paired)		1 (Paired)	

Table 69 shows the system behavior with port pairing. Since ports P4 - P5 are still at a lower priority, they will be ignored unless none of the higher priority ports (i.e. P0, P1, P2 or P3) are requesting. Note the following points:

- When either port of a port pair wins arbitration, the counters for both ports of the pair increment.
- In Cycle 3, the port pair P0/P1 reaches its allocated relative priority.

Note: The port pair dynamically moves to the bottom of the scan order.

- In Cycle 8, the port pair P4/P5 reaches its allocated relative priority. However, since these are the only requests at priority 1, the scan order does not change.

Table 69. System G operation

Cycle	Ports Requesting						Arbitration Winner	Next Counter						Next Scan Order
	P0	P1	P2	P3	P4	P5		P0	P1	P2	P3	P4	P5	
														Priority 0: P0-P1-P2-P3 Priority 1: P4-P5
0	Y		Y				P0	1	1	0	0	0	0	P0-P1-P2-P3 P5-P4
1	Y		Y			Y	P0	2	2	0	0	0	0	P0-P1-P2-P3 P5-P4
2			Y			Y	P2	2	2	1	0	0	0	P0-P1-P2-P3 P5-P4
3	Y	Y		Y		Y	P0	3	3	1	0	0	0	P2-P3-P0-P1 P5-P4
4		Y		Y		Y	P3	0	0	1	1	0	0	P2-P3-P0-P1 P5-P4
5		Y		Y		Y	P3	0	0	1	2	0	0	P2-P0-P1-P3 P5-P4
6		Y				Y	P1	1	1	1	0	0	0	P2-P0-P1-P3 P5-P4

Table 69. System G operation (continued)

Cycle	Ports Requesting						Arbitration Winner	Next Counter						Next Scan Order
	P0	P1	P2	P3	P4	P5		P0	P1	P2	P3	P4	P5	
7						Y	P5	1	1	1	0	1	1	P2-P0-P1-P3 P5-P4
8					Y		P4	1	1	1	0	2	2	P2-P0-P1-P3 P5-P4
9					Y	Y	P5	1	1	1	0	1	1	P2-P0-P1-P3 P5-P4
10			Y		Y	Y	P2	1	1	2	0	1	1	P0-P1-P3-P2 P5-P4
11					Y	Y	P1	2	2	0	0	1	1	P0-P1-P3-P2 P5-P4

10.5.9 Error conditions

Because of the programming difficulties of the weighted round-robin arbitration scheme, an error reporting mechanism is included to notify users of illegal programming scenarios. These error conditions generate a Memory Controller core interrupt and set a bit in the wrr_param_value_err parameter to 1'b1. The potential error conditions are:

- Bit 0 = The 5 ahbX_port_ordering parameters do not contain unique values.
- Bit 1 = Any of the ahbX_priorityY_relative_priority parameters have been set to 10 value. A 0 value leads to unknown behavior. The minimum accepted value is 1.
- Bit 2 = Any port, whose related bit of the weighted_round_robin_weight_sharing parameter is set to 1'b1, do not have the same values in its ahbX_priorityY_relative_priority parameter.
- Bit 3 = For ports whose related bit of the weighted_round_robin_weight_sharing parameter is set to 1'b1, the values of the ahbX_port_ordering parameters are not sequential.

If bits 0, 2 or 3 are set to 1'b1 in the wrr_param_value_err parameter, and any of the ports are paired in the weighted_round_robin_weight_sharing parameter, all weight sharing data will be ignored during Memory Controller initialization and the ports will be prioritized by port number. If port pairing is not being used, but the bit 0 error condition is set to 1'b1, then ports with a non-unique port ordering are prioritized by port number.

Note: The user is strongly cautioned against modifying the values of the port ordering or relative priority parameters during active port usage.

10.5.10 Command queue with placement logic

From the Arbiter, commands are routed to the command queue of the Memory Controller core. The command queue is fed using a placement algorithm. For more information on this algorithm, refer to below Chapter “Core Command Queue with Placement Logic”.

10.6 Core command queue with placement logic

The Memory Controller core contains a command queue that accepts commands from the Arbiter. This command queue uses a placement algorithm to determine the order in which commands will be executed in the Memory Controller core. The placement logic follows many rules to determine where new commands should be inserted into the queue, relative to the contents of the command queue at the time.

Placement is determined by considering address collisions, source collisions, data collisions, command types and priorities. In addition, the placement logic attempts to maximize efficiency of the Memory Controller core through command grouping and bank splitting. Once placed into the command queue, the relative order of commands is constant.

Many of the rules used in placement may be individually enabled/disabled. In addition, the queue may be programmed by the placement_en parameter to disable the placement logic entirely, resulting in an in-line queue that attends requests in the order they are received. If the placement_en parameter is cleared to 1'b0, the placement algorithm will be ignored.

10.6.1 Rules of the placement algorithm

The factors affecting command placement together work to identify where a new command fits into the execution order. They are listed in order of importance.

Address collision/Data coherency violation

The order in which READ and WRITE commands are processed inside Memory Controller is critical to proper system behavior. While READs and WRITEs to different addresses are independent and may be re-ordered without affecting system performance, READs and WRITEs that access the same address are significantly related. If the port requests a READ after a WRITE to the same address, then repositioning the READ before the WRITE would return the original data, not the changed data. Similarly, if the READ was requested ahead of the WRITE but accidentally positioned after the WRITE, the READ would return the new data, not the original data prior to being overwritten. These are significant data coherency mistakes.

To avoid address collisions, READs or WRITEs that access the same chip select, bank and row as a command already in the command queue will be inserted into the command queue after the original command, even if the new command is of a higher priority.

This factor may be enabled/disabled through the addr_cmp_en parameter and should only be disabled if the system can guarantee coherency of READs and WRITEs.

Source ID collision

Each port is assigned a specific source ID that identifies the source uniquely. This allows the Memory Controller to map data from/to the correct source/destination.

Note: A source ID does contain port identification information which means that the rules for placement are dependent on the requesting port. There will not be source ID collisions between ports.

In general, commands of the same type from the same source ID will be placed in the command queue in order. Therefore, a READ/WRITE command with the same source ID of a READ/WRITE command being already in the command queue will be processed afterward.

The behavior of commands of different types from the same source ID is dependent on the user configuration. For Memory Controller, the placement of new READ/WRITE commands that collide in terms of source ID with existing entries in the command queue will only depend on other commands of the same type, not on different types. This means that, if there are no address conflicts, a READ command could be executed ahead of a WRITE command with the same source ID, as a WRITE command could be executed ahead of a READ command with the same source ID as well.

This feature is always enabled.

Write buffer collision

Incoming WRITE requests in the command queue are allocated to one of the 8 WRITE buffers of the Memory Controller core automatically based on availability. New WRITE commands will be designated to any available buffer. However, back-to-back write requests from a particular source ID will be allocated to the same WRITE buffer as the previous command.

Since the Memory Controller core must pull data out of the buffers in the order it was stored, if a WRITE command is linked to a buffer associated with another command in the queue, the new command will be placed in the command queue after that command, regardless of priority.

This feature is always enabled.

Priority

Priorities are used to distinguish important commands from less important commands. Each command is given a priority based on the command type through the programmable parameters ahbX_r_priority and ahbX_w_priority (X is the x-th port), where '0' value is the highest priority and '7' is the lowest.

The placement algorithm will attempt to place higher priority commands ahead lower priority commands, as long as they have no source ID, WRITE buffer or address collisions.

Higher priority commands will be placed lower in the command queue in case:

- They access the same address and
- They are from the same requestor or
- They use the same buffer used by lower priority commands being in the command queue already.

This feature is enabled through the priorities parameter.

Bank splitting

Before accesses to two different rows within the same bank could be performed, first active row must be closed (pre-charged) and the new row must be opened (activated). Therefore, the both activities require some timing overhead for optimization, the placement queue will attempt to insert the new command into the command queue such that commands to other banks may execute during this timing overhead.

Still the placement of the new commands will follow priority, source ID, WRITE buffer and address collision rules. The placement logic will also attempt to optimize the Memory Controller core, by inserting a command to the same bank of any existing command in the command queue, immediately after the original command. This reduces the overall timing overhead by potentially eliminating one pre-charge/activate cycle. This placement will only

be possible if there are no priority, source ID, WRITE buffer or address collisions or conflicts with other commands in the command queue.

Every bank splitting feature could be enabled through the bank_split_en parameter.

Read/Write grouping

The memory suffers a small timing overhead when switching from READ to WRITE mode. For efficiency, the placement queue will attempt to place a new read command sequentially with other read commands in the command queue, or a new WRITE command sequentially with other WRITE commands in the command queue. Grouping will only be possible if no priority, source ID, WRITE buffer or address collision rules are violated.

This feature is enabled through the rw_same_en parameter.

10.6.2 Command execution order after placement

Once a command has been placed in the command queue, its order relative to the other commands in the queue at that time is fixed. Even if this simplifies the algorithm there are some drawbacks. For this reason, Memory Controller offers two options that affect commands once they have been placed in the command queue.

High-Priority command swapping

Commands are assigned priority values to ensure that critical commands are executed before less important commands. Therefore, it is desirable that high-priority commands cross the Memory Controller core as sooner as possible. The placement algorithm sorts commands by priority though it can not avoid a such scenario in which a high-priority command is waiting at the top of the command queue while another command, possibly of a lower priority, is in process.

The high-priority command swapping feature allows this new high-priority command to be executed quite sooner. If the user has enabled the swapping function by the swap_en parameter, the entry at the top of the command queue will be compared with the current command in progress. If the command queue's top entry is of a higher priority (not the same priority), and it does not have an address, source ID or WRITE buffer conflict with the current command being executed, the original command will be interrupted.

If the command has to be interrupted, it will be halted after completing the current burst, stacked and placed at the top of the queue, while the new command will be executed. As long as the command queue is not full, new commands may continue to be inserted into the command queue based on the placement rules, even at the head of the queue ahead of the interrupted command. The top entry in the command queue will be executed next. Whenever the interrupted command is resumed, it will start from the point at which it was interrupted.

Note: Priority 0 commands could never be interrupted, so the user should set any commands that should not be interrupted to priority 0.

Command ageing

Since commands can be inserted ahead of existing commands in the command queue, it could happen a low priority command remains at the bottom of the queue indefinitely. To avoid such a lockout condition, aging counters have been included in the placement logic that measure the number of cycles that each command has been waiting. If command aging is enabled through the active_ageing parameter and an aging counter hits its maximum, the

priority of the associated command will be decremented by one (lower priority commands are executed first). This increases the likelihood that this command will move to the top of the command queue and be executed.

Note: This command does not move relative positions in the command queue when it ages; the new priority will be considered when placing new commands into the command queue.

Aging is controlled by a master aging-rate counter and command aging counters associated with each command in the command queue. The age_count and command_age_count parameters hold the initial values for each of these counters, respectively. When the master counter counts down the age_count value, a signal is sent to the command aging counters to decrement. When the command aging counters have completely decremented, the priority of the associated command is decremented by one and the counter is reset. Therefore, a command does not age by a priority level until the total elapsed cycles has reached the product of the age_count and command_age_count values. The maximum number of cycles that any command can wait in the command queue until reaching the top priority level is the product of the age_count value, the command_age_count value, and the number of priority levels in the system.

10.7 Low power operation

In many applications, it is highly desirable to minimize power consumption. The Memory Controller provides various user configurable low power options to manage power savings. In addition, a partial-array self-refresh option is included for mobile memory devices.

10.7.1 Low power modes

Five low power modes are available in the Memory Controller. The low power modes are listed from least to most power saving.

Note: It is not possible to exit one low power mode and enter another low power mode simultaneously. The user should plan for a minimum delay between exit and entry between the two low power modes of 15 cycles in which the Memory Controller must remain stable.

1. **Memory Power-Down**

   The Memory Controller sets the memory devices into power-down which reduces the overall power consumption of the system. This is the low power modes having the least effect: The Memory Controller and memory clocks are fully operational, but the CKE input bit toward the memory devices is de-asserted.

   The Memory Controller will continue to monitor memory refresh needs and will automatically bring the memory out of power-down to perform these refreshes. Whenever a refresh is required, the CKE bit will be turned enabled. This action drives memory devices out power-down. Once the refresh has been completed, the memory devices will be returned to power-down by de-asserting the CKE input bit.

2. **Memory Power-Down with Memory Clock Gating**

   The Memory Controller sets the memory devices into power-down and gates off the clock to the memory devices. Refresh operations will be handled as seen above for the Memory Power-Down mode (Mode 1), but the gating on the memory clock will be now removed before asserting the CKE pin. After the refresh has been completed, the memory devices will be returned to power-down with the clock gated. Before the memory devices are removed from power-down, the clock will be gated on again. Although this mode is supported in both mobile and non-mobile memory devices, clock gating while in power-down is only allowed for mobile memory devices. Therefore, the

Memory Controller will only attempt to gate the clock if it is configured for mobile device operation. For non-mobile memory devices in this low power mode, the Memory Controller will operate identically to the Memory Power-Down mode without the clock gating (Mode 1).

3. **Memory Self-Refresh**

   The Memory Controller sets the memory devices into self-refresh. In this mode, the Memory Controller and memory clocks are fully operational and the CKE input bit to the memory devices is de-asserted. Since the memory automatically refreshes its contents, the Memory Controller does not need to send explicit refreshes to the memory.

4. **Memory Self-Refresh with Memory Clock Gating**

   The Memory Controller sets the memory devices into self-refresh and gates off the clock to the memory devices. Before the memory devices are removed from self-refresh, the clock will be gated on again.

5. **Memory Self-Refresh with Memory and Controller Clock Gating**

   This is the most effective low power mode of the Memory Controller: The Memory Controller sets the memory devices self-refreshing and gates off the clock toward them. In addition, the clock toward the Memory Controller and the programming parameters will be gated off, except to a small portion of the DLL, which must remain active to maintain the lock. Before the memory devices are removed from self-refresh, the Memory Controller and memory clocks will be gated on.

10.7.2 Low power mode control

The Memory Controller may enter and exit the various low power modes in the following ways:

- Automatic Entry:
- When the Memory Controller is idle, 4 different timing counters begin counting the cycles of inactivity. If any of the counters expires, the Memory Controller enters the low power mode associated with that counter.
- Manual Entry:
- The user may set any low power mode by setting the bit of the lowpower_control parameter associated with the desired mode. The Memory Controller will enter the selected low power mode when it is has completed its current burst.

Automatic and Manual entry methods are both controlled by two parameters: lowpower_control and lowpower_auto_enable. The lowpower_control parameter contains individual enable/disable bits for each low power mode, and the lowpower_auto_enable parameter controls whether each mode will be entered automatically or manually.

Automatic entry

Automatic Entry will occur as all the following conditions are matched:

- The mode is programmed for automatic entry by setting the relevant bit in the lowpower_auto_enable parameter to 1'b1.
- The particular mode is enabled in the lowpower_control parameter.
- The Memory Controller is idle.
- The counter associated with this mode expires.

There are 4 counters to full cover the 5 low power modes. There are separate counters for each of the three memory self-refresh low power modes (Modes 3, 4 and 5). Memory Power-Down mode (Mode 1) and Memory Power-Down with Memory Clock Gating mode (Mode 2) share the same counter.

The counters determine the number of idle cycles before entering into the associated low power mode. Every counter is re-initialized each time there is a new READ or WRITE transaction entering or executing in the Memory Controller. This guarantees that the Memory Controller will not enter any of the low power modes when active.

Each low power mode can be entered through automatic entry, and will be exited automatically whenever any of the following conditions occur:

- A new READ or WRITE transaction appears at the Memory Controller interface.
- The Memory Controller must refresh the memory when in either of the power-down modes (Modes 1 or 2). After completing the memory refresh, the Memory Controller re-enters power-down.
- The counter for a deeper low power mode expires. The Memory Controller must exit the current low power mode in order to enter the deeper low power mode. A minimum of 15 cycles occur between exit from one low power mode before entering into the next low power mode, even if the counters expire within 15 cycles of each other.

Note: The Memory Controller will not enter a less deep low power mode, regardless of which counter expires.

Manual “On-Demand” entry

Manual Entry occurs when the 2 following conditions are matched:

- The mode is programmed for manual entry by clearing the relevant bit in the lowpower_auto_enable parameter to 1'b0.
- The particular mode is set to 1'b1 in the lowpower_control parameter.

For manual entry, the lowpower_control parameter triggers the entry into the low power modes. The Memory Controller does not need to be idle when the low power mode bit is enabled. When a particular mode that is programmed for manual entry is enabled, the Memory Controller will complete the current memory burst access, and then, regardless of the activity inside the Memory Controller or at the memory interface, it will enter the selected low power mode.

If any new transaction would involve the Memory Controller while it is in any of the low power modes, the transaction itself will be stacked inside the Memory Controller's command queue until the queue is completely full.

The way to exit from a manually-entered low power mode is manual too. Clearing the lowpower_control parameter bits to 1'b0 will trigger the Memory Controller to pull the memory devices out of powerdown or self-refresh, and command processing will resume.

Note: In the deepest low power mode (Mode 5), the clock toward the programming registers module is gated off. However, manual low power mode exit requires the user to clear the lowpower_control parameter to 1'b0, which is not possible when the clock is off. As a result, the user should not manually activate the deepest low power mode. If Memory Self-Refresh with Memory and Controller Clock Gating Mode (Mode 5) was entered manually, the device would not be able to be brought out of low power mode again.

If a different lowpower_control bit is set to 1'b1 while in one of the low power modes, or on clearing of the original bit to 1'b0, the Memory Controller will exit the current low power

mode. There will be at least a 15 cycle delay before the component being either fully operational or enters the new low power mode.

Register programming

The low power modes of the Memory Controller are controlled by the lowpower_control and lowpower_auto_enable parameters. These 5 bit parameters contain each one bit to control each low power mode. The lowpower_control parameter enables the associated low power mode, and the lowpower_auto_enable parameter sets the entry method into that mode as manual or automatic. *Table 70* displays the relationship between the 5 bits of the lowpower_control and lowpower_auto_enable parameters and the various low power modes. Parameters meaning is discussed in *Section 10.14: Summary of memory controller parameters*.

Table 70. Low power mode parameters

Low Power Mode	Enable	Entry
Memory Power-Down (Mode 1)	lowpower_control [4] =1'b1	lowpower_auto_enable [4]: 1'b0 = Manual 1'b1 = Automatic
Memory Power-Down with Memory Clock Gating (Mode 2)	lowpower_control [3] =1'b1	lowpower_auto_enable [3]: 1'b0 = Manual 1'b1 = Automatic
Memory Self-Refresh (Mode 3)	lowpower_control [2] =1'b1	lowpower_auto_enable [2]: 1'b0 = Manual 1'b1 = Automatic
Memory Self-Refresh with Memory Clock Gating (Mode 4)	lowpower_control [1] =1'b1	lowpower_auto_enable [1]: 1'b0 = Manual 1'b1 = Automatic
Memory Self-Refresh with Memory and Controller Clock Gating (Mode 5)	lowpower_control [0] =1'b1	lowpower_auto_enable [0]: 1'b0 = Manual 1'b1 = Automatic

When a lowpower_control parameter bit is set to 1'b1 by the user, the Memory Controller checks the lowpower_auto_enable parameter.

- If the associated bit in the lowpower_auto_enable parameter is set to 1'b1, the Memory Controller will watch the associated counter for expiration and then enter that low power mode.
- *Table 71* shows the correlation between the low power modes and the counters that control each mode's automatic entry.
- If the associated bit in the lowpower_auto_enable parameter is cleared to 1'b0, the Memory Controller will complete its current memory burst access and then enter the specified low power mode.

Table 71. Low power mode controls

Low Power Mode	Counter
Memory Power-Down (Mode 1)	lowpower_power_down_cnt
Memory Power-Down with Memory Clock Gating (Mode 2)	lowpower_power_down_cnt
Memory Self-Refresh (Mode 3)	lowpower_self_refresh_cnt
Memory Self-Refresh with Memory Clock Gating (Mode 4)	lowpower_external_cnt
Memory Self-Refresh with Memory and Controller Clock Gating (Mode 5)	lowpower_internal_cnt

Note: The values in lowpower_auto_enable parameter are only relevant when the associated lowpower_control bit is set to 1'b1.

Multiple bits of the lowpower_control and lowpower_auto_enable parameters can be set to 1'b1 at the same time. When this happens, the Memory Controller always enters the deepest low power mode of all the modes that are enabled.

If the Memory Controller is already in one low power mode when a deeper low power mode is automatically or manually requested, first it exits the current low power mode and then enters the deeper low power mode. A minimum 15 cycle delay occurs before the second entry occurs.

The timing for automatic entry into any of the low power modes is based on the number of idle cycles that have elapsed in the Memory Controller. There are 4 counters related to the 5 low power modes to determine when any particular low power mode will be entered if the automatic entry option is chosen. The counters are also shown in *Table 70*. Since the two power-down modes share one counter, wishing the user automatically enter Memory Power-Down mode (Mode 1), the Memory Power-Down with Memory Clock Gating mode (Mode 2) must not be enabled.

Controller clock gating and Re-locking

When the Memory Controller is in its deepest low power modes, the clock to the Memory Controller is gated off, except a small portion of the DLL to maintain the current frequency. Since the voltage and temperature differences of the chip can change and the DLL would not adjust to these changes, it is possible that the Memory Controller DLL shifts out of range of the controller clock.

The DLL can respond to slight variations of the frequency very quickly, but it requires a long time to manage large shifts. Therefore, it is better the Memory Controller to be periodically awoken from controller clock gating, so it can resynchronize to the controller clock and then resume controller clock gating.

The interval among synchronizations is user-defined by the lowpower_refresh_hold parameter. This value directly feeds a counter and sets the number of cycles that the Memory Controller will wait before attempting to re-lock the DLL.

Note: This is only relevant when the controller clock is gated (Mode 5).

When this counter expires, the DLL will be un-gated and the DLL will attempt to re-lock. The clock will be kept un-gated for at least 16 cycles, even if the DLL locks during that time. Once the DLL has re-locked and the 16 cycles have elapsed, the DLL controller clock will be gated again and the counter will restart countdown at the value in the lowpower_refresh_hold parameter.

Refresh masking

Regular refresh commands will be issued at the same intervals as the Memory Controller is operating normally, is idle, or is in any of the low power modes. However, for memory arrays with multiple chip selects, the Memory Controller can mask refreshes during any of the low power modes. By setting bits of the lowpower_refresh_enable parameter to 1'b1, auto-refreshes will be masked for the associated chip selects.

User should ensure that refreshes are not constantly masked, and that each chip select is refreshed periodically.

Low power and clock forwarding

The Memory Controller implements a clock forwarding scheme, which eliminates the need for a de-skew PLL and reduces the overall power consumption of the device. Controllers that are configured for low power should also be configured for clock forwarding.

Mobile DDR/SDRAM devices

- **Enabling mobile usage**

  When the device is used inside a mobile device, the parameter en_lowpower_mode must be set to 1'b1. This enables the Memory Controller to use the initialization sequence and EMRS addressing suitable for mobile equipments. When the en_lowpower_mode parameter is cleared to 1'b0, a standard DDR SDRAM or SDRAM device may be used.

- **Partial array self-refresh**

  For mobile applications, the Memory Controller is able to support refresh operations to subsections of the memory array. To simplify this duty, separate parameters are provided to supply the EMRS data for each chip select. These are the emrs_data_X parameters, where X represents the chip select.

  Having separate control parameters for the EMRS data allows the individual chips to set their own masked refresh. The write_modereg parameter controls the writing of this EMRS data into the registers. When write_modereg is set to 1'b1 initially, the EMRS register of chip select 0 will be written. Each subsequent setting of the write_modereg parameter to 1'b1 will write the EMRS register of the next chip select (1, 2, then 3).

Note: The Memory Controller does not check if operations attempt to access addresses outside the refresh ranges set by the EMRS registers, so any access to these addresses may result in corrupt or lost data.

10.8 Additional features

10.8.1 Out-of-range address checking

It is possible that the master attempts to write to an invalid address. For this reason, all incoming addresses are always checked against the addressable physical memory space. If a transaction is addressed to an out-of-range memory location, bit 0 of the int_status parameter will be set to 1'b1 to alert the user of this condition. The Memory Controller will record the address, source ID, length and type of transaction that caused the out-of-range interrupt in the out_of_range_addr, out_of_range_source_id, out_of_range_length and out_of_range_type parameters.

Reading the out-of-range parameters will trigger the Memory Controller to empty these parameters and allow them to store out-of-range access information for future errors. The interrupt should be acknowledged by setting bit 0 of the int_ack parameter to 1'b1, which will in turn cause bit 0 of the int_status parameter to be cleared to 1'b0.

If a second out-of-range access occurs before the first out-of-range interrupt is acknowledged, bit 1 of the int_status parameter will be set to 1'b1 to indicate that multiple out-of-range accesses have occurred. If the out-of-range parameters have been read when the second out-of-range error occurs, the details for this transaction will be stored in the out-of-range parameters. If they have not been read, the details of the second error will be lost.

Even though the address has been identified as erroneous, the Memory Controller will still process the READ or WRITE transaction. A READ transaction will return random data which the user must receive to avoid stalling the Memory Controller. A WRITE transaction will write the associated data to an unknown location in the memory array, potentially over-writing other stored data. The command can not be aborted once accepted into the Memory Controller.

10.8.2 Mobile devices DQS

For Mobile applications the user must add pull-down resistors on the DRAM boundary to the DQS and DQS_n pins. These resistors enable the system to open the gate early without receiving bad data. Both the resistors are very important and the system can not function accurately without them.

10.8.3 Half datapath option

Memory Controller can also reduce the usable size of the bus between the Memory Controller itself and memory devices. This feature is very useful when a different memory part, having a smaller data width, is utilized. To use a memory device with a smaller datapath, the half datapath option could be enabled setting the programmable reduce parameter to 1'b1. The memory interface consists of the data signal (DQ) [15:0], data strobe (DQS) [1:0] and data mask (DM) [1:0]. When the reduce parameter is set to 1'b1, only the lower half of the memory interface is used. In this setting, the upper half bits of the dm_disable, dqs_disable and data_disable signals are driven high, causing the upper half of the data and data strobe buses to be driven low. The upper half of the data mask bus is driven high.

If the reduce parameter is cleared to 1'b0, the Memory Controller will ignore the half datapath option and work normally. In this case, the entire memory interface will be used. *Table 72* displays the effect of the Half Datapath option on the memory interface buses.

Table 72. Memory interface buses with Half Datapath option

Signal	Reduce parameter is set (1‘b1)		Reduce parameter is cleared(1‘b0)	
	Bus size	Relevant bit	Bus size	Relevant bit
Data DQ	[15:00]	[07:00]	[15:00]	[15:00]
Data DQS	[01:00]	[00:00]	[01:00]	[01:00]
Data DM	[01:00]	[00:00]	[01:00]	[01:00]

10.8.4 User-defined registers

Memory Controller contains two user-defined parameters. These register-width size parameters hold user-defined values that will be available as output signals param_user_def_reg_X (X is 0 or 1) at the Memory Controller core (stp_memcd) level. These parameters have no effect on the Memory Controller except that utilizing addresses in the register map.

Only a single bit of the user-defined registers is used in this configuration. All other bits of user_def_reg_0 and user_def_reg_1 are available if customer hookup is required.

Bit 0 of user_def_reg_0 controls the READ data retime function. This function provides the user with a choice in clock synchronization paths in the PHY.

An alternative path has been created allowing the PHY to synchronize the READ data to the core clock domain and then pass the data to the read FIFO synchronously, i.e. in 1one clock cycle. This constrains the 1/4 cycle path for clock synchronization within the PHY between two flip-flops which are easily co-located, however it adds a cycle of latency to the path.

Bit 0 of user_def_reg_0 allows the user to select the desired path. If the bit is cleared to 1'b0, the alternative path is used and synchronization occurs in the PHY, incurring the extra cycle of latency. If the bit is set to 1'b1, the standard path is used.

10.9 Address mapping

The Memory Controller automatically maps user addresses to the DRAM memory in a contiguous block. Address map begins at user address 0 and finishes at the highest available address according to the size and number of DRAM devices present. This mapping is dependent on how the Memory Controller is configured and how the parameters in the internal Memory Controller registers are programmed (*Section 10.13*). The exact number and values of these parameters depends on the configuration and the type of memory for which the Memory Controller was designed.

The mapping of the address space to the internal data storage structure of the DRAM devices is based on the actual size of the DRAM devices available. The size is stored in user-programmable parameters that must be initialized at power up. Certain DRAM devices allow for different mapping options to be chosen, while other DRAM devices depend on the burst length chosen.

10.9.1 DDR SDRAM address mapping options

The address structure of DDR SDRAM devices contains five fields. Each of these fields can be individually addressed when accessing the DRAM. The address map for this Memory Controller is ordered as follows:

```
"Chip Select - Row - Bank - Column - Datapath"
```

The maximum widths of the fields are based on the configuration settings. The actual widths of the fields may be smaller if the device address width parameters (addr_pins, eight_bank_mode and column_size) are programmed differently.

10.9.2 Maximum address space

The maximum user address range is determined by the width of the memory datapath, the number of chip select pins, and the address space of the DRAM device. The maximum amount of memory can be calculated by the following formula:

```
Max Memory Bytes = ChipSelects x 2Address x NumBanks x DPWidthBytes
```

For this Memory Controller, the maximum values for these fields are as follows:

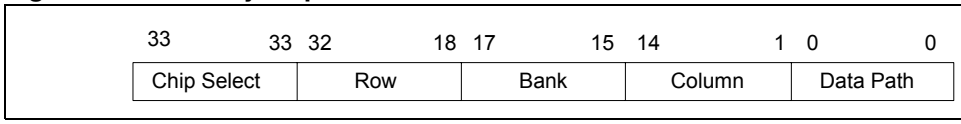
- Chip Selects = 2
- Device Address = 15+14 (Row+Column)
- Number of Banks = 8
- Datapath Width in Bytes = 2 bytes

As a result, the maximum accessible memory area is 16 GB.

10.9.3 Memory mapping to address space

The maximum allowable address space and mapping into the DRAM devices for the Memory Controller is shown in *Figure 11*. This map corresponds to a memory device with 15 row bits and 14 column bits.

Figure 11. Memory map: Maximum



The addr_pins and column_size parameters can range from the maximum configured for the Memory Controller to seven bits smaller than the maximum configured. This allows the Memory Controller to work with a wide variety of memory device sizes.

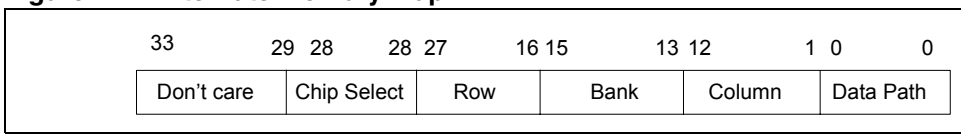
The settings for the addr_pins and column_size parameters control how the address map is used to decode the user address to the DRAM chip selects and row and column addresses. The eight_bank_mode parameters control the address in DDR2 mode. It is assumed that the values in these parameters never exceed the maximum values configured.

Using the example shown in *Figure 11*, if the Memory Controller is wired to devices with 12 row pins and 12 column bits, the maximum accessible memory space would be reduced. The accessible memory space for this configuration is 512 MB.

The address map for this configuration is shown in *Figure 12*.

Note: Address bits 29 through 33 are not used. These bits are ignored when generating the address to the DRAM devices

Figure 12. Alternate memory map



Note: The Chip Select, Row, Bank, and Column fields are used to address an entire memory word, and the Datapath bits are used to address individual bytes within that user word. For example, for a read starting at byte address 0x2, the Datapath bits must be defined as 3'b010 in order to address this byte directly. READs and WRITEs are memory word-aligned if all the Datapath bits are 0.

10.10 DCC tuning timing

The command and address for the transaction are sent from the memory controller coincident with the falling edge of the memory controller clock. Since the clock, command, and address signals will all have roughly the same pad and flight delays to travel to the memory, the rising edge of the clock at the memory will be centered with the command and address signals, allowing reliable capture. To ensure proper memory read write sequences the DDR memory controller contains a delay compensation circuit that, in conjunction with I/O cell circuitry, can be used to meet the memory target timing requirements. The delay compensation circuit offers the following features:

1. **Programmable read clock delay specified as a percentage of a clock cycle:**

Read transfer path, should also take in account the memory the flight paths. There is a certain time lag from when the clock is sent from the memory controller to when the data and DQS signals are received at the memory controller from the memory. Since the DQS from the memory will be sent coincident with the data, and the data must be captured reliably, the DQS signal is delayed through the register field dll_dqs_delay_X so that it is centered in the data valid window (nominally approximately 1/4 cycle).

2. **Programmable write clock and write DQS delays specified as percentages of a clock cycle:**

Write transfer path are control from dqs_out_shift and wr_dqs_shift register parameters which set the delay for the DQS signal for dll_wr_dqs_slice and for the clk_wr signal, respectively. These parameters should be programmed such that clk_dqs_out is in phase with clk and that clk_wr is 1/4 cycle ahead of clk_dqs_out.

3. **Delay compensation circuit re-sync circuitry activated during refresh cycles to compensate for temperature and voltage drift:**

The delay compensation circuitry relies on a master/slave approach. There is a master delay line which is used to determine how many delay elements constitute a complete cycle. This count is used, along with the programmable fractional delay settings, to determine the actual number of delay elements to program into the slave delay lines. The master and slave delay lines are identical. This approach allows the memory controller to observe a clock and then delay other signals a fixed percentage of that clock. The DCC logic does not actively generate clock signals.

The delay parameters are listed in *Table 73*. The total delay can be determined based on the following equation, where param is one of the parameters in the table:

delay = #delays in one cycle × (param[6:0]) /128

Table 73. Delay parameters

Operation	Parameter
Clock	wr_dqs_shift
Read	dll_dqs_delay_X
Write	dqs_out_shift

1. Separate delay chains for each DQS signal from the DRAM devices.
2. Support for multiple DQ:DQS ratios

The DQS bus is a bidirectional bus that is driven by the memory controller on writes and the memory on reads. When neither device is driving the bus, DQS will remain in a high-

impedance state. However, DQS is only relevant to the memory controller during reads in order to capture valid data. For this reason, the DQS signal from memory must be gated so that it is ignored at all other times. Gating of the DQS signal is shown in Figure 4, DQS gating.

The timing of when to start gating the DQS depends on the design itself, the flight time of the clock to memory, and the flight time of the data/DQS to the memory controller, as follows:

- If the round trip time is between ½ cycle and 1½ cycles, program the caslat_lin parameter equal to the caslat parameter.
- If the round trip time is less than ½ cycle, program the caslat_lin parameter one value less (which translates to ½ cycle) than the caslat parameter to open the gate ½ cycle sooner.
- If the round trip time is longer than 1½ cycles, program the caslat_lin parameter one value more (which translates to ½ cycle) than the caslat parameter to open the gate ½ cycle later.

In addition, the caslat_lin_gate parameter controls the opening of the gating signal. Nominally, caslat_lin_gate should have the same value as the caslat_lin parameter. However, to accommodate the skew of the memory devices, it may be necessary to open the gate a 1/2-cycle sooner or later. Adjusting the value of caslat_lin_gate modifies the gate opening by this factor.

10.11 External pin connection Of DDR interface in SPEAr300

Refer to *Chapter 5: Pin description*.

10.12 Initialization protocol

After power on, once the power supply to memory devices and the device itself is stable, the Memory Controller must be initialized: afterward it will automatically initialize the memory devices.

The procedure to initialize the Memory Controller is as follows:

1. Clear the rst_n signal by driving it to 1'b0. All programmable registers will be cleared.
2. Set the rst_n signal synchronously with the Memory Controller clock by driving the signal to 1'b1.
3. Issue write register commands to configure the DRAM protocols and the settings for the DCC. Keep the start parameter de-asserted during this initialization step. For more details, refer to *Section 10.13*.
4. Assert the start parameter. This triggers the Memory Controller to execute the initialization sequence using the parameters written into the registers.

The Memory Controller will automatically initialize the DRAM memory devices and lock the internal DCC. The DLL will process and send a signal to the initialization block when it has locked.

Table 74. MT47H128M8-3 (DDR2@333 MHz cl5) initialization table

Register name	Value	Register name	Value
MEM0_CTL	0x01010101	MEM55_CTL	0x00000000
MEM1_CTL	0x00000101	MEM56_CTL	0x5B1C00C8
MEM2_CTL	0x01000000	MEM57_CTL	0x00C8002E
MEM3_CTL	0x00000101	MEM58_CTL	0x00000000
MEM4_CTL	0x00000101	MEM59_CTL	0x00000043
MEM5_CTL	0x01000000	MEM60_CTL	0x00000000
MEM6_CTL	0x00010001	MEM61_CTL	0x00000000
MEM7_CTL	0x00000100	MEM62_CTL	0x00000000
MEM8_CTL	0x00010001	MEM63_CTL	0x00000000
MEM9_CTL	0x01020003	MEM64_CTL	0x00000000
MEM10_CTL	0x01000102	MEM65_CTL	0x001C0000
MEM11_CTL	0x02000102	MEM66_CTL	0x0019001C
MEM12_CTL	0x02020102	MEM67_CTL	0x005A0000
MEM13_CTL	0x03020202	MEM68_CTL	0x001E007A
MEM14_CTL	0x02040202	MEM69_CTL	0x00000000
MEM15_CTL	0x00000002	MEM70_CTL	0x00000000
MEM16_CTL	0x00000000	MEM71_CTL	0x00000000
MEM17_CTL	0x03000405	MEM72_CTL	0x00000000
MEM18_CTL	0x03030002	MEM73_CTL	0x00000000
MEM19_CTL	0x04000305	MEM74_CTL	0x00000000
MEM20_CTL	0x0505053F	MEM75_CTL	0x00000000
MEM21_CTL	0x05050505	MEM76_CTL	0x00000000
MEM22_CTL	0x04040405	MEM77_CTL	0x00000000
MEM23_CTL	0x04040404	MEM78_CTL	0x00000000
MEM24_CTL	0x03030304	MEM79_CTL	0x00000000
MEM25_CTL	0x03030303	MEM80_CTL	0x00000000
MEM26_CTL	0x02020203	MEM81_CTL	0x00000000
MEM27_CTL	0x02020202	MEM82_CTL	0x00000000
MEM28_CTL	0x01010102	MEM83_CTL	0x00000000
MEM29_CTL	0x01010101	MEM84_CTL	0x00000000
MEM30_CTL	0x00000001	MEM85_CTL	0x00000000
MEM31_CTL	0x00000000	MEM86_CTL	0x00000000
MEM32_CTL	0x00000000	MEM87_CTL	0x00000000
MEM33_CTL	0x00000000	MEM88_CTL	0x00000000
MEM34_CTL	0x0A0C0A00	MEM89_CTL	0x00000000

Table 74. MT47H128M8-3 (DDR2@333 MHz cl5) initialization table (continued)

Register name	Value	Register name	Value
MEM35_CTL	0x0000023F	MEM90_CTL	0x00000000
MEM36_CTL	0x00050A00	MEM91_CTL	0x00000000
MEM37_CTL	0x0D000000	MEM92_CTL	0x00000000
MEM38_CTL	0x00001302	MEM93_CTL	0x00000000
MEM39_CTL	0x00001E1E	MEM94_CTL	0x00000000
MEM40_CTL	0x7F000000	MEM95_CTL	0x00000000
MEM41_CTL	0x005F0000	MEM96_CTL	0x00000000
MEM42_CTL	0x2B050E00	MEM97_CTL	0x00000000
MEM43_CTL	0x00640064	MEM98_CTL	0x00000000
MEM44_CTL	0x00640064	MEM99_CTL	0x00000000
MEM45_CTL	0x00000064	MEM100_CTL	0x01010001
MEM46_CTL	0x00000000	MEM101_CTL	0x01000000
MEM47_CTL	0x00200020	MEM102_CTL	0x00000001
MEM48_CTL	0x00200020	MEM103_CTL	0x00000000
MEM49_CTL	0x00200020	MEM104_CTL	0x00000000
MEM50_CTL	0x00200020	MEM105_CTL	0x00000000
MEM51_CTL	0x00200020	MEM106_CTL	0x00000000
MEM52_CTL	0x00000000	MEM107_CTL	0x00860000
MEM53_CTL	0x00000000	MEM108_CTL	0x00000002
MEM54_CTL	0x00000A24		

Table 75. MT46H6M16LF-6(Low Power DDR @166 MHz cl3) initialization table

Register name	Value	Register n ame	Value
MEM0_CTL	0x03030301	MEM55_CTL	0x00000000
MEM1_CTL	0x00000303	MEM56_CTL	0x2D890000
MEM2_CTL	0x01000000	MEM57_CTL	0x00180018
MEM3_CTL	0x00000001	MEM58_CTL	0x00000000
MEM4_CTL	0x00000001	MEM59_CTL	0x00000022
MEM5_CTL	0x01000000	MEM60_CTL	0x00000000
MEM6_CTL	0x00010001	MEM61_CTL	0x00000000
MEM7_CTL	0x00000100	MEM62_CTL	0x00000000
MEM8_CTL	0x00010001	MEM63_CTL	0x00000000
MEM9_CTL	0x01020003	MEM64_CTL	0x00000000
MEM10_CTL	0x01000102	MEM65_CTL	0x003A0000
MEM11_CTL	0x02000102	MEM66_CTL	0x0030003A

Table 75. MT46H6M16LF-6(Low Power DDR @166 MHz cl3) initialization table

Register name	Value	Register n ame	Value
MEM12_CTL	0x02020102	MEM67_CTL	0x00B70000
MEM13_CTL	0x03020202	MEM68_CTL	0x003C00F4
MEM14_CTL	0x02040202	MEM69_CTL	0x00000000
MEM15_CTL	0x00000002	MEM70_CTL	0x00000000
MEM16_CTL	0x00000000	MEM71_CTL	0x00000000
MEM17_CTL	0x01000403	MEM72_CTL	0x00000000
MEM18_CTL	0x02020002	MEM73_CTL	0x00000000
MEM19_CTL	0x01000103	MEM74_CTL	0x00000000
MEM20_CTL	0x0505053F	MEM75_CTL	0x00000000
MEM21_CTL	0x05050505	MEM76_CTL	0x00000000
MEM22_CTL	0x04040405	MEM77_CTL	0x00000000
MEM23_CTL	0x04040404	MEM78_CTL	0x00000000
MEM24_CTL	0x03030304	MEM79_CTL	0x00000000
MEM25_CTL	0x03030303	MEM80_CTL	0x00000000
MEM26_CTL	0x02020203	MEM81_CTL	0x00000000
MEM27_CTL	0x02020202	MEM82_CTL	0x00000000
MEM28_CTL	0x01010102	MEM83_CTL	0x00000000
MEM29_CTL	0x01010101	MEM84_CTL	0x00000000
MEM30_CTL	0x00000001	MEM85_CTL	0x00000000
MEM31_CTL	0x00000000	MEM86_CTL	0x00000000
MEM32_CTL	0x00000000	MEM87_CTL	0x00000000
MEM33_CTL	0x00000000	MEM88_CTL	0x00000000
MEM34_CTL	0x05060A00	MEM89_CTL	0x00000000
MEM35_CTL	0x0000023F	MEM90_CTL	0x00000000
MEM36_CTL	0x00030600	MEM91_CTL	0x00000000
MEM37_CTL	0x0A000000	MEM92_CTL	0x00000000
MEM38_CTL	0x00000A02	MEM93_CTL	0x00000000
MEM39_CTL	0x00001E1E	MEM94_CTL	0x00000000
MEM40_CTL	0x70000000	MEM95_CTL	0x00000000
MEM41_CTL	0x00250000	MEM96_CTL	0x00000000
MEM42_CTL	0x18030700	MEM97_CTL	0x00000000
MEM43_CTL	0x00640064	MEM98_CTL	0x00000001
MEM44_CTL	0x00640064	MEM99_CTL	0x00000000
MEM45_CTL	0x00000064	MEM100_CTL	0x01010001
MEM46_CTL	0x00000000	MEM101_CTL	0x01010001

Table 75. MT46H6M16LF-6(Low Power DDR @166 MHz cl3) initialization table

Register name	Value	Register n ame	Value
MEM47_CTL	0x00200020	MEM102_CTL	0x00000001
MEM48_CTL	0x00200020	MEM103_CTL	0x00000000
MEM49_CTL	0x00200020	MEM104_CTL	0x00000000
MEM50_CTL	0x00200020	MEM105_CTL	0x00000000
MEM51_CTL	0x00200020	MEM106_CTL	0x00000000
MEM52_CTL	0x00000000	MEM107_CTL	0x00000000
MEM53_CTL	0x00000000	MEM108_CTL	0x00000001
MEM54_CTL	0x00000283		

10.13 Register interface

10.13.1 Register overview

A register may contain multiple parameters, a single parameter, or partial data for a parameter. As a result, a READ from or a WRITE to a particular parameter may require multiple READ or WRITE commands to different register addresses.

While parameters can be of any size, each parameter is mapped to byte boundaries that will fit the entire parameter. Unused bits are considered reserved and indicated with a RESV tag. Reserved fields will return 0 on all register reads.

Table 76. Parameter size to mapping conditions

Parameter size (in Bits)	Mapping size	Starting address
1 to 8	1 byte	Byte Boundary
9 to 16	2 bytes	2 Byte Boundary
17 to 128	4 bytes	4 Byte Boundary

10.13.2 MPMC base address In SPEAr300

Base address = 0xFC60.0000

10.13.3 Register map

Note: The address refers to the register address reg_addr, not a signal on the command address line. The registers are not byte addressable unless the register width is defined as 8 bits. To read or write a single parameter, use the register mask to mask other bits.

Table 77. Registers overview

Register name	Offset	Mem. CTRL core Reg. Address	Type(1)	Parameter(s)
MEM0_CTL	0x00	0x00	RW RW RW RW	AHB2_FIFO_TYPE_REG AHB1_FIFO_TYPE_REG AHB0_FIFO_TYPE_REG ADDR_CMP_EN
MEM1_CTL	0x04	0x01	RW RW	AHB4_FIFO_TYPE_REG AHB3_FIFO_TYPE_REG
MEM2_CTL	0x08	0x02	RW RW WR RW	BANK_SPLIT_EN AUTO_REFRESH_MODE AREFRESH AP
MEM3_CTL	0x0C	0x03	RW RD RW RW	DLL_BYPASS_MODE DLLLOCKREG DDRII_SDRAM_MODE CONCURRENTAP
MEM4_CTL	0x10	0x04	RW RW RW RW	INTRPTAPBURST FAST_WRITE EIGHT_BANK_MODE DQS_N_EN
MEM5_CTL	0x14	0x05	RW RW RW RW	ODT_ADD_TURN_CLK_EN NO_CMD_INIT INTRPTWRITEA INTRPTREADA
MEM6_CTL	0x18	0x06	RW RW RW RW	REDUC PRIORITY_EN POWER_DOWN PLACEMENT_EN
MEM7_CTL	0x1C	0x07	RW RW+ RW RW	START SREFRESH RW_SAME_EN REG_DIMM_ENABLE
MEM8_CTL	0x20	0x08	WR RW RW RW	WRITE_MODEREG WRITEINTERP WEIGHTED_ROUND_ROBIN_LATENCY_CONTROL TRAS_LOCKOUT
MEM9_CTL	0x24	0x09	RW RW RD RW	ODT_RD_MAP_CS1 ODT_RD_MAP_CS0 MAX_CS_REG CS_MAP

Table 77. Registers overview (continued)

Register name	Offset	Mem. CTRL core Reg. Address	Type(1)	Parameter(s)
MEM10_CTL	0x28	0x0A	RW RD RW RW	RTT_0 OUT_OF_RANGE_TYPE ODT_WR_MAP_CS1 ODT_WR_MAP_CS0
MEM11_CTL	0x2C	0x0B	RW RW RW RW	AHB0_R_PRIORITY AHB0_PORT_ORDERING ADDR_PINS RTT_PAD_TERMINATION
MEM12_CTL	0x30	0x0C	RW RW RW RW	AHB1_W_PRIORITY AHB1_R_PRIORITY AHB1_PORT_ORDERING AHB0_W_PRIORITY
MEM13_CTL	0x34	0x0D	RW RW RW RW	AHB3_PORT_ORDERING AHB2_W_PRIORITY AHB2_R_PRIORITY AHB2_PORT_ORDERING
MEM14_CTL	0x38	0x0E	RW RW RW RW	AHB4_R_PRIORITY AHB4_PORT_ORDERING AHB3_W_PRIORITY AHB3_R_PRIORITY
MEM15_CTL	0x3C	0x0F	RW	AHB4_W_PRIORITY
MEM16_CTL	0x40	0x10	-	This register intentionally blank.
MEM17_CTL	0x44	0x11	RW RD RW RW	TCKE OUT_OF_RANGE_SOURCE_ID COLUMN_SIZE CASLAT
MEM18_CTL	0x48	0x12	RW RW RW	TRTP TRRD TEMRS
MEM19_CTL	0x4C	0x13	RW RW RW RW	WRLAT WEIGHTED_ROUND_ROBIN_WEIGHT_SHARING TWTR TWR_INT
MEM20_CTL	0x50	0x14	RW RW RW RW	AHB0_PRIORITY2_RELATIVE_PRIORITY AHB0_PRIORITY1_RELATIVE_PRIORITY AHB0_PRIORITY0_RELATIVE_PRIORITY AGE_COUNT

Table 77. Registers overview (continued)

Register name	Offset	Mem. CTRL core Reg. Address	Type(1)	Parameter(s)
MEM21_CTL	0x54	0x15	RW RW RW WR	AHB0_PRIORITY6_RELATIVE_PRIORITY AHB0_PRIORITY5_RELATIVE_PRIORITY AHB0_PRIORITY4_RELATIVE_PRIORITY AHB0_PRIORITY3_RELATIVE_PRIORITY
MEM22_CTL	0x58	0x16	RW RW RW RW	AHB1_PRIORITY2_RELATIVE_PRIORITY AHB1_PRIORITY1_RELATIVE_PRIORITY AHB1_PRIORITY0_RELATIVE_PRIORITY AHB0_PRIORITY7_RELATIVE_PRIORITY
MEM23_CTL	0x5C	0x17	RW RW RW RW	AHB1_PRIORITY6_RELATIVE_PRIORITY AHB1_PRIORITY5_RELATIVE_PRIORITY AHB1_PRIORITY4_RELATIVE_PRIORITY AHB1_PRIORITY3_RELATIVE_PRIORITY
MEM24_CTL	0x60	0x18	RW RW RW RW	AHB2_PRIORITY2_RELATIVE_PRIORITY AHB2_PRIORITY1_RELATIVE_PRIORITY AHB2_PRIORITY0_RELATIVE_PRIORITY AHB1_PRIORITY7_RELATIVE_PRIORITY
MEM25_CTL	0x64	0x19	RW RW RW RW	AHB2_PRIORITY6_RELATIVE_PRIORITY AHB2_PRIORITY5_RELATIVE_PRIORITY AHB2_PRIORITY4_RELATIVE_PRIORITY AHB2_PRIORITY3_RELATIVE_PRIORITY
MEM26_CTL	0x68	0x1A	RW RW RW RW	AHB3_PRIORITY2_RELATIVE_PRIORITY AHB3_PRIORITY1_RELATIVE_PRIORITY AHB3_PRIORITY0_RELATIVE_PRIORITY AHB2_PRIORITY7_RELATIVE_PRIORITY
MEM27_CTL	0x6C	0x1B	RW RW RW RW	AHB3_PRIORITY6_RELATIVE_PRIORITY AHB3_PRIORITY5_RELATIVE_PRIORITY AHB3_PRIORITY4_RELATIVE_PRIORITY AHB3_PRIORITY3_RELATIVE_PRIORITY
MEM28_CTL	0x70	0x1C	RW RW RW RW	AHB4_PRIORITY2_RELATIVE_PRIORITY AHB4_PRIORITY1_RELATIVE_PRIORITY AHB4_PRIORITY0_RELATIVE_PRIORITY AHB3_PRIORITY7_RELATIVE_PRIORITY
MEM29_CTL	0x74	0x1D	RW RW RW RW	AHB4_PRIORITY6_RELATIVE_PRIORITY AHB4_PRIORITY5_RELATIVE_PRIORITY AHB4_PRIORITY4_RELATIVE_PRIORITY AHB4_PRIORITY3_RELATIVE_PRIORITY
MEM30_CTL	0x78	0x1E	RW	AHB4_PRIORITY7_RELATIVE_PRIORITY
MEM31_CTL	0x7C	0x1F	-	This register intentionally blank.
MEM32_CTL	0x80	0x20	-	This register intentionally blank.
MEM33_CTL	0x84	0x21	-	This register intentionally blank.

Table 77. Registers overview (continued)

Register name	Offset	Mem. CTRL core Reg. Address	Type(1)	Parameter(s)
MEM34_CTL	0x88	0x22	RW RW RW	CASLAT_LIN_GATE CASLAT_LIN APREBIT
MEM35_CTL	0x8C	0x23	RD RD RW RW	MAX_ROW_REG MAX_COL_REG INITAREF COMMAND_AGE_COUNT
MEM36_CTL	0x90	0x24	RD RW RW RW	WRR_PARAM_VALUE_ERR TRP TDAL Q_FULLNESS
MEM37_CTL	0x94	0x25	RW RW RW WR	TFAW OCD_ADJUST_PUP_CS_0 OCD_ADJUST_PDN_CS_0 INT_ACK
MEM38_CTL	0x98	0x26	RD RW RW RW	INT_STATUS INT_MASK TRC TMRD
MEM39_CTL	0x9C	0x27	RW RW	DLL_DQS_DELAY_1 DLL_DQS_DELAY_0
MEM40_CTL	0xA0	0x28	RW	DQS_OUT_SHIFT
MEM41_CTL	0xA4	0x29	RW	WR_DQS_SHIFT
MEM42_CTL	0xA8	0x2A	RW RW RW	TRFC TRCD_INT TRAS_MIN
MEM43_CTL	0xAC	0x2B	RW RW	AHB1_PRIORITY_RELAX AHB0_PRIORITY_RELAX
MEM44_CTL	0xB0	0x2C	RW RW	AHB3_PRIORITY_RELAX AHB2_PRIORITY_RELAX
MEM45_CTL	0xB4	0x2D	RW	AHB4_PRIORITY_RELAX
MEM46_CTL	0xB8	0x2E	RD	OUT_OF_RANGE_LENGTH
MEM47_CTL	0xBC	0x2F	RW RW	AHB0_WRCNT AHB0_RDCNT
MEM48_CTL	0xC0	0x30	RW RW	AHB1_WRCNT AHB1_RDCNT
MEM49_CTL	0xC4	0x31	RW RW	AHB2_WRCNT AHB2_RDCNT

Table 77. Registers overview (continued)

Register name	Offset	Mem. CTRL core Reg. Address	Type(1)	Parameter(s)
MEM50_CTL	0xC8	0x32	RW RW	AHB3_WRCNT AHB3_RDCNT
MEM51_CTL	0xCC	0x33	RW RW	AHB4_WRCNT AHB4_RDCNT
MEM52_CTL	0xD0	0x34	-	This register intentionally blank.
MEM53_CTL	0xD4	0x35	-	This register intentionally blank.
MEM54_CTL	0xD8	0x36	RW	TREF
MEM55_CTL	0xDC	0x37	RW	EMRS3_DATA
MEM56_CTL	0xE0	0x38	RW RW	TRAS_MAX TDLL
MEM57_CTL	0xE4	0x39	RW RW	TXSR TXSNR
MEM58_CTL	0xE8	0x3A	RD	VERSION
MEM59_CTL	0xEC	0x3B	RW	TINIT
MEM60_CTL	0xF0	0x3C	RD	OUT_OF_RANGE_ADDR[31:0]
MEM61_CTL	0xF4	0x3D	RD	OUT_OF_RANGE_ADDR[33:32]
MEM62_CTL	0xF8	0x3E	-	This register intentionally blank.
MEM63_CTL	0xFC	0x3F	-	This register intentionally blank.
MEM64_CTL	0x100	0x40	-	This register intentionally blank.
MEM65_CTL	0x104	0x41	RW	DLL_DQS_DELAY_BYPASS_0
MEM66_CTL	0x108	0x42	RW RW	DLL_INCREMENT DLL_DQS_DELAY_BYPASS_1
MEM67_CTL	0x10C	0x43	RW RD	DLL_START_POINT DLL_LOCK
MEM68_CTL	0x110	0x44	RW RW	WR_DQS_SHIFT_BYPASS DQS_OUT_SHIFT_BYPASS
MEM69_CTL	0x114	0x45	-	This register intentionally blank.
MEM70_CTL	0x118	0x46	-	This register intentionally blank.
MEM71_CTL	0x11C	0x47	-	This register intentionally blank.
MEM72_CTL	0x120	0x48	-	This register intentionally blank.
MEM73_CTL	0x124	0x49	-	This register intentionally blank.
MEM74_CTL	0x128	0x4A	-	This register intentionally blank.
MEM75_CTL	0x12C	0x4B	-	This register intentionally blank.
MEM76_CTL	0x130	0x4C	-	This register intentionally blank.
MEM77_CTL	0x134	0x4D	-	This register intentionally blank.

Table 77. Registers overview (continued)

Register name	Offset	Mem. CTRL core Reg. Address	Type(1)	Parameter(s)
MEM78_CTL	0x138	0x4E	-	This register intentionally blank.
MEM79_CTL	0x13C	0x4F	-	This register intentionally blank.
MEM80_CTL	0x140	0x50	-	This register intentionally blank.
MEM81_CTL	0x144	0x51	-	This register intentionally blank.
MEM82_CTL	0x148	0x52	-	This register intentionally blank.
MEM83_CTL	0x14C	0x53	-	This register intentionally blank.
MEM84_CTL	0x150	0x54	-	This register intentionally blank.
MEM85_CTL	0x154	0x55	-	This register intentionally blank.
MEM86_CTL	0x158	0x56	-	This register intentionally blank.
MEM87_CTL	0x15C	0x57	-	This register intentionally blank.
MEM88_CTL	0x160	0x58	-	This register intentionally blank.
MEM89_CTL	0x164	0x59	-	This register intentionally blank.
MEM90_CTL	0x168	0x5A	-	This register intentionally blank.
MEM91_CTL	0x16C	0x5B	-	This register intentionally blank.
MEM92_CTL	0x170	0x5C	-	This register intentionally blank.
MEM93_CTL	0x174	0x5D	-	This register intentionally blank.
MEM94_CTL	0x178	0x5E	-	This register intentionally blank.
MEM95_CTL	0x17C	0x5F	-	This register intentionally blank.
MEM96_CTL	0x180	0x60	-	This register intentionally blank.
MEM97_CTL	0x184	0x61	-	This register intentionally blank.
MEM98_CTL	0x188	0x62	RW	USER_DEF_REG_0
MEM99_CTL	0x18C	0x63	RW	USER_DEF_REG_1
MEM100_CTL	0x190	0x64	RW RW RW RW	ENABLE_QUICK_SREFRESH DRIVE_DQ_DQS BIG_ENDIAN_EN ACTIVE_AGING
MEM101_CTL	0x194	0x65	RW RW RW RW	SWAP_EN RD2RD_TURN PWRUP_SREFRESH_EXIT EN_LOWPOWER_MODE
MEM102_CTL	0x198	0x66	RW RW RW RW	LOWPOWER_AUTO_ENABLE CKE_DELAY LOWPOWER_REFRESH_ENABLE TREF_ENABLE
MEM103_CTL	0x19C	0x67	RW RW	EMRS1_DATA LOWPOWER_CONTROL

Table 77. Registers overview (continued)

Register name	Offset	Mem. CTRL core Reg. Address	Type(1)	Parameter(s)
MEM104_CTL	0x1A0	0x68	RW RW	EMRS2_DATA_1 EMRS2_DATA_0
MEM105_CTL	0x1A4	0x69	RW RW	LOWPOWER_INTERNAL_CNT LOWPOWER_EXTERNAL_CNT
MEM106_CTL	0x1A8	0x6A	RW RW	LOWPOWER_REFRESH_HOLD LOWPOWER_POWER_DOWN_CNT
MEM107_CTL	0x1AC	0x6B	RW RW	TCPD LOWPOWER_SELF_REFRESH_CNT
MEM108_CTL	0x1B0	0x6C	RW	TPDEX

1. Type refers to the writeability of the parameter.

   RW=READ/WRITE.

   RD=Read Only.

   WR=Write Only.

   RW=READ/WRITE, where one or more bits of the parameter have additional functionality and require special handling.

Note: For a comprehensive explanation of the meaning of parameters, please refer to Section 10.14

10.13.4 Register description

10.13.5 MEM0_CTL register

Table 78. MEM0_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB2_FIFO_TYPE	0x0	0x0-0x3	Clock domain correlation between port 2 and Memory Controller core.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB1_FIFO_TYPE	0x0	0x0-0x3	Clock domain correlation between port 1 and Memory Controller core.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB0_FIFO_TYPE	0x0	0x0-0x3	Clock domain correlation between port 0 and Memory Controller core.

Table 78. MEM0_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	ADDR_CMP_EN	0x0	0x0-0x1	Enable address collision detection for CMD queue placement logic.

10.13.6 MEM1_CTL register

Table 79. MEM1_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:08]	AHB4_FIFO_TYPE	0x0	0x0-0x3	Clock domain correlation between port 4 and Memory Controller core.
[07:02]	-	-	-	Reserved. Read undefined. Write should be zero.
[01:00]	AHB3_FIFO_TYPE	0x0	0x0-0x1	Clock domain correlation between port 3 and Memory Controller core.

10.13.7 MEM2_CTL register

Table 80. MEM2_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	BANK_SPLI_EN	0x0	0x0-0x1	Enable bank splitting for CMD queue placement logic.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	AUTO_RFSH_MODE	0x0	0x0-0x1	Define autorefresh to occur at next burst or next CMD boundary.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	AREFRESH	0x0	0x0-0x1	Trigger autorefresh at boundary specified by AUTO_RFSH_MODE. WRITE-ONLY
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	AP	0x0	0x0-0x1	Enable auto pre-charge mode of controller.

10.13.8 MEM3_CTL register

Table 81. MEM3_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	DLL_BYPASS_MODE	0x0	0x0 - 0x1	Enable the DLL bypass feature of the controller.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	DLLLOCK	0x0	0x0 - 0x1	Status of DLL lock coming out of master delay. READ-ONLY
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	DDRII_DDRI_MODE	0x0	0x0 - 0x1	Define mode of controller as DDRI(mobile) or DDRII.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	CONCURRENTAP	0x0	0x0 - 0x1	Allow controller to issue CMDs to other banks while a bank is in auto precharge.

10.13.9 MEM4_CTL register

Table 82. MEM4_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	INTRPTAPBURST	0x0	0x0 - 0x1	Allow the controller to interrupt an auto pre-charge CMD with another CMD.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	FAST_WRITE	0x0	0x0 - 0x1	Define when write CMDs are issued to DRAM devices.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	EIGHT_BANK_MODE	0x0	0x0 - 0x1	Number of banks on the DRAM(s).
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	DQS_N_EN	0x0	0x0 - 0x1	Set DQS pin as single-ended or differential.

10.13.10 MEM5_CTL register

Table 83. MEM5_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	ODT_ADD_TURN_CLK_EN	0x0	0x0 - 0x1	Enable extra turn-around clock between back-to-back READs/WRITEs to different chip selects.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	NOCMDINIT	0x0	0x0 - 0x1	Disable DRAM CMDs until TDLL has expired during initialization.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	INTRPTWRITEA	0x0	0x0 - 0x1	Allow the controller to interrupt a combined write CMD with auto pre-charge with another write CMD.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	INTRPTREADA	0x0	0x0 - 0x1	Allow the controller to interrupt a combined read with auto precharge CMD with another read CMD.

10.13.11 MEM6_CTL register

Table 84. MEM6_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	REDUC	0x0	0x0 - 0x1	Enable the half datapath feature of the controller.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	PRIORITY_EN	0x0	0x0 - 0x1	Enable priority for CMD queue placement logic.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	POWER_DOWN	0x0	0x0 - 0x1	Disable clock enable and set DRAMs in power-down state.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	PLACEMENT_EN	0x0	0x0 - 0x1	Enable placement logic for CMD queue.

10.13.12 MEM7_CTL register

Table 85. MEM7_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	START	0x0	0x0 - 0x1	Begin CMD processing in the controller.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	SREFRESH	0x0	0x0 - 0x1	Place DRAMs in self-refresh mode.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	SW_SAME_EN	0x0	0x0 - 0x1	Enable read/ write grouping for CMD queue placement logic.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	REG_DIMM_EN	0x0	0x0 - 0x1	Enable registered DIMM operation of the controller.

10.13.13 MEM8_CTL register

Table 86. MEM8_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	WRITE_MODE_REG	0x0	0x0 - 0x1	Write EMRS data to the DRAMs. WRITE-ONLY
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	WRITE_INTRPT	0x0	0x0 - 0x1	Allow controller to interrupt write bursts to the DRAMs with a read CMD.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	WEIGHTED_ROUND_ROBIN_LATECNCY_CONTROL	0x0	0x0 - 0x1	Free-running or limited WRR latency counters.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	TRAS_LOCKOUT	0x0	0x0 - 0x1	Allow the controller to execute auto pre-charge CMDs before TRAS_MIN expires.

10.13.14 MEM9_CTL register

Table 87. MEM9_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:24]	ODT_RD_MAP_CS1	0x0	0x0 - 0x3	ODT Chip Select 1 map for READs. Determines which chip(s) will have termination when a read occurs on chip 1.
[23:18]	-	-	-	Reserved. Read undefined. Write should be zero.
[17:16]	ODT_RD_MAP_CS0	0x0	0x0 - 0x3	ODT Chip Select 0 map for READs. Determines which chip(s) will have termination when a read occurs on chip 0.
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:80]	MAX_CS	0x2	0x0 - 0x2	Maximum number of chip selects available. READ-ONLY
[07:02]	-	-	-	Reserved. Read undefined. Write should be zero.
[01:00]	CS_MAP	0x0	0x0 - 0x3	Specify which chip selects are active.

10.13.15 MEM10_CTL register

Table 88. MEM10_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:24]	RTT_0	0x0	0x0 - 0x3	On-Die termination resistance setting for all DRAM devices.
[23:18]	-	-	-	Reserved. Read undefined. Write should be zero.
[17:16]	OUTOFRANGETYPE	0x0	0x0 - 0x3	Type of CMD that caused an Out-of-Range interrupt. READ-ONLY
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:08]	ODT_WR_MAP_CS1	0x0	0x0 - 0x3	ODT Chip Select 1 map for WRITEs. Determines which chip(s) will have termination when a write occurs on chip 1.

Table 88. MEM10_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[07:02]	-	-	-	Reserved. Read undefined. Write should be zero.
[01:00]	ODT_WR_MAP_CS0	0x0	0x0 - 0x3	ODT Chip Select 0 map for WRITEs. Determines which chip(s) will have termination when a write occurs on chip 0.

10.13.16 MEM11_CTL register

Table 89. MEM11_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:27]	-	-	-	Reserved. Read undefined. Write should be zero.
[26:24]	AHB0_R_PRIOTRITY	0x0	0x0 - 0x7	Priority of read CMDs from port 0.
[23:19]	-	-	-	Reserved. Read undefined. Write should be zero.
[18:16]	AHB0_PORT_ORDERING	0x0	0x0 - 0x7	Reassigned port order for port 0.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:08]	ADDR_PINS	0x0	0x0 - 0x7	Difference between number of addr pins available and number being used.
[07:02]	-	-	-	Reserved. Read undefined. Write should be zero.
[01:00]	RTT_PAD_TERMINATION	0x0	0x0 - 0x3	Set termination resistance in controller pads

10.13.17 MEM12_CTL register

Table 90. MEM12_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:27]	-	-	-	Reserved. Read undefined. Write should be zero.
[26:24]	AHB1_W_PRIORITY	0x0	0x0 - 0x7	Priority of write commands from port 1.
[23:19]	-	-	-	Reserved. Read undefined. Write should be zero.
[18:16]	AHB1_R_PRIORITY	0x0	0x0 - 0x7	Priority of read commands from port 1.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.

Table 90. MEM12_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[10:08]	AHB1_PORT_ORDERING	0x0	0x0 - 0x7	Reassigned port order for port 1.
[07:03]	-	-	-	Reserved. Read undefined. Write should be zero.
[02:00]	AHB0_W_PRIORITY	0x0	0x0 - 0x7	Priority of write commands from port 0.

10.13.18 MEM13_CTL register

Table 91. MEM13_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:27]	-	-	-	Reserved. Read undefined. Write should be zero.
[26:24]	AHB3_PORT_ORDERING	0x0	0x0 - 0x7	Reassigned port order for port 3.
[23:19]	-	-	-	Reserved. Read undefined. Write should be zero.
[18:16]	AHB2_W_PRIORITY	0x0	0x0 - 0x7	Priority of write commands from port 2.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:08]	AHB2_R_PRIORITY	0x0	0x0 - 0x7	Priority of read commands from port 2.
[07:03]	-	-	-	Reserved. Read undefined. Write should be zero.
[02:00]	AHB2_PORT_ORDERING	0x0	0x0 - 0x7	Reassigned port order for port 2.

10.13.19 MEM14_CTL register

Table 92. MEM14_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:27]	-	-	-	Reserved. Read undefined. Write should be zero.
[26:24]	AHB4_R_PRIORITY	0x0	0x0 - 0x7	Priority of read commands from port 4.
[23:19]	-	-	-	Reserved. Read undefined. Write should be zero.
[18:16]	AHB4_PORT_ORDERING	0x0	0x0 - 0x7	Reassigned port order for port 4.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:08]	AHB3_W_PRIORITY	0x0	0x0 - 0x7	Priority of write commands from port 3.

Table 92. MEM14_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[07:03]	-	-	-	Reserved. Read undefined. Write should be zero.
[02:00]	AHB3_R_PRIORITY	0x0	0x0 - 0x7	Priority of read commands from port 3.

10.13.20 MEM15_CTL register

Table 93. MEM15_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:03]	-	-	-	Reserved. Read undefined. Write should be zero.
[02:00]	AHB4_W_PRIORITY	0x0	0x0 - 0x7	Priority of write commands from port 4.

10.13.21 MEM16_CTL register

Table 94. MEM16_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.22 MEM17_CTL register

Table 95. MEM17_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:27]	-	-	-	Reserved. Read undefined. Write should be zero.
[26:24]	TCKE	0x0	0x0 - 0x7	Minimum CKE pulse width.
[23:19]	-	-	-	Reserved. Read undefined. Write should be zero.
[18:16]	OUT_OF_RANGE_SOURCE_ID	0x0	0x0 - 0x7	Source ID of CMD that caused an Out-of-Range interrupt. READ-ONLY
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:08]	COLUMN_SIZE	0x0	0x0 - 0x7	Difference between number of column pins available and number being used.

Table 95. MEM17_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[07:03]	-	-	-	Reserved. Read undefined. Write should be zero.
[02:00]	CAS_LATENCY	0x0	0x0 - 0x7	Encoded CAS latency sent to DRAMs during initialization.

10.13.23 MEM18_CTL register

Table 96. MEM18_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:27]	-	-	-	Reserved. Read undefined. Write should be zero.
[26:24]	TRTP	0x0	0x0 - 0x7	DRAM TRTP parameter in cycles.
[23:19]	-	-	-	Reserved. Read undefined. Write should be zero.
[18:16]	TRRD	0x0	0x0 - 0x7	DRAM TRRD parameter in cycles.
[15:03]	-	-	-	Reserved. Read undefined. Write should be zero.
[02:00]	TEMRS	0x0	0x0 - 0x7	DRAM TEMRS parameter in cycles.

10.13.24 MEM19_CTL register

Table 97. MEM19_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:27]	-	-	-	Reserved. Read undefined. Write should be zero.
[26:24]	WRLAT	0x0	0x0 - 0x7	DRAM WRLAT parameter in cycles.
[23:18]	-	-	-	Reserved. Read undefined. Write should be zero.
[17:16]	WEIGHTED_ROUND_ROBIN_WEIGHT_SHARING	0x0	0x0 - 0x3	Per-port pair shared arbitration for WRR.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:08]	TWTR	0x0	0x0 - 0x7	DRAM TWTR parameter in cycles.
[07:03]	-	-	-	Reserved. Read undefined. Write should be zero.
[02:00]	TWR_INT	0x0	0x0 - 0x7	DRAM TWR parameter in cycles.

10.13.25 MEM20_CTL register

Table 98. MEM20_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB0_PRIORITY2_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 2 CMDs from port 0.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB0_PRIORITY1_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 1 CMDs from port 0.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB0_PRIORITY0_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 0 CMDs from port 0.
[07:06]	-	-	-	Reserved. Read undefined. Write should be zero.
[05:00]	AGE_COUNT	0x0	0x0 - 0x3F	Initial value of master generate counter for CMD aging.

10.13.26 MEM21_CTL register

Table 99. MEM21_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB0_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 6 CMDs from port 0.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB0_PRIORITY5_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 5 CMDs from port 0.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB0_PRIORITY4_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 4 CMDs from port 0.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB0_PRIORITY3_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 3 CMDs from port 0.

10.13.27 MEM22_CTL register

Table 100. MEM22_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB1_PRIORITY2_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 2 CMDs from port 1.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB1_PRIORITY1_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 1 CMDs from port 1.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB1_PRIORITY0_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 0 CMDs from port 1.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB0_PRIORITY7_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 7 CMDs from port 0.

10.13.28 MEM23_CTL register

Table 101. MEM23_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB1_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 6 CMDs from port 1.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB1_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 5 CMDs from port 1.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB1_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 4 CMDs from port 1.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB1_PRIORITY3_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 3 CMDs from port 1.

10.13.29 MEM24_CTL register

Table 102. MEM24_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB2_PRIORITY2_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 2 CMDs from port 2.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB2_PRIORITY1_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 1 CMDs from port 2.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB2_PRIORITY0_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 0 CMDs from port 2.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB1_PRIORITY7_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 7 CMDs from port 1.

10.13.30 MEM25_CTL register

Table 103. MEM25_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB2_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 6 CMDs from port 2.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB2_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 5 CMDs from port 2.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB2_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 4 CMDs from port 2.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB2_PRIORITY3_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 3 CMDs from port 2.

10.13.31 MEM26_CTL register

Table 104. MEM26_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB3_PRIORITY2_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 2 CMDs from port 3.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB3_PRIORITY1_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 1 CMDs from port 3.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB3_PRIORITY0_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 0 CMDs from port 3.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB2_PRIORITY7_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 7 CMDs from port 2.

10.13.32 MEM27_CTL register

Table 105. MEM27_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB3_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 6 CMDs from port 3.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB3_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 5 CMDs from port 3.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB3_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 4 CMDs from port 3.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB3_PRIORITY3_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 3 CMDs from port 3.

10.13.33 MEM28_CTL register

Table 106. MEM28_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB4_PRIORITY2_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 2 CMDs from port 4.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB4_PRIORITY1_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 1 CMDs from port 4.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB4_PRIORITY0_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 0 CMDs from port 4.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB3_PRIORITY7_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 7 CMDs from port 3.

10.13.34 MEM29_CTL register

Table 107. MEM29_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	AHB4_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 6 CMDs from port 4.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	AHB4_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 5 CMDs from port 4.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	AHB4_PRIORITY6_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 4 CMDs from port 4.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB4_PRIORITY3_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 3 CMDs from port 4.

10.13.35 MEM30_CTL register

Table 108. MEM30_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	AHB4_PRIORITY7_RELATIVE_PRIORITY	0x0	0x0 - 0xF	Relative priority of priority 7 CMDs from port 4.

10.13.36 MEM31_CTL/MEM32_CTL/MEM33_CTL register

Table 109. MEM31_CTL/MEM32_CTL/MEM33_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.37 MEM34_CTL register

Table 110. MEM34_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	CASLAT_LIN_GATE	0x0	0x0 - 0xF	Adjusts data capture gate open by half cycles.
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	CASLAT_LIN	0x0	0x0 - 0xF	Sets latency from read CMD send to data receive from/to controller.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	APREBIT	0x0	0x0 - 0xF	Location of the auto pre-charge bit in the DRAM address.
[07:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.38 MEM35_CTL register

Table 111. MEM35_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	MAX_ROW	0xF	0x0 - 0xF	Maximum width of memory addresses bus. READ-ONLY
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	MAX_COL	0xE	0x0 - 0xE	Maximum width of column address in DRAMs. READ-ONLY
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	INITAREF	0x0	0x0 - 0xF	Number of auto-refresh CMDs to execute during DRAM initialization.
[07:06]	-	-	-	Reserved. Read undefined. Write should be zero.
[05:00]	COMMAND_AGE_COUNT	0x00	0x0 - 0x3F	Initial value of individual CMD aging counters for CMD aging.

10.13.39 MEM36_CTL register

Table 112. MEM36_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:28]	-	-	-	Reserved. Read undefined. Write should be zero.
[27:24]	WRR_PARAM_VALUE_ERR	0x0	0x0 - 0xF	Errors/warnings related to the WRR parameters. READ-ONLY
[23:20]	-	-	-	Reserved. Read undefined. Write should be zero.
[19:16]	TRP	0x0	0x0 - 0xF	DRAM TRP parameter in cycles.
[15:12]	-	-	-	Reserved. Read undefined. Write should be zero.
[11:08]	TDAL	0x0	0x0 - 0xF	DRAM TDAL parameter in cycles.
[07:04]	-	-	-	Reserved. Read undefined. Write should be zero.
[03:00]	Q_FULLNESS	0x0	0x0 - 0xF	Quantity that determines CMD queue full.

10.13.40 MEM37_CTL register

Table 113. MEM37_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:29]	-	-	-	Reserved. Read undefined. Write should be zero.
[28:24]	TFAW	0x0	0x0 - 0x1F	DRAM TFAW parameter in cycles.

Table 113. MEM37_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[23:21]	-	-	-	Reserved. Read undefined. Write should be zero.
[20:16]	OCD_ADJUST_PUP_CS0	0x0	0x0 - 0x1F	OCD pull-up adjust setting for DRAMs for chip select 0.
[15:13]	-	-	-	Reserved. Read undefined. Write should be zero.
[12:08]	OCD_ADJUST_PDN_CS0	0x0	0x0 - 0x1F	OCD pull-down adjust setting for DRAMs for chip select 0.
[07:06]	-	-	-	Reserved. Read undefined. Write should be zero.
[05:00]	INT_ACK	0x0	0x0 - 0x3F	Clear mask of the INT_STATUS parameter. WRITE-ONLY

10.13.41 MEM38_CTL register

Table 114. MEM38_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31]	-	-	-	Reserved. Read undefined. Write should be zero.
[30:24]	INT_STATUS	0x0	0x0 - 0x7F	Status of interrupt features in the controller. READ-ONLY
[23]	-	-	-	Reserved. Read undefined. Write should be zero.
[22:16]	INT_MASK	0x0	0x0 - 0x7F	Mask for controller_int signals from the INT_STATUS parameter.
[15:13]	-	-	-	Reserved. Read undefined. Write should be zero.
[12:08]	TRC	0x0	0x0 - 0x1F	DRAM TRC parameter in cycles.
[07:05]	-	-	-	Reserved. Read undefined. Write should be zero.
[04:00]	TMRD	0x0	0x0 - 0x1F	DRAM TMRD parameter in cycles.

10.13.42 MEM39_CTL register

Table 115. MEM39_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:15]	-	-	-	Reserved. Read undefined. Write should be zero.
[14:08]	DLL_DQS_DELAY1	0x0	0x0 - 0x7F	Fraction of a cycle to delay the dqs signal from the DRAMs for dll_rd_dqs_slice 1 during READs.

Table 115. MEM39_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[07]	-	-	-	Reserved. Read undefined. Write should be zero.
[06:00]	DLL_DQS_DELA 0	0x0	0x0 - 0x7F	Fraction of a cycle to delay the dqs signal from the DRAMs for dll_rd_dqs_slice 0 during READs.

10.13.43 MEM40_CTL register

Table 116. MEM40_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31]	-	-	-	Reserved. Read undefined. Write should be zero.
[30:24]	DQS_OUT_ SHIFT	0x0	0x0 - 0x7F	Fraction of a cycle to delay the write dqs signal to the DRAMs during WRITEs.
[23:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.44 MEM41_CTL register

Table 117. MEM41_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:23]	-	-	-	Reserved. Read undefined. Write should be zero.
[22:16]	WR_DQS_SHIFT	0x0	0x0 - 0x7F	Fraction of a cycle to delay the ddr_close signal in the controller.
[15:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.45 MEM42_CTL register

Table 118. MEM42_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:24]	TRFC	0x0	0x0 - 0xFF	DRAM TRFC parameter in cycles.
[23:16]	TRCD_INT	0x0	0x0 - 0xFF	DRAM TRCD parameter in cycles.
[15:08]	TRAS_MIN	0x0	0x0 - 0xFF	DRAM TRAS_MIN parameter in cycles.
[07:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.46 MEM43_CTL register

Table 119. MEM43_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	AHB1_PRIORITY_RELAX	0x000	0x000 - 0x3FF	Counter value to trigger priority relax on port 1.
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:00]	AHB0_PRIORITY_RELAX	0x000	0x000 - 0x3FF	Counter value to trigger priority relax on port 0.

10.13.47 MEM44_CTL register

Table 120. MEM44_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	AHB3_PRIORITY_RELAX	0x000	0x000 - 0x3FF	Counter value to trigger priority relax on port 3.
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:00]	AHB2_PRIORITY_RELAX	0x000	0x000 - 0x3FF	Counter value to trigger priority relax on port 2.

10.13.48 MEM45_CTL register

Table 121. MEM45_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:00]	AHB4_PRIORITY_RELAX	0x000	0x000 - 0x3FF	Counter value to trigger priority relax on port 4.

10.13.49 MEM46_CTL register

Table 122. MEM46_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	OUT_OF_RANGE _LENGTH	0x000	0x000 - 0x3FF	Length of CMD that caused an Out-of-Range interrupt. READ-ONLY
[15:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.50 MEM47_CTL register

Table 123. MEM47_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	AHB0_WRCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR WRITE CMD on port 0.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:00]	AHB0_RDCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR READ CMD on port 0.

10.13.51 MEM48_CTL register

Table 124. MEM48_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	AHB1_WRCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR WRITE CMD on port 1.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:00]	AHB1_RDCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR READ CMD on port 1.

10.13.52 MEM49_CTL register

Table 125. MEM49_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	AHB2_WRCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR WRITE CMD on port 2.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:00]	AHB2_RDCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR READ CMD on port 2.

10.13.53 MEM50_CTL register

Table 126. MEM50_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	AHB3_WRCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR WRITE CMD on port 3.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:00]	AHB3_RDCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR READ CMD on port 3.

10.13.54 MEM51_CTL register

Table 127. MEM51_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	AHB4_WRCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR WRITE CMD on port 4.
[15:11]	-	-	-	Reserved. Read undefined. Write should be zero.
[10:00]	AHB4_RDCNT	0x000	0x000 - 0x7FF	Number of bytes for an INCR READ CMD on port 4.

10.13.55 MEM52_CTL/MEM53_CTL register

Table 128. MEM52_CTL/MEM53_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.56 MEM54_CTL register

Table 129. MEM54_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:14]	-	-	-	Reserved. Read undefined. Write should be zero.
[13:00]	TREF	0x0000	0x0000 - 0x3FFF	DRAM TREF parameter in cycles.

10.13.57 MEM55_CTL register

Table 130. MEM55_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:15]	-	-	-	Reserved. Read undefined. Write should be zero.
[14:00]	EMRS3_DATA	0x0000	0x0000 - 0x7FFF	EMRS3 data.

10.13.58 MEM56_CTL register

Table 131. MEM56_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:16]	TRAS_MAX	0x0000	0x0000 - 0xFFFF	DRAM TRAS_MAX parameter in cycles.
[15:00]	TDLL	0x0000	0x0000 - 0xFFFF	DRAM TDLL parameter in cycles.

10.13.59 MEM57_CTL register

Table 132. MEM57_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:16]	TXSR	0x0000	0x0000 - 0xFFFF	DRAM TXSR parameter in cycles.
[15:00]	TXSNR	0x0000	0x0000 - 0xFFFF	DRAM TXSNR parameter in cycles.

10.13.60 MEM58_CTL register

Table 133. MEM58_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:16]	-	-	-	Reserved. Read undefined. Write should be zero.
[15:00]	VERSION	0x2041	-	Controller version number. READ-ONLY

10.13.61 MEM59_CTL register

Table 134. MEM59_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:24]	-	-	-	Reserved. Read undefined. Write should be zero.
[23:00]	TINIT	0x000000	0x0 - 0xFFFFFF	DRAM TINIT parameter in cycles.

10.13.62 MEM60_CTL register

Table 135. MEM60_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:00]	out_rng_addr	0x0000.0000	0x0 - 0xFFFF.FFFF	Lower portion of address of CMD that caused an Out-of-Range interrupt. READ-ONLY

10.13.63 MEM61_CTL register

Table 136. MEM61_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:02]	-	-	-	Reserved. Read undefined. Write should be zero.
[01:00]	out_rng_addr	0x0	0x0 - 0x3	Upper portion of address of CMD that caused an Out-of-Range interrupt. READ-ONLY

10.13.64 MEM62_CTL/MEM63_CTL/MEM64_CTL register

Table 137. MEM62_CTL/MEM63_CTL/MEM64_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.65 MEM65_CTL register

Table 138. MEM65_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	dll_dqs_dly_byps0	0x0	0x1 - 0x3FF	Number of delay elements to include in the dqs signal from the DRAMs for dll_rd_dqs_slice 0 during READs when DLL is being bypassed.
[15:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.66 MEM66_CTL register

Table 139. MEM66_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	dll_increment	0x0	0x1 - 0x3FF	Number of elements to add to DLL_START_POINT when searching for lock.
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:00]	dll_dqs_dly_byps1	0x0	0x1 - 0x3FF	Number of delay elements to include in the dqs signal from the DRAMs for dll_rd_dqs_slice 1during READs when DLL is being bypassed.

10.13.67 MEM67_CTL register

Table 140. MEM67_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	dll_start_point	0x0	0x1 - 0x3FF	Initial delay count when searching for lock in master DLL.

Table 140. MEM67_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:00]	dll_lock	0x0	0x1 - 0x3FF	Number of delay elements in master DLL lock. READ ONLY

10.13.68 MEM68_CTL register

Table 141. MEM68_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:26]	-	-	-	Reserved. Read undefined. Write should be zero.
[25:16]	wr_dqs_shft_byps	0x0	0x1 - 0x3FF	Number of delay elements to include in the ddr_close signal in the controller when the DLL is being bypassed.
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:00]	dqs_out_shft_byps	0x0	0x1 - 0x3FF	Number of delay elements to include in the write dqs signal to the DRAMs during WRITEs when the DLL is being bypassed.

10.13.69 MEM[69-97]_CTL register

Table 142. MEM[69-97]_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:00]	-	-	-	Reserved. Read undefined. Write should be zero.

10.13.70 MEM[98-99]_CTL register

Table 143. MEM[98-99]_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:00]	user_def_reg(x)	0x0	0x0 - 0xFFFF_FFFF	User defined register.

Note: Only the USER_DEF_REG(0) bit 0 is used in SPEAr300. All the other bits are reserved.

10.13.71 MEM100_CTL register

Table 144. MEM100_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	enable_quick_sref resh	0x0	0x0 - 0x1	Allow user to interrupt memory initialization to enter self refresh mode.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	drive_dq_dqs	0x0	0x0 - 0x1	Sets DQ/DQS output enable behavior when controller is idle.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	big_endian_enabl e	0x0	0x0 - 0x1	Set byte ordering as little endian or big endian.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	active_aging	0x0	0x0 - 0x1	Enable command aging in the command queue.

10.13.72 MEM101_CTL register

Table 145. MEM101_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:25]	-	-	-	Reserved. Read undefined. Write should be zero.
[24]	swap_enable	0x0	0x0 - 0x1	Enable command swapping logic in execution unit.
[23:17]	-	-	-	Reserved. Read undefined. Write should be zero.
[16]	rd2rd_turn	0x0	0x0 - 0x1	Enable insertion of addition turn around clock for back to back READs to different css.
[15:09]	-	-	-	Reserved. Read undefined. Write should be zero.
[08]	pwrup_srefresh_exit	0x0	0x0 - 0x1	Allow powerup via self-refresh instead of full memory initialization.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	en_lowpower_mode	0x0	0x0 - 0x1	Enable low power mode in controller.

10.13.73 MEM102_CTL register

Table 146. MEM102_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:29]	-	-	-	Reserved. Read undefined. Write should be zero.
[28:24]	lowpower_auto_enable	0x0	0x0 - 0x1F	Enables automatic entry into the low power mode on idle.
[23:19]	-	-	-	Reserved. Read undefined. Write should be zero.
[18:16]	cke_delay	0x0	0x0 - 0x7	Additional cycles to delay CKE for status reporting.
[15:10]	-	-	-	Reserved. Read undefined. Write should be zero.
[09:08]	lowpower_refresh_enable	0x0	0x0 - 0x3	Enable refreshes during power down.
[07:01]	-	-	-	Reserved. Read undefined. Write should be zero.
[00]	tref_enable	0x0	0x0 - 0x1	Issue self refresh CMDs to the DRAMs every TREF cycles.

10.13.74 MEM103_CTL register

Table 147. MEM103_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:23]	-	-	-	Reserved. Read undefined. Write should be zero.
[22:08]	emrs1_data	0x0	0x0 - 0x7FFF	EMRS1 data.
[07:05]	-	-	-	Reserved. Read undefined. Write should be zero.
[04:00]	lowpower_control	0x0	0x0 - 0x1F	Controls entry into the low power modes.

10.13.75 MEM104_CTL register

Table 148. MEM104_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:15]	-	-	-	Reserved. Read undefined. Write should be zero.
[30;16]	emrs2_data1	0x0000	0x0000 - 0x7FFF	EMRS2 data for chips select 1.

Table 148. MEM104_CTL register bit assignments (continued)

Bit	Name	Reset value	Range	Description
[15]	-	-	-	Reserved. Read undefined. Write should be zero.
[14:00]	emrs2_data0	0x0000	0x0000 - 0x7FFF	EMRS2 data for chip select 0.

10.13.76 MEM105_CTL register

Table 149. MEM105_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:16]	lowpower_int_cnt	0x0	0x0 - 0xFFFF	Counts idle cycles to self refresh with memory and controller clk gating.
[15:00]	lowpower_ext_cnt	0x0	0x0 - 0xFFFF	Counts idle cycles to self refresh with memory clock gating.

10.13.77 MEM106_CTL register

Table 150. MEM106_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:16]	lowpower_rfsh_hold	0x0	0x0 - 0xFFFF	Re-Sync counter for DLL in Clock Gate Mode.
[15:00]	lowpower_pwdwn_cnt	0x0	0x0 - 0xFFFF	Counts idle cycles to memory powerdown.

10.13.78 MEM107_CTL register

Table 151. MEM107_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:16]	tcpd	0x0	0x0 - 0xFFFF	DRAM TCPD parameter in cycles.
[15:00]	lowpower_srfsh_cnt	0x0	0x0 - 0xFFFF	Counts idle cycles to memory self refresh.

10.13.79 MEM108_CTL register

Table 152. MEM108_CTL register bit assignments

Bit	Name	Reset value	Range	Description
[31:16]	Reserved	-	-	Reserved. Read undefined. Write should be zero.
[15:00]	TPDEX	0x0	0x0 - 0xFFFF	DRAM TPDEX parameter in cycles.

10.14 Summary of memory controller parameters

Note: The table below gives a description of the parameters referred to throughout this chapter. To fully understand the concepts related to each parameter, please refer to the relevant sections of the document.

Table 153. Memory controller parameters

Parameter	Description
AGING[0]	Enables aging of commands in the command queue when using the placement logic to fill the command queue. The total number of cycles required to decrement the priority value on a command by one is the product of the values in the age_count and command_age_count parameters. 1'b0 - Disabled 1'b1 - Enabled
addr_cmp_en [0]	Enables address collision/data coherency detection as a condition when using the placement logic to fill the command queue. 1'b0 - Disabled 1'b1 - Enabled
addr_pins [2:0]	Defines the difference between the maximum number of address pins configured (15) and the actual number of pins being used. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this parameter. For details, please refer to *Section 10.9 on page 142*.
age_count [5:0]	Holds the initial value of the master aging-rate counter. When using the placement logic to fill the command queue, the command aging counters will be decremented one each time the master aging-rate counter counts down age_count cycles.
ahbX_fifo_type_reg [1:0]	Sets the correlation of the clock domains between AHB port X and the Memory Controller core clock. 2'b00 - Asynchronous 2'b01 - 2:1 Reserved 2'b10 - 1:2 Port: Core Pseudo-Synchronous 2'b11 - Synchronous
ahbX_port_ordering [2:0]	Used in weighted round-robin arbitration to modify the order than the ports are scanned when multiple commands are at the same priority level and have the same relative priorities.
ahbX_priority_relax [9:0]	Holds the counter value for AHB port X at which the priority relax condition is triggered in weighted round robin arbitration.
ahbX_priorityY_relative_priority [3:0]	Holds the relative priority of AHB port X for priority Y commands in weighted round robin arbitration.
ahbX_r_priority [2:0]	Sets the priority of READ commands from AHB port X. A value of 0 is the highest priority.

Table 153. Memory controller parameters (continued)

Parameter	Description
ahbX_rdcnt [10:0]	Holds the number of bytes to be responded to AHB port X after an INCR READ AHB command. The AHB logic will subdivide an INCR request into Memory Controller core commands of the size of this parameter. The logic will continue requesting bursts of this size as soon as the previous request has been received by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this parameter should be a multiple of the number of bytes in the AHB port width. Clearing this parameter will cause the port to issue commands of 0 length to the Memory Controller core, which the core interprets as the pre-configured value of 1024 bytes.
ahbX_w_priority [2:0]	Sets the priority of WRITE commands from AHB port X. A value of 0 is the highest priority.
ahbX_wrcnt [10:0]	Holds the number of bytes to send to the Memory Controller core from AHB port X for an INCR WRITE AHB command. The AHB logic will subdivide an INCR request into Memory Controller core commands of the size of this parameter. The logic will continue sending bursts of this size as the previous request has been transmitted by the AHB port. If the INCR command is terminated on an unnatural boundary, the logic will discard the unnecessary words. The value defined in this parameter should be a multiple of the number of bytes in the AHB port width. Clearing this parameter will cause the port to issue commands of 0 length to the Memory Controller core, which the core interprets as the pre-configured value of 1024 bytes.
ap [0]	Enables auto pre-charge mode for DRAM devices.(1) 'b0 - Auto pre-charge mode disabled. Memory banks will stay open until another request requires this bank, the maximum open time (tras_max) has elapsed, or a refresh command closes all the banks. 'b1 - Auto pre-charge mode enabled. All READ and WRITE transactions must be terminated by an auto pre-charge command. If a transaction consists of multiple READ or write bursts, only the last command is issued with an auto pre-charge.
aprebit [3:0]	Defines the location of the auto pre-charge bit in the DRAM address in decimal encoding.(1)
arefresh [0]	Begins an automatic refresh to the DRAM devices based on the setting of the auto_refresh_mode parameter. If there are any open banks when this parameter is set, the Memory Controller will automatically close these banks before issuing the auto-refresh command. This parameter will always read back 'b0. 1'b0 - No action 1'b1 - Issue refresh to the DRAM devices

Table 153. Memory controller parameters (continued)

Parameter	Description
auto_refresh_mode [0]	Sets the mode to be performed as the automatic refresh will occur. If auto_refresh_mode is set and a refresh is required to memory, the Memory Controller will either delay this refresh until the end of the current transaction has been reached (if the transaction is fully contained inside a single page), or until the current transaction hits the end of the current page. 1'b0 - Issue refresh on the next DRAM burst boundary, even if the current command is not complete. 1'b1 - Issue refresh on the next command boundary.
bank_split_en [0]	Enables bank splitting as a condition when using the placement logic to fill the command queue. 1'b0 - Disabled 1'b1 - Enabled
big_endian_en [0]	Selects the byte ordering for Memory Controllers with programmable endian setting. 1'b0 - Little Endian 1'b1 - Big Endian
caslat [2:0]	Sets the CAS latency encoding that the memory uses. The binary value of this parameter is dependent on the memory device, since the same caslat value may have different meanings to different memories. This will be programmed into the DRAM devices at initialization. The CAS encoding will be specified in the DRAM spec sheet, and should correspond to the caslat_lin parameter.
caslat_lin [3:0]	Sets the CAS latency linear value as half-cycles expressed increments number. This sets an internal adjustment for the delay since the READ command is sent from the Memory Controller until data will be received back. The timing window inside which the data is captured is a fixed length. The caslat_lin parameter adjusts the start of this data capture window. Not all linear values are supported by every memory devices: please refer to the specification for the memory devices being actually used. 4'b0000 - 4'b0001 - Reserved 4'b0010 -1 cycle 4'b0011 - 1.5 cycles 4'b0100 - 2 cycles 4'b0101 - 2.5 cycles 4'b0110 - 3 cycles 4'b0111 - 3.5 cycles 4'b1000 - 4 cycles 4'b1001 - 4.5 cycles 4'b1010 - 5 cycles 4'b1011 - 5.5 cycles 4'b1100 - 6 cycles 4'b1101 - 6.5 cycles 4'b1110 - 7 cycles 4'b1111 - 7.5 cycles

Table 153. Memory controller parameters (continued)

Parameter	Description
caslat_lin_gate [3:0]	Adjusts the data capture gate open time by half-cycle expressed increments number. This parameter is set differently than caslat_lin one whether there are fixed offsets in the flight path between the memories and the Memory Controller for clock gating. When caslat_lin_gate is a larger value than caslat_lin, the data capture window will become shorter. A caslat_lin_gate value smaller than caslat_lin may have no effect on the data capture window, depending on the fixed offsets in the device and the board. 4'b0000 - 4'b0001 - Reserved 4'b0010 - 1 cycle 4'b0011 - 1.5 cycles 4'b0100 - 2 cycles 4'b0101 - 2.5 cycles 4'b0110 - 3 cycles 4'b0111 - 3.5 cycles 4'b1000 - 4 cycles 4'b1001 - 4.5 cycles 4'b1010 - 5 cycles 4'b1011 - 5.5 cycles 4'b1100 - 6 cycles 4'b1101 - 6.5 cycles 4'b1110 - 7 cycles 4'b1111 - 7.5 cycles
cke_delay [2:0]	Sets the number of additional cycles of delay to include in the CKE signal cke_status for status reporting. The default delay is 0 cycles.
column_size [2:0]	Shows the difference between the maximum column width available (14) and the actual number of column pins being used. The user address is automatically shifted so that its space is mapped contiguously into the memory map based on the value of this parameter. For details, refer to *Section 10.9 on page 142*.
command_age_count [5:0]	Holds the initial value of the command aging counters associated with each command in the command queue. When using the placement logic to fill the command queue, the command aging counters decrement one each time the master aging-rate counter counts down age_count cycles.
concurrentap [0]	Enables concurrent auto pre-charge. Some DRAM devices do not allow one bank to be auto pre-charged while another bank is reading or writing. The JEDEC standard allows concurrent auto pre-charge. Set this parameter for the DRAM device being used. 1'b0 - Concurrent auto pre-charge disabled. 1'b1 - Concurrent auto pre-charge enabled.

Table 153. Memory controller parameters (continued)

Parameter	Description
cs_map [1:0]	Sets the mask that determines which chip select pins are active, with each bit representing a different chip select. The user address chip select field will be mapped into the active chip selects indicated by this parameter in ascending order from lowest to highest. This allows the Memory Controller to map the entire contiguous user address into any group of chip selects. Bit 0 of this parameter corresponds to chip select [0], bit 1 corresponds to chip select [1], etc. The number of chip selects, i.e. the number of bits set to 1 in this parameter, must be a power of 2.
ddrii_sdram_mode [0]	Selects between the DDR1(Mobile) and DDR2 modes of operation.(2) 1'b0 - DDR1(mobile) mode 1'b1 - DDR2 mode
dll_bypass_mode [0]	Defines the behavior of the DLL bypass logic and establishes which set of delay parameters will be used. When dll_bypass_mode is set to 'b0, the values programmed in the dll_dqs_delay_X, dqs_out_shift, and wr_dqs_shift are used. These parameters add fractional increments of the clock to the specified lines. When dll_bypass_mode is set to 'b1, the values programmed into the dll_dqs_delay_bypass_X, dqs_out_shift_bypass, and wr_dqs_shift_bypass are used. These parameters specify the actual number of delay elements added to each of the lines. If the total delay time programmed into the delay parameters exceeds the number of delay elements in the delay chain, the delay will be set to the maximum number of delay elements in the delay chain. 1'b0 - Normal operational mode. 1'b1 - Bypass the DLL master delay line.
dll_dqs_delay_X [6:0]	Sets the delay for the read_dqs signal from the DDR SDRAM devices for dll_rd_dqs_slice X. This delay is used center the edges of the read_dqs signal so that the READ data will be captured in the middle of the valid window in the I/O logic. Each increment of this parameter adds a delay of 1/128 of the system clock. The same delay will be added to the read_dqs signal for each byte of the READ data. (3)
dll_dqs_delay_bypass_X [9:0]	Sets the delay for the read_dqs signal from the DDR SDRAM devices for dll_rd_dqs_slice X for READs when the DLL is being bypassed. This delay is used to center the edges of the read_dqs signal so that the READ data will be captured in the middle of the valid window in the I/O logic. The value programmed into this parameter sets the actual number of delay elements in the read_dqs line. The same delay will be added to the read_dqs signal for each byte of the READ data. If the total delay time programmed exceeds the number of delay elements in the delay chain, the delay will be set internally to the maximum number of delay elements available.(4)
dll_increment [9:0]	Defines the number of delay elements to recursively increment the dll_start_point parameter with when searching for lock.

Table 153. Memory controller parameters (continued)

Parameter	Description
dll_lock [9:0]	Shows the actual number of delay elements used to capture one full clock cycle. This parameter is automatically updated every time a refresh operation is performed. This parameter is read-only.
dll_start_point [9:0]	Sets the number of delay elements to place in the master delay line to start searching for lock in master DLL.
dlllockreg [0]	DLL lock/unlock. This parameter is read-only.
dqs_n_en [0]	Enables differential data strobe signals from the DRAM. 1'b0 - Single-ended DQS signal from the DRAM. 1'b1 - Differential DQS signal from the DRAM.
dqs_out_shift [6:0]	Sets the delay for the clk_dqs_out signal of the ddr_close to ensure correct data capture in the I/O logic. Each increment of this parameter adds a delay of 1/128 of the system clock. (5)
dqs_out_shift_bypass [9:0]	Sets the delay for the clk_dqs_out signal of the ddr_close when the DLL is being bypassed. This is used to ensure correct data capture in the I/O logic. The value programmed into this parameter sets the actual number of delay elements in the clk_dqs_out line. If the total delay time programmed exceeds the number of delay elements in the delay chain, the delay will be set internally to the maximum number of delay elements available. (6)
drive_dq_dqs [0]	Selects whether the DQ output enables and DQS output enables will be driven active when the Memory Controller is in idle state. 1'b0 - Leave the output enables de-asserted when idle. 1'b1 - Drive the output enables active when idle.
eight_bank_mode [0].	Reports that the memory devices have eight banks. 1'b0 - Memory devices have 4 banks. 1'b1 - Memory devices have 8 banks
emrs1_data [14:0]	Holds the EMRS1 data written during DDRII initialization. The contents of this parameter will be programmed into the DRAM at initialization or when the write_modereg parameter is set to 1'b1. Consult the DRAM specification for the correct settings of this parameter.
emrs2_data_X [14:0]	Holds the EMRS2 data written during DDRII initialization for chip select X. The contents of this parameter will be programmed into the DRAM at initialization or when the write_modereg parameter is set to 1'b1. Consult the DRAM specification for the correct settings for this parameter.
emrs3_data [14:0]	Holds the EMRS3 data written during DDRII initialization. The contents of this parameter will be programmed into the DRAM at initialization or when the write_modereg parameter is set to 1'b1. Consult the DRAM specification for the correct settings for this parameter.
en_lowpower_mode [0]	Enables the mobile mode of the Memory Controller. (7) 1'b0 - Disabled 1'b1 - Enabled

Table 153. Memory controller parameters (continued)

Parameter	Description
enable_quick_srefresh [0]	When this bit is set, the memory initialization sequence will be interrupted and self-refresh mode will be entered. 1'b0 - Continue memory initialization. 1'b1 - Interrupts memory initialization and enter self-refresh mode.
fast_write [0]	Controls the mode and timing the WRITE commands are issued toward the DRAM devices. 1'b0 - The Memory Controller will issue a WRITE command to the DRAM devices as it has received enough data for one DRAM burst. In this mode, WRITE data can be sent in any cycle relative to the WRITE command. This mode also allows multi-word WRITE command data to arrive in non-sequential cycles. 1'b1 - The Memory Controller will issue a WRITE command to the DRAM devices after the first word of the WRITE data is received by the Memory Controller. The first word can be sent at any time relative to the WRITE command. In this mode, multi-word WRITE command data must be available to the Memory Controller in sequential cycles.
initaref [3:0]	Defines the number of auto-refresh commands needed by the DRAM devices to satisfy the initialization sequence.
int_ack [5:0]	Sets the clearing of the int_status parameter. If any of the int_ack bits are set to 'b1 the corresponding bit in the int_status parameter will be set to 'b0. Any int_ack bits set to 1'b0 does not affect the corresponding bit in the int_status parameter. This parameter will always read back as “0”.
int_mask [6:0]	Active-high mask bits that control the value of the Memory controller_int signal on the Memory Controller interface. This mask is inverted and then logically AND'ed with the outputs of the int_status parameter.
int_status [6:0]	Reports the status of all possible interrupts generated by the Memory Controller. The MSB is the result of a logical OR of all the lower bits. This parameter is read-only. The int_status bits correspond to these interrupts: Bit 0 - A single access outside the defined PHYSICAL memory space detected. Bit 1 - Multiple accesses outside the defined PHYSICAL memory space detected. Bit 2 - DRAM initialization complete. Bit 3 - Address cross page boundary detected. Bit 4 - Both DDR2 and Mobile modes have been enabled. Bit 5 - DLL unlock condition detected. Bit 6 - Logical OR of all other bits.
intrptapburst [0]	Controls whether an interruption of an auto pre-charge command, by another command for a different bank, is allowed. If enabled, the current operation will be interrupted. However, the bank will be pre-charged as if the current operation were allowed to continue. 1'b0 - Interrupt Disable. 1'b1 - Interrupt Enable.

Table 153. Memory controller parameters (continued)

Parameter	Description
intrptreada [0]	Controls whether an interruption of a combined READ with auto pre-charge command, by another READ command toward the same bank before the current READ command has been completed, is allowed. 1'b0 - Interrupt Disable. 1'b1 - Interrupt Enable.
intrptwritea [0]	Controls whether an interruption of a combined WRITE with auto pre-charge command, by another READ or WRITE command toward the same bank before the current WRITE command has been completed, is allowed. 1'b0 - Interrupt Disable. 1'b1 - Interrupt Enable.
Lowpower_auto_enable [4:0]	Sets automatic entry into the low power modes for the Memory Controller. Bit 0 - Controls memory self-refresh with memory and controller clock gating mode (Mode 5). Bit 1 - Controls memory self-refresh with memory clock gating mode (Mode 4). Bit 2 - Controls memory self-refresh mode (Mode 3). Bit 3 - Controls memory power-down with memory clock gating mode (Mode 2). Bit 4 - Controls memory power-down mode (Mode 1). For every bit: 1'b0 - Automatic entry into this mode is disabled. The user may enter this mode manually by setting the associated lowpower_control bit. 1this parameter is set to "b1 - Automatic entry into this mode is enabled. The mode will be entered automatically when the proper counters expire, and only if the associated lowpower_control bit is set. Please refer to *Section 10.7 on page 135* for more details.
lowpower_control [4:0]	Controls the individual low power modes of the device. Bit 0 - Controls memory self-refresh with memory and controller clock gating mode (Mode 5). Bit 1 - Controls memory self-refresh with memory clock gating mode (Mode 4). Bit 2 - Controls memory self-refresh mode (Mode 3). Bit 3 - Controls memory power-down with memory clock gating mode (Mode 2). Bit 4 - Controls memory power-down mode (Mode 1). For every bit: 1'b0 - Disabled. 1'b1 - Enabled. Please refer to *Section 10.7 on page 135* for more details.
lowpower_external_cnt [15:0]	Counts the number of idle cycles before memory self-refresh with memory clock gating low power mode. Please refer to *Section 10.7 on page 135* for more details.

Table 153. Memory controller parameters (continued)

Parameter	Description
lowpower_internal_cnt [15:0]	Counts the number of idle cycles before memory self-refresh with memory and controller clock gating low power mode. Please refer to *Section 10.7 on page 135* for more details.
lowpower_power_down_cnt [15:0]	Counts the number of idle cycles before either memory power-down or power-down with memory clock gating low power mode. Please refer to *Section 10.7 on page 135* for more details.
lowpower_refresh_enable [1:0]	Sets whether refreshes will occur while the Memory Controller is in any of the low power modes. 1'b0 - Refreshes still occur 1'b1 - Refreshes do not occur Please refer to *Section 10.7 on page 135* for more details.
lowpower_refresh_hold [15:0]	Sets the number of cycles that the Memory Controller will wait before attempting to re-lock the DLL when using the controller clock gating mode low power mode. This counter will ONLY be used in this mode, the deepest low power mode. When this counter expires, the DLL will be un-gated for at least 16 cycles during which the DLL will attempt to re-lock. After 16 cycles have elapsed and the DLL has locked, the DLL controller clock will be gated again and the counter will reset to this value. If the DLL requires more than 16 cycles to re-lock, the un-gated time will be longer. Please refer to *Section 10.7 on page 135* for more details.
lowpower_self_refresh_cnt [15:0]	Counts the number of cycles to the next memory self-refresh low power mode. Please refer to *Section 10.7 on page 135* for more details.
max_col_reg [3:0]	Shows the maximum width of column address in the DRAM devices. This value can be used to set the column_size parameter. This parameter is read-only. column_size = max_col_reg - <number of column bits in memory device>.
max_cs_reg [1:0]	Defines the maximum number of chip selects for the Memory Controller as the log2 of the number of chip selects.
max_row_reg [3:0]	Shows the maximum width of the memory address bus (number of row bits) for the Memory Controller. This value can be used to set the addr_pins parameter. This parameter is read-only. addr_pins = max_row_reg - <number of row bits in memory device>.
no_cmd_init [0]	Disables DRAM commands until DLL initialization is complete and tdll has expired. 1'b0 - Issue only REF and PRE commands during DLL initialization of the DRAM devices. 1'b1 - Do not issue any type of command during DLL initialization of the DRAM devices.
ocd_adjust_pdn_cs [4:0]	Sets the off-chip driver (OCD) pull-down adjustment settings for the DRAM devices. The Memory Controller will issue OCD adjust commands to the DRAM devices during power up. Bits 3:0 - Number of OCD adjust commands to be issued. Bit 4 - Increment(1) or decrement(0) OCD settings.

Table 153. Memory controller parameters (continued)

Parameter	Description
ocd_adjust_pup_cs [4:0]	Sets the off-chip driver (OCD) pull-up adjustment settings for the DRAM devices. The Memory Controller will issue OCD adjust commands to the DRAM devices during power up. Bits 3:0 - Number of OCD adjust commands to be issued. Bit 4 - Increment(1) or decrement(0) OCD settings.
odt_add_turn_clk_en [0]	Adds a turn-around clock between back-to-back READs or back-to-back WRITEs to different chip selects. The additional clock may be needed at higher clock frequencies. The “turn off” and “turn on” time of termination resistors are not scalable. At higher clock frequencies, it is possible that these times may overlap, resulting in two active resistors while the DQS line is still active. This could compromise the signal integrity of the DQS signal. The additional clock prevents this overlap. 1'b0 - No additional clocking required. 1'b1 - Additional clock added for back-to-back READs or back-to-back WRITEs that occur to different banks.
odt_rd_map_csX [1:0]	Sets up which (if any) chip(s) will have their ODT termination active while a READ occurs on chip select X.(8) i.e. Since that the system consists of 2 chip selects if the odt_rd_map_cs0 is set to ‘b10, then when CS0 is performing a READ, CS1 will have active ODT termination. Bit 0 - If set to ‘b1 CS0 will have active ODT termination when chip select X us performing a READ. Bit 1 - If set to ‘b1 CS1 will have active ODT termination when chip select X is performing a READ.
odt_wr_map_csX [1:0]	Sets up which (if any) chip(s) will have their ODT termination active while a WRITE occurs on chip select X.(9) i.e. Since that the system consists of 2 chip selects, if the odt_wr_map_cs0 is set to ‘b10, then when CS0 is performing a WRITW, CS1 will have active ODT termination. Bit 0 = CS0 will have active ODT termination when chip select X is performing a WRITE. Bit 1 = CS1 will have active ODT termination when chip select X is performing a WRITE.
out_of_range_addr [33:0]	Holds the address of the command that has begotten an out-of-range interrupt request to the memory devices. This parameter is read-only. For more information on out-of-range address checking, refer to *Section 10.8 on page 140*.
out_of_range_length [9:0]	Holds the length of the command that has begotten an out-of-range interrupt request to the memory devices. This parameter is read-only. For more information on out-of-range address checking, refer to *Section 10.8 on page 140*.
out_of_range_source_id [2:0]	Holds the Source ID of the command that has begotten an out-of-range interrupt request to the memory devices. This parameter is read-only. For more information on out-of-range address checking, refer to *Section 10.8 on page 140*.

Table 153. Memory controller parameters (continued)

Parameter	Description
out_of_range_type [1:0]	Holds the type of command that caused an out-of-range interrupt request to the memory devices. This parameter is read-only. For more information on out-of-range address checking, refer to *Section 10.8 on page 140*.
placement_en [0]	Enables using the placement logic to fill the command queue. 1‘b0 - Placement logic is disabled. The command queue is a straight FIFO. 1‘b1 - Placement logic is enabled. The command queue will be filled according to the placement logic factors.
power_down [0]	When this parameter is set to 1'b1, the Memory Controller will complete processing of the current burst for the current transaction (if any), issue a pre-charge all command and then disable the clock enable signal to the DRAM devices. Any subsequent commands in the command queue will be suspended until this parameter is set to 1'b0. 1'b0 - Enable full power state. 1'b1 - Disable the clock enable and power down the Memory Controller.
priority_en [0]	Controls priority as a condition when using the placement logic to fill the command queue. 1'b0 - Disabled 1'b1 - Enabled
pwrup_srefresh_exit [0]	Controls controller to exit power-down mode by executing a self-refresh instead of the full memory initialization. 1'b0 - Disabled 1'b1 - Enabled
q_fullness [3:0]	Defines quantity of data that will be considered full for the command queue.
rd2rd_turn [0]	Adds an additional clock between back-to-back READ operations to different chip selects. The extra clock is required for mobile DDR devices where: tac_max > (period/2+tac_min) Without this additional clock, the first READ may drive DQS out at tac_max and the second READ may drive DQS out at tac_min, resulting in a contention on the DQS line. 1'b0 - Disabled 1'b1 - Enabled
reduc [0]	Controls the width of the memory datapath. When enabled, the upper half of the memory buses (DQ, DQS and DM) are unused and relevant data only exists in the lower half of the buses. This parameter expands the Memory Controller for use with memory devices of the configured width or half of the configured width. For more information on half datapath mode, refer to *Section 10.8 on page 140*. 1'b0 - Standard operation using full memory bus. 1'b1 - Memory datapath width is half of the maximum size.

Table 153. Memory controller parameters (continued)

Parameter	Description
reg_dimm_enable [0]	Enables registered DIMM operations to control the address and command pipeline of the Memory Controller. 1'b0 - Normal operation 1'b1 - Enable registered DIMM operation.
rtt_0 [1:0]	Defines the On-Die termination resistance for all DRAM devices. The Memory Controller can not be set for different termination values for each chip select. 2'b00 - Termination Disabled 2'b01 - 75 Ohm 2'b10 - 150 Ohm 2'b11 - Reserved
rtt_pad_termination [1:0]	Sets the termination resistance in the Memory Controller pads. The Memory Controller decodes this information and sets the param_75_ohm_sel output signal accordingly. The param_75_ohm_sel signal will be asserted if this parameter is set to 'b01 and de-asserted otherwise. This parameter also disables the output signal tsel, an active-high, dynamic signal which is used in the pads to enable termination on READs. If this parameter is set to 2'b00, the tsel signal will be held low. 2'b00 = Termination Disabled 2'b01 = 75 Ohm 2'b10 = 150 Ohm 2'b11 = Reserved
rw_same_en [0]	Enables READ/WRITE grouping as a condition when using the placement logic to fill the command queue. 1'b0 - Disabled 1'b1 - Enabled
srefresh [0]	When this parameter is set to 1'b1, the DRAM device(s) will be placed in self-refresh mode. For this, the current burst for the current transaction (if any) will complete, all banks will be closed, the self-refresh command will be issued to the DRAM, and the clock enable signal will be de-asserted. The system will remain in self-refresh mode until this parameter is set to 1'b0. The DRAM devices will return to normal operating mode after the self-refresh exit time (txsr) of the device and any DLL initialization time for the DRAM is reached. The Memory Controller will resume processing of the commands from the break point. This parameter will be updated with an assertion of the srefresh_enter pin, regardless of the behavior on the register interface. To disable self-refresh again after a srefresh_enter pin assertion, the user will need to clear the parameter to 1'b0. 1'b0 - Disable self-refresh mode. 1'b1 - Begin self-refresh of the DRAM devices.

Table 153. Memory controller parameters (continued)

Parameter	Description
start [0]	With this parameter set to 'b0, the Memory Controller will not issue any command to the DRAM devices or respond to any signal activity except for reading and writing parameters. Once this parameter is set to 'b1, the Memory Controller will respond to inputs from the device. When set, the Memory Controller begins its initialization routine. When the interrupt bit in the int_status parameter associated with completed initialization is set, the user may begin to submit transactions. 1'b0 - Controller is not in active mode. 1'b1 - Begin active mode for the Memory Controller.
swap_en [0]	Enables swapping of the active command for a new higher-priority command when using the placement logic. 1’b0 - Disabled 1’b1 - Enabled
tcke [2:0]	Defines the minimum CKE pulse width, in cycles.
tcpd [15:0]	Defines the clock enable to pre-charge delay time for the DRAM devices, in cycles.
tdal [3:0]	Defines the auto pre-charge WRITE recovery time when auto pre-charge is enabled (ap is set), in cycles. This is defined internally as tRP (pre-charge time)+auto pre-charge WRITE recovery time. Not all memories use this parameter. If tDAL is defined in the memory specification, then program this parameter to the specified value. If the memory does not specify a tDAL time, then this parameter should be set to tWR+tRP. If this parameter is set to of 0x0 the Memory Controller will not function properly when auto pre-charge is enabled.
tdll [15:0]	Defines the DRAM DLL lock time, in cycles.
temrs [2:0]	Defines the DRAM extended mode parameter set time, in cycles.
tfaw [4:0]	Defines the DRAM tFAW parameter, in cycles.
tinit [23:0]	Defines the DRAM initialization time, in cycles.
tmrd [4:0]	Defines the DRAM mode register set command time, in cycles.
tpdex [15:0]	Defines the DRAM power-down exit command period, in cycles.
tras_lockout [0]	Defines the tRAS lockout setting for the DRAM device. tRAS lockout allows the Memory Controller to execute auto pre-charge commands before the tras_min parameter has expired. 1’b0 - tRAS lockout not supported by memory device. 1’b1 - tRAS lockout supported by memory device.
tras_max [15:0]	Defines the DRAM maximum row active time, in cycles.
tras_min [7:0]	Defines the DRAM minimum row activate time, in cycles.
trc [4:0]	Defines the DRAM period between active commands for the same bank, in cycles.
trcd_int [7:0]	Defines the DRAM RAS to CAS delay, in cycles

Table 153. Memory controller parameters (continued)

Parameter	Description
tref [13:0]	Defines the DRAM cycles between refresh commands. Please refer to *Section 10.8 on page 140* to have more details.
tref_enable [0]	Enables internal refresh commands. If command refresh mode is configured, then refresh commands will be issued based on the internal tref counter and any refresh commands sent through the command interface. 1'b0 - Internal refresh commands disabled. 1'b1 - Internal refresh commands enabled.
trfc [7:0]	Defines the DRAM refresh command time, in cycles. Please refer to *Section 10.8 on page 140* to have more details.
trp [3:0]	Defines the DRAM pre-charge command time, in cycles.
trrd [2:0]	Defines the DRAM activate to activate delay for different banks, in cycles.
trtp [2:0]	Defines the DRAM tRTP (READ to pre-charge time) parameter, in cycles.
twr_int [2:0]	Defines the DRAM WRITE recovery time, in cycles.
twtr [2:0]	Sets the number of cycles needed to switch from a WRITE to a READ operation, as requested by the DDR SDRAM specification.
txsnr [15:0]	Defines the DRAM tXSNR parameter, in cycles.
txsr [15:0]	Defines the DRAM self-refresh exit time, in cycles.
user_def_reg_0 [31:0]	Bit[31:1] - Reserved Bit[0] - Controls READ data retime: 1'b0 = Read data retime in circuit 1'b1 = Read data retime is bypassed For more information on the READ data retime function, please refer to *Section 10.8.4 on page 142*.
user_def_reg_1 [31:0]	This register is not used in the device and should be considered as reserved.
version [15:0]	Holds the Memory Controller version number for this controller. This parameter is read-only For the actual silicon revision (XX,YY) the controller revision is 0x2041
weighted_round_robin_latency _control [0]	Controls the weighted round-robin latency option. 1'b0 - Counters only count when their port has a command waiting to be processed. 1'b1 - Counters are always running.
weighted_round_robin_weight _sharing [1:0]	Reports that the port pair is tied together in arbitration decisions during weighted round-robin arbitration. Bit 0 represents ports 0 and 1, bit 1 represents ports 2 and 3, etc. Bit setting is as follows: 1'b0 - The represented ports are treated independently in arbitration. 1'b1- The represented ports are tied together for arbitration.

Table 153. Memory controller parameters (continued)

Parameter	Description
wr_dqs_shift [6:0]	Sets the delay for the ddr_close signal to ensure correct data capture in the I/O logic. Each increment of this parameter adds a delay of 1/128 of the system clock. The same delay will be added to the clk_dqs_out signal for each slice.(10)
wr_dqs_shift_bypass [9:0]	Sets the delay for the ddr_close signal when the DLL is being bypassed. This is used to ensure correct data capture in the I/O logic The value programmed into this parameter sets the actual number of delay elements in the ddr_close line. If the total delay time programmed exceeds the number of delay elements in the delay chain, the delay will be set internally to the maximum number of delay elements available. (11)
write_modereg [0]	Supplies the EMRS data for each chip select to allow individual chips to set masked refreshing. When this parameter is set to 1'b1, the mode parameter(s) [EMRS register] within the DRAM devices will be written. Each subsequent write_modereg setting will write the EMRS register of the next chip select. This parameter will always read back as 1'b0. The mode registers are automatically written at initialization of the Memory Controller. There is no need to initiate a mode register WRITE after setting the start parameter in the Memory Controller unless some value in these registers needs to be changed after initialization. (12)
writeinterp [0]	Defines whether the Memory Controller can interrupt a WRITE burst with a READ command. Some memory devices do not allow this functionality. 1'b0 - The device does not support READ commands interrupting WRITE commands. 1'b1 - The device does support READ commands interrupting WRITE commands.
wrlat [2:0]	Defines the WRITE latency since the WRITE command is issued until the time the WRITE data is presented to the DRAM devices, in cycles.
wrr_param_value_err [3:0]	Shows the weighted round-robin arbitration errors/warnings. This parameter is read-only. Bit 0 - The port ordering parameters do not all contain unique values. Bit 1 - Any of the relative priority parameters have been programmed with a zero value. Bit 2 - The relative priority values for any of the ports paired through the weighted_round_robin_weight_sharing parameter are not identical. Bit 3 - The port ordering parameter values for paired ports is not sequential.

1. For this parameter and the following ones involving pre-charge concepts, please refer to Bank Splitting. This parameter may not be modified after the start parameter has been asserted.
2. SPEAR™ Memory Controller does not support the MOBILE feature in DDR2 mode. Therefore, setting this bit in conjunction with the MOBILE mode enable bit (en_lowpower_mode) will cause an interrupt.
3. These parameters must be static during normal operation.
4. These parameters must be static during normal operation. While these parameters default to 0x0, the minimum valid value is 0x1. The user should program these parameters to a non-zero value during initialization
5. This parameter must be static during normal operation.

6. This parameter must be static during normal operation. While this parameter defaults to 0x0, the minimum valid value is 0x1. The user should program this parameter to a non-zero value during initialization.
7. SPEAR™ Memory Controller does not support the MOBILE feature in DDR2 mode. Therefore, setting this bit in conjunction with the DDR2 mode enable bit (ddrii_sdram_mode) will cause an interrupt.
8. Only one chip select (and therefore 1 bit) may be set at any time.
9. Only one chip select (and therefore 1 bit) may be set at any time.
10. This parameter must be static during normal operation.
11. This parameter must be static during normal operation.While this parameter defaults to 0x0, The minimum valid value is 0x1. The user must program this parameter to a non-zero value during initialization.
12. This parameter may not be changed when the memory is in power-down mode (when the CKE input is de-asserted).

11 Clock & reset system

The Clock system block is able to generate all clocks necessary at the chip. The main clocks, at default operative frequency, are:

- CPU_CLK @ 333 MHz for the CPUs. (1)
- HCLK @ 166 MHz for AHB Bus and AHB peripherals. (1)
- PCLK @ 83 MHz for APB Bus and APB peripherals. (1)
- DDR_CLK @ 100-333 MHz for DDR memory interface. (2)
- Clock @ 12 MHz, 30 MHz, 48 MHz.(3)

The above frequencies are the maximum allowed values. All these clocks are generated by three PLLs.

PLL1 and PLL2 sources are fully programmable through dedicated registers.

See the sections from *Section 12.4.5: PLL 1/2_CTR registers* in the *Chapter 12: Miscellaneous registers (Misc)*.

The PLLs input reference clocks can be chosen between (see *Section 12.4.8: PLL_CLK_CFG register*)

- 24 MHz oscillator or PL_CLK(4) pad for PLL1
- 24 MHz oscillator or PL_CLK(3) pad for PLL2

At reset the 24 MHz source is selected.

To reduce the electromagnetic emission both PLL1 and PLL2 can be programmed to work in dithered mode.

When the dithered mode is enabled the PLL output clock is modulated and the frequency assumes a triangular shape. In this way the clock power spectrum is spread on a small range (programmable) of frequencies decreasing the emission power peak.

This method replaces the other traditional methods of E.M.I. reduction, as filtering, ferrite beads, chokes, adding power layers and ground planes to PCBs, metal shielding etc., allowing sensible cost saving for customers.

PLL1 and PLL2 can work in three operating modes:

- Normal Mode (Dither-Off Mode): the PLL behaves as a normal PLL
- Fractional-N Synthesis Mode: with this mode it is possible to select VCO frequencies that aren't integer multiples of the reference frequency.
- Dither-On Mode (double side modulation): in this mode a the triangular wave is added to the VCO frequency.
- Dither-On Mode (single side modulation): it is similar to double side modulation but the modulation is only subtracting from the main frequency.

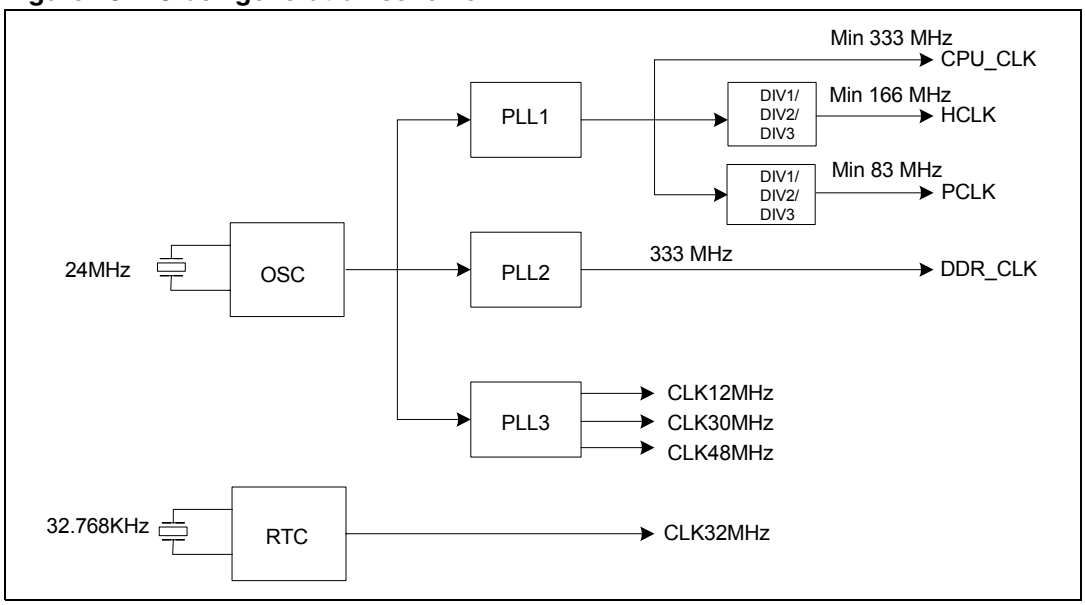
For further information see PLL registers in *Chapter 12: Miscellaneous registers (Misc)*.

PLL3 is used to generate the USB controller clocks and it can't be configured through registers.

1. This frequency is based on the PLL1.
2. This frequency is typically based on the PLL2.
3. This frequency is typically based on the PLL3.

11.1 Clock generation scheme

Figure 13. Clock generation scheme



11.1.1 Jitter at PLL output clock

The three clocks outgoing from the PLLs have a jitter that can be calculated using the formula contained in the figure below.

Table 154. Jitter at PLL output clock

Jitter Type	Jitter Due to Supply Noise (Peak)	Jitter Due to Device Noise (I Sigma)	Total Jitter (Peak to Peak)	Total Jitter at PLL Output Clock @ 333 MHz Input @ 24 MHz N@3		
	A*	B	+/-(A + N * B)**	A*	B	+/-(A + 3 * B)**
Single Period Jitter	30 ps	0.06% of output time period	+/-(A + N * B)	30 ps	1.8 ps	+/-35.4 ps
Cycle to Cycle Jitter	30 ps	0.12% of output time period	+/- (A + N * B)	30 ps	3.6 ps	+/- 40.8 ps

*The jitter specification holds true only up to 50mV noise (peak to peak) on power supply.

**Depending on the requirements of the application, please refer to the following table for estimating the value of N.

The peak to peak jitter is a statistical effect. If a large number of samples (of the clock jitter) are measured the values will usually hold to a normal distribution. The interpretation of peak to peak jitter depends upon the effects of the jitter.

The B parameter (random jitter) is the value of one sigma of this normal distribution, while A parameter is the deterministic jitter.

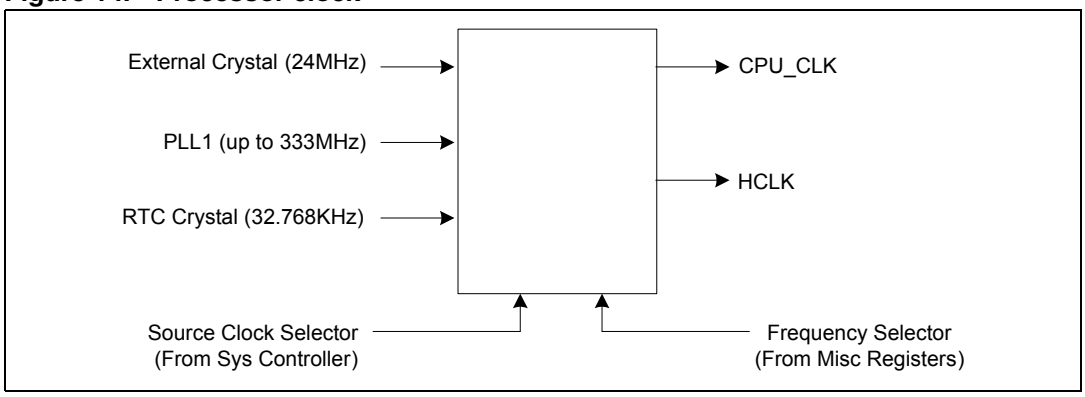
Single period jitter can be defined as the difference of the Tmax and Tmin, where Tmax is maximum time period of the CLOCK and Tmin is the minimum time period of the CLOCK.

Cycle-to-cycle jitter is the cycle time variation between adjacent cycles over a random sample of adjacent clock cycle pairs.

11.2 Clock distribution scheme

11.2.1 Processor clock

Figure 14. Processor clock



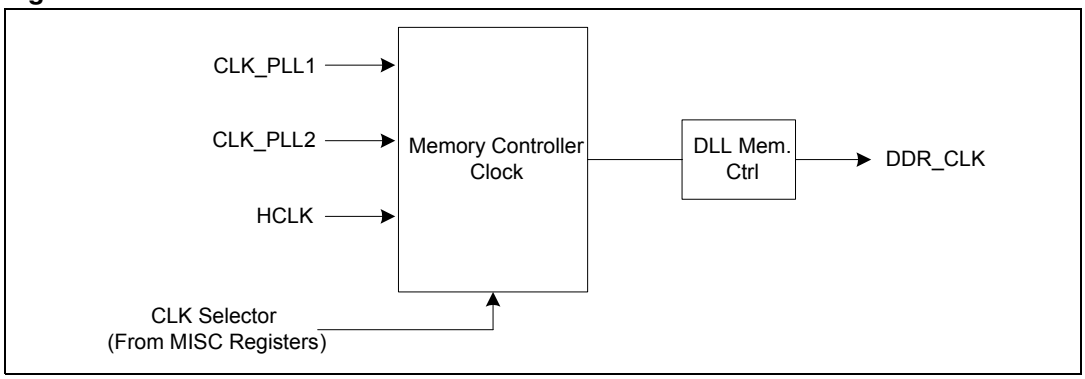
According to the state machine (see *Chapter 14: BS_System controller* for full detail) the CPU clock can be derived from the following sources:

- External crystal (24 MHz)
- PLL1 (up to 333 MHz)
- RTC crystal (32.768 kHz)

By various setting inside the miscellaneous registers is possible to define the frequency of the PLL1 and also the ratio between the CPU clock and the BUS (HCLK) clock.

11.2.2 DDR controller clock

Figure 15. DDR Controller Clock



The Memory controller use the HCLK to synchronize the internal bus access and the other clock, that can be chosen from PLL1 or PLL2 (Misc register setting), is used on the external memory Interface. Two clock domains can be synchronous or asynchronous. In example we can have the CPU running at 333 MHz and the HCLK bus running at 166 MHz but having the external DDR memory running at 266 MHz to reduce the board cost. In this example the PLL1 will provide the clock to all the internal blocks (CPU and bus) while PLL2 will provide the frequency to the external memory. Having the two clock domain asynchronous offer more flexibility in the system definition but also add some more latency due to the resynchronization stages.

11.2.3 Bus clocks

Through the misc CORE_CLK_CFG, PRPH_CLK_CFG register is possible to define the ratio between the CPU clock and its HCLK, the ratio between the AHB and the APB clock, and also the source for the peripheral clock. Typically (default value) this is derived from the fixed frequency of 48 MHz avoiding any setting problem when the bus clock change value because of different system state or because of different PLL setting. Some other register offer also the possibility to enable or disable the clock for each peripheral allowing a sophisticated power management.

11.2.4 Configurable logic clock

The RAS block contains SDIO, Telecom, FSMC and CLCD IPs that are working at AHB frequency. Keyboard and ARM GPIO blocks are working with APB clock frequency.

Figure 16. RAS block diagram

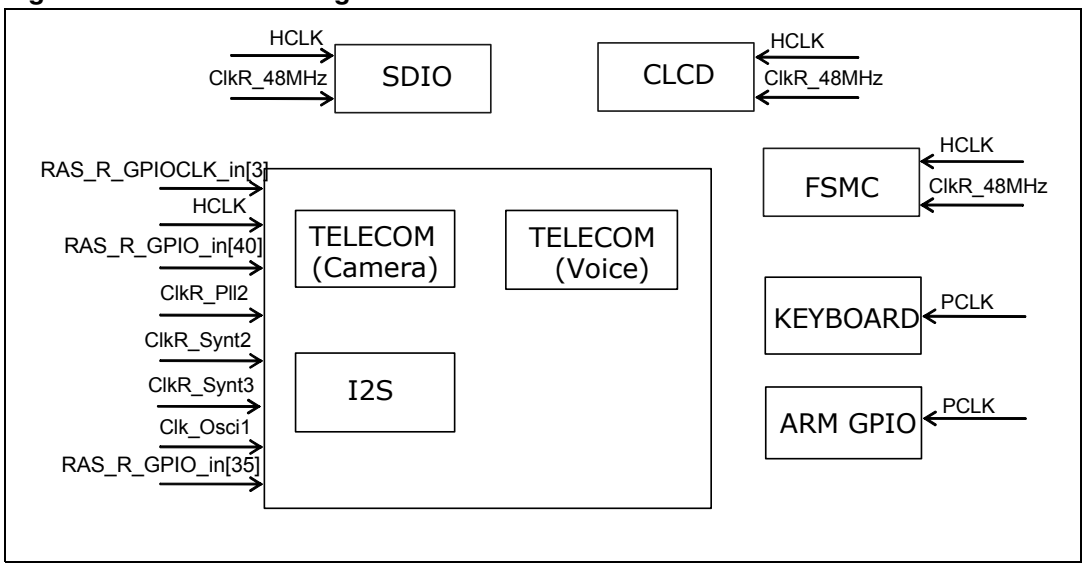


Figure 17. I2S clock schematic

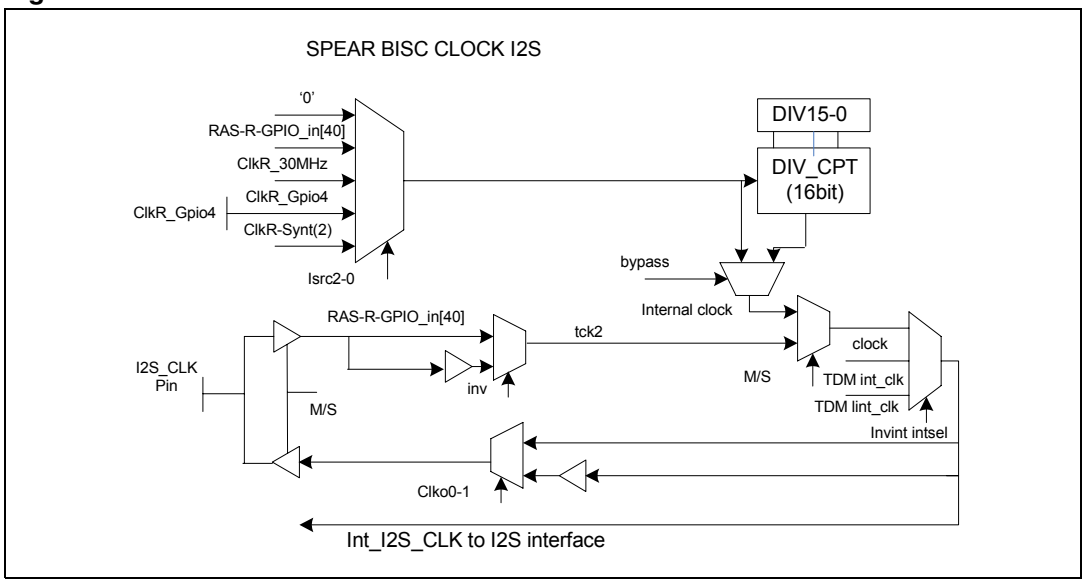
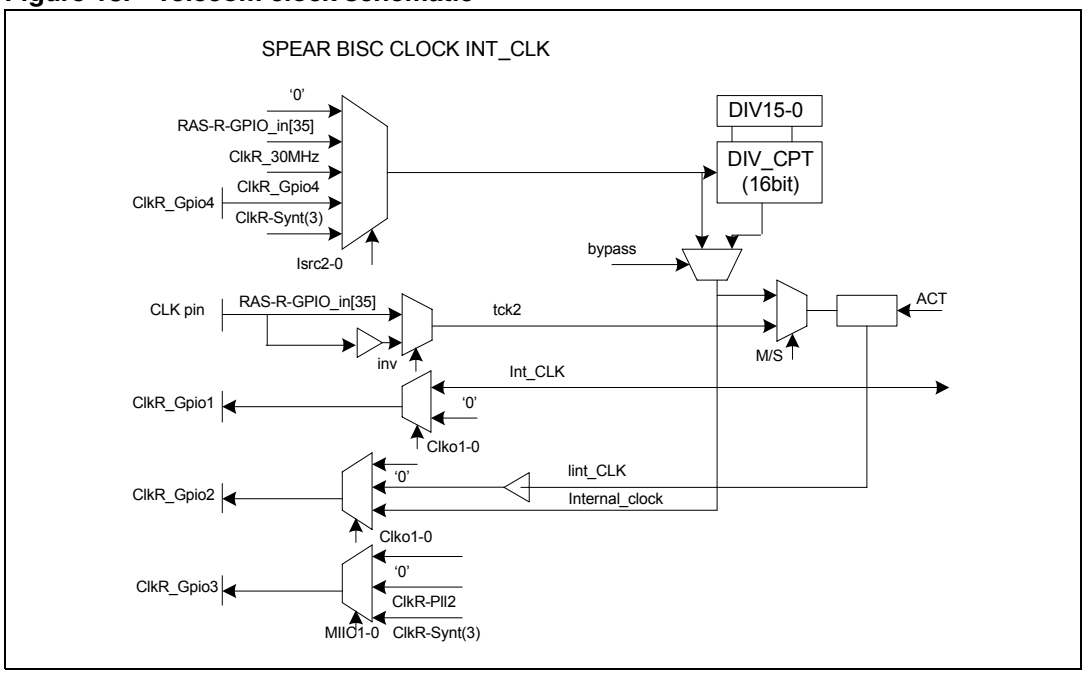


Figure 18. Telecom clock schematic



11.2.5 Clock synthesizer

Clock synthesizer is a digital signal generator. It is used to perform a fractional clock divider.

Giving an input clock with frequency Fin and two integers X (synt_xdiv field of programming register) and Y (synt_xdiv field), it generates a new clock with frequency

With X Y/2, if the post divider is enabled (synt_clkout_sel field is cleared); while

$$F_{out}1 = \left(Fin * \frac{X}{Y} \right) / 2$$

With X Y/2, if post divider is disable (synt_clkout_sel field is set).

$$F_{out}2 = \left(Fin * \frac{X}{Y} \right)$$

Clock synthesizer is based on Y-modulo counter incremented by X. After reset the counter value is zero and it increments of X every input clock cycle. If N is the number of input clock cycle the output is high when:

The counter loads the value Y-NX a clock cycle after that it is verified the previous condition,

$$N * X \geq Y$$

the output become low and the process iterates again. If Y is a multiple of X

$$N = \frac{Y}{X}$$

is a constant and the output period is

$$T_{out} = T_{in} * N = T_{in} * \frac{Y}{X}$$

The output frequency is given by formula.

When Y/X is not an integer value the output period swings between N and N+1 times the input clock period, with N the integer part of Y/X.

This means that the maximum period drift is of one input clock period.

E.g:

With Fout= 40 MHz, Fin= 333 MHz the synthesizer parameters using are:

X=40 and Y=333

This means that the output clock period is on average: Tout = 8.325 * Tin.

The output period will be 8 or 9 times the input clock period:

Tout = 8 * 3 = 24 ns and Tout = 9 * 3 = 27 ns

The maximum output period drift is 27-25 =2 ns.

Since the output clock is high for one input clock cycle the duty cycle is:

$$DC(\%) = \left(\frac{X}{Y} \right) \cdot 100$$

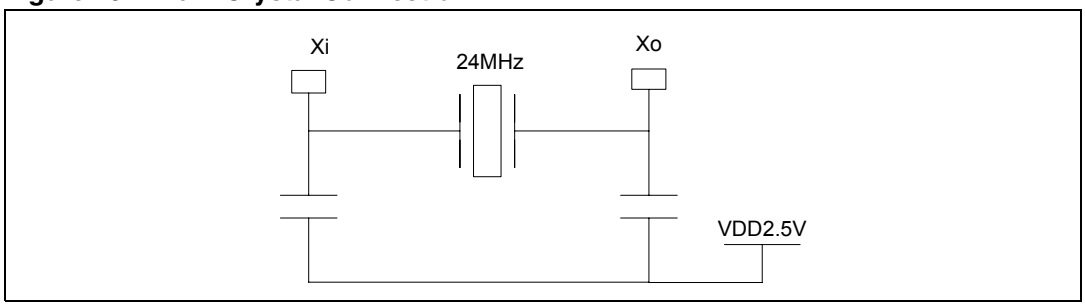
If the post divider by two is enabled the D.C. is 50%; in this case the output frequency is given by.

It can be shown that the period drift in this case is twice the input clock period.

To program the synthesizer please refer to paragraph *Section 12.4.15: Auxiliary clock synthesizer registers* in the Miscellaneous registers (MISC) chapter.Main oscillator

11.2.6 Crystal connection

Figure 19. Main Crystal Connection

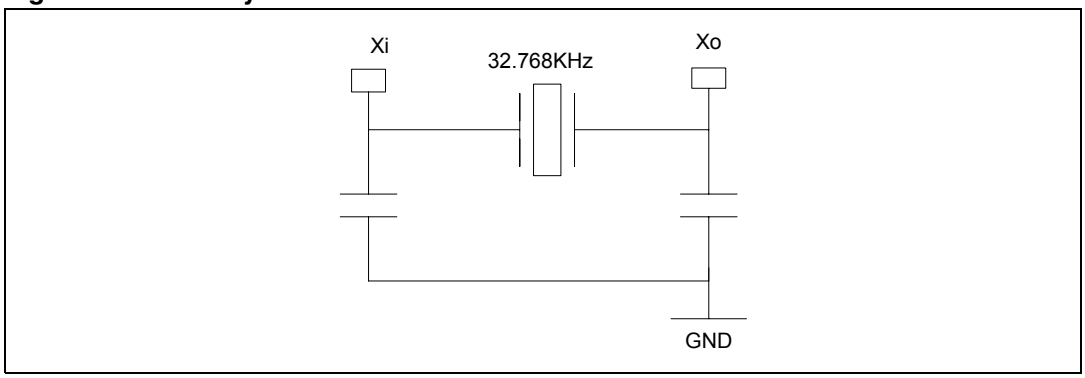


Note: The value of the capacitors depend on the type of the selected crystal. As an example, in STM reference board we have chosen RAKON, P/N Xtal003325 24 MHz oscillator, the value of the capacitors is 33pF.

11.3 RTC oscillator

11.3.1 Crystal connection

Figure 20. RTC crystal connection

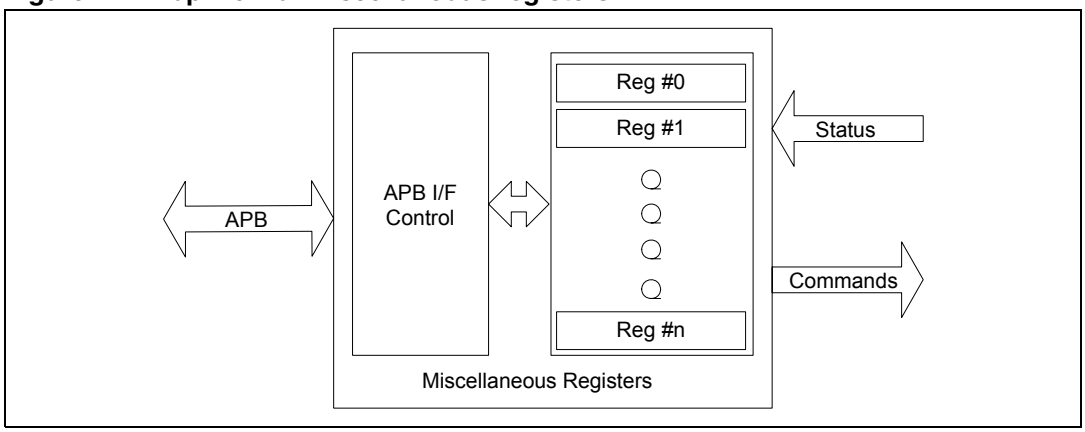


Note: The value of the capacitors depend on the type of the selected crystal. As an example, in STM reference board we have chosen Fox Electronics, P/N NC26LF-327 32.768 kHz oscillator, the value of the capacitors is 15pF.

12 Miscellaneous registers (Misc)

The miscellaneous block is an array of registers which manages the SoC main configuration schemes and controls all basic device functionalities; the top view is given in the next figure.

Figure 21. Top view of miscellaneous registers



12.1 Signal description

The next table shows the APB system interface

Table 155. APB interface signals

APB interface signals		
Signal	**Type**	**Description**
Apb_clk	In	APB port clock.
Apb_resetn	In	APB Input reset.
Apb_addr(31:0)	In	APB Address bus.
Apb_sel	In	APB select.
Apb_enable	In	APB strobe signal.
Apb_write	In	APB write signal.
Apb_wdata(31:0)	In	APB write data bus.
Apb_rdata(31:0)	Out	APB read data bus.

12.2 Overview features

Miscellaneous register is organized in two distinct register regions: local and global register spaces.

- **Local space**: it's a private registers region assigned in order to ensure the right operability of the platform avoiding the register over assignment. The below region controls:
  - SoC application schemes definition.
  - Platform configuration parameters.
- **Global space**: it's a general registers area used to share common functionalities among embedded processor inside the chip. The region controls:
  - Programmable logic (RAS) configuration.
  - Global command and status events.
  - Optional processor mail box data.

12.3 Register address map

Two different register address maps are provided for local and global register spaces which are split into two 32Kbyte sub-region associated with the processor. The local sub-regions are singularly assigned at different physical register regions, while all the global sub-regions are alias of a unique physical region as detailed in the next table.

Table 156. Miscellaneous register main memory map

Miscellaneous register main memory map				
Processor Number	**Local Space**		**Global Space**	
	Region 1-2	**Offset address range**	**Region-1**	**Offset address range**
Proc-1	Region-1	0x0.0000 – 0x0.7FFF	Alias-1	0x0.8000 – 0x0.FFFF

12.4 Miscellaneous register local space

12.4.1 Overview

The local register space controls the following functionalities:-

- SoC main configuration
  - Functional mode (up to 7 configuration are allowed):
  - Normal operating mode.
  - Debug mode: enable and control the processors embedded trace module and Embedded ICE diagnostic functionalities.
  - Test manufacture mode.
- Clock definition and control:
  - Source clock definition.
  - Setting operating frequency.
  - Clock gating control.
  - Auxiliary clock configuration.
- Soft reset control.
- Platform basic configuration parameters:
  - Switch matrix arbitration protocol and priority definition.
  - DMA channel assignment scheme.
  - USB2 Pays setting parameter.
- Special configuration parameters:
  - Compensation pad parameters.
  - Fast IO pad configuration parameters.
  - SSTL pad basic functionality
  - Wake up configuration type.
- Functional memory BIST execution control.
- Diagnostic error detection.

12.4.2 Miscellaneous register local space address map

Next table shows the miscellaneous register map.

Table 157. Miscellaneous local space registers overview

Misc. Local Space Register Map		**Base Address: 0xFCA8.0000**	
Register Name	**Region-1 Offset 0x0.0000**	**Region-2 Offset 0x1.0000**	**Type**
SOC_CFG_CTR	0x000		RO
DIAG_CFG_CTR	0x004		R/W
PLL1_CTR	0x008		R/W
PLL1_FRQ	0x00C		R/W
PLL1_MOD	0x010		R/W

Table 157. Miscellaneous local space registers overview (continued)

Misc. Local Space Register Map		Base Address: 0xFCA8.0000	
Register Name	**Region-1 Offset 0x0.0000**	**Region-2 Offset 0x1.0000**	**Type**
PLL2_CTR	0x014		R/W
PLL2_FRQ	0x018		R/W
PLL2_MOD	0x01C		R/W
PLL_CLK_CFG	0x020		R/W
CORE_CLK_CFG	0x024		R/W
PRPH_CLK_CFG	0x028		R/W
PERIP1_CLK_ENB	0x02C		R/W
Reserved	0x030		-
RAS_CLK_ENB	0x034		R/W
PERIP1_SOF_RST	0x038		R/W
Reserved	0x03C		-
RAS_SOF_RST	0x040		R/W
PRSC1_CLK_CFG	0x044		R/W
PRSC2_CLK_CFG	0x048		R/W
PRSC3_CLK_CFG	0x04C		R/W
AMEM_CFG_CTRL	0x050		R/W
Reserved	0x054		
Reserved	0x058		
Reserved	0x05C		
IRDA_CLK_SYNT_CFG	0x060		R/W
UART0_CLK_SYNT_CFG	0x064		R/W
MAC_CLK_SYNT_CFG	0x068		R/W
RAS_CLK_SYNT1_CFG	0x06C		R/W
RAS_CLK_SYNT2_CFG	0x070		R/W
RAS_CLK_SYNT3_CFG	0x074		R/W
RAS_CLK_SYNT4_CFG	0x078		R/W
ICM1_ARB_CFG	0x07C		R/W
ICM2_ARB_CFG	0x080		R/W
ICM3_ARB_CFG	0x084		R/W
ICM4_ARB_CFG	0x088		R/W
ICM5_ARB_CFG	0x08C		R/W
ICM6_ARB_CFG	0x090		R/W
ICM7_ARB_CFG	0x094		R/W

Table 157. Miscellaneous local space registers overview (continued)

Misc. Local Space Register Map		Base Address: 0xFCA8.0000	
Register Name	**Region-1 Offset 0x0.0000**	**Region-2 Offset 0x1.0000**	**Type**
ICM8_ARB_CFG	0x098		R/W
ICM9_ARB_CFG	0x09C		R/W
DMA_CHN_CFG	0x0A0		R/W
USB2_PHY_CFG	0x0A4		R/W
MAC_CFG_CTR	0x0A8		R/W
Reserved	0x0AC		
Reserved	0x0B0		
Reserved	0x0B4		
Reserved	0x0B8		
Reserved	0x0BC		
Reserved	0x0C0		R/W
Reserved	0x0C4	0xC0	R/W
Reserved	0x0C8		
Reserved	0x0CC		
Reserved	0x0D0		R/W
Reserved	0x0D4	0x0D0	R/W
Reserved	0x0D8		
Reserved	0x0DC		
POWERDOWN_CFG_CTR	0x0E0		R/W
COMPSSTL_1V8_CFG	0x0E4		R/W
Reserved	0x0E8		
COMPCOR_3V3_CG	0x0EC		R/W
DDR_PAD	0x0F0		R/W
BIST1_CFG_CTR	0x0F4		R/W
BIST2_CFG_CTR	0x0F8		R/W
BIST3_CFG_CTR	0x0FC		R/W
BIST4_CFG_CTR	0x100		R/W
Reserved	0x104		R/W
BIST1_STS_RES	0x108		R/W
BIST2_STS_RES	0x10C		R/W
BIST3_STS_RES	0x110		R/W
BIST4_STS_RES	0x114		R/W
Reserved	0x118		

Table 157. Miscellaneous local space registers overview (continued)

Misc. Local Space Register Map		Base Address: 0xFCA8.0000	
Register Name	**Region-1 Offset 0x0.0000**	**Region-2 Offset 0x1.0000**	**Type**
SYSERR_CFG_CTR	0x11C		R/W
USB0_TUN_PRM	0x120		R/W
USB_TUN_PRM	0x124		R/W
USB2_TUN_PRM	0x128		R/W
Reserved[1]	0x12C		R/W
PLGPIO0_PAD_PRG	0x130		R/W
PLGPIO1_PAD_PRG	0x134		R/W
PLGPIO2_PAD_PRG	0x138		R/W
PLGPIO3_PAD_PRG	0x13C		R/W
PLGPIO4_PAD_PRG	0x140		R/W
Reserved[32448]	0x144 0x7FFC		

12.4.3 SoC_CFG_CTR register

The SOC_CFG_CTR is a RO register which handles both functional and test manufacture SoC basic configuration type. The register bit assignments is given in the next table.

Table 158. SoC_CFG_CTR register bit assignments

SoC Functional Configuration Type			0x000
Bit	**Name**	**Reset Value**	**Description**
[31:20]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[19]	boot_sel	-	Reserved for future use (Write don't care - Read return zeros).
[18:06]	Fixed Value	13'h880	Fixed Value

Table 158. SoC_CFG_CTR register bit assignments (continued)

<table>
<tr><td colspan="3">SoC Functional Configuration Type</td><td colspan="3">0x000</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="3">Description</td></tr>
<tr><td rowspan="10">[05:00]</td><td rowspan="10">SoC_cfg</td><td rowspan="10">-</td><td colspan="3">SoC operating mode; this field reflects the Test(4:0) signal values which configure the ASIC main operating modes:
Functional (ref. SoC Functional configuration type Table)
Test manufacture (ref. SoC Test configuration Table)</td></tr>
<tr><td colspan="3">SoC Functional Configuration Type</td></tr>
<tr><td>Soc-cfg</td><td>Name</td><td>Description</td></tr>
<tr><td>X00000</td><td>Dyn_cfg0_0</td><td>Default configuration, I/O standard features (ARM internal debug resources disabled)</td></tr>
<tr><td>X000001</td><td>Dyn_cfg0_1</td><td>Same as Dyn_cfg0_0 but ARM JTAG connected with main JTAG interface</td></tr>
<tr><td>X00010</td><td>Dyn_cfg0_2</td><td>Same as Dyn_cfg0_1 but ETM Interface (Single and double packet mode) multiplexed with programmable PL_GPIOs [73:97]</td></tr>
<tr><td>X00100</td><td>Dyn_cfg1_0</td><td>UART and TIMER ports available on PLGPIO [37:50]</td></tr>
<tr><td>X00101</td><td>Dyn_cfg1_1</td><td>Same as Dyn_cfg1_0 but ARM JTAG connected with main JTAG interface</td></tr>
<tr><td>X00110</td><td>Dyn_cfg1_2</td><td>Same as Dyn_cfg1_1 but ETM Interface (Single and double packet mode) multiplexed with programmable PL_GPIOs [73:97]</td></tr>
<tr><td>X01000</td><td>Dyn_cfg2_0</td><td>Ethernet ports disabled from PLGPIO [27:10]</td></tr>
</table>

Table 158. SoC_CFG_CTR register bit assignments (continued)

SoC Functional Configuration Type					0x000
Bit	**Name**	**Reset Value**	**Description**		
[05:00]	SoC_cfg	-	X01001	Dyn_cfg2_1	Same as Dyn_cfg2_0 but ARM JTAG connected with main JTAG interface
			X01010	Dyn_cfg2_2	Same as Dyn_cfg2_1 but ETM Interface (Single & double packet mode) multiplexed with programmable PL_GPIOs [73:97]
			X01100	Dyn_cfg3_0	UART, TIMER, ETHERNET, I2C and FIrDA ports shared with [50:0]
			X01101	Dyn_cfg3_1	Same as Dyn_cfg3_0 but ARM JTAG connected with main JTAG interface
			X01110	Dyn_cfg3_2	Same as Dyn_cfg3_1 but ETM Interface (Single and double packet mode) multiplexed with programmable PL_GPIOs [73:97]
			SoC Test Configuration Type		
			SoC Cfg	Name	**Description**
			X10000	BSD	Boundary Scan
			X10001	TOP_ATPG	Scan ATPG activities on SoC
			X10010	RAS_ATPG	Scan ATPG activities on programmable logic
			X10011	BIST_MEM	BIST mode involving internal RAMs/ROM
			X10100	USBBIST_PLL_O SCI_ADC	Analog Test: PLLs OSCIs ADC USB2PHY
			X10101	BIST_DLL	DLL BIST mode
			X10110	USB_phy	USB Phy tests

12.4.4 DIAG_CFG_CTR register

The DIAG_CFG_CTR is an R/W register which configures the embedded processors ETM9 (Embedded Trace Module) and Embedded ICE-RT (TAP base debug support) diagnostic functionalities. The register bit assignments is given in the next table.

Table 159. DIAG_CFG_CTR register bit assignments

DIAG_CFG_CTR Register			0x004
Bit	**Name**	**Reset Value**	**Description**
[31:16]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[15]	debug_freez	1'h0	Enable freeze condition when processor enters in debug mode.
[14:12]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[11]	sys_error	-	SoC internal error (RO); reflects SYSERR_CFG_CTR bit(2) value; it's active when an internal error is detected (further details can be found into the SYSERR_CFG_CTR register description 1'b0: No error pending. 1'b1: Active SoC internal error event.
[10:06]	RFU	-	Reserved for future use (Write don't care - Read return zeros).

Table 159. DIAG_CFG_CTR register bit assignments (continued)

<table>
<tr><td colspan="3">DIAG_CFG_CTR Register</td><td colspan="3">0x004</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="3">Description</td></tr>
<tr><td rowspan="20">[05:04]</td><td rowspan="20">SOC_dbg6</td><td rowspan="20">-</td><td colspan="3">SPEAr300 debug configuration (RO); this field is directly reflects the Test (1:0) signals value and it’s used to configure the internal processor Embedded ICE-RT (JTAG port) and ETM debugging features as detailed in the next table.</td></tr>
<tr><td colspan="3">SoC Processor debug cfg6 Configuration Table</td></tr>
<tr><td>SoC-Cfg</td><td>Name</td><td>Description</td></tr>
<tr><td>2’b00</td><td>Dyn_cfg0/1/2/3_0</td><td>Normal mode (ARM internal debug resources disable)</td></tr>
<tr><td>2’b10</td><td>Dyn_cfg0/1/2/3_1</td><td>JTAG1 (ARM JTAG port): connected with main JTAG Interface</td></tr>
<tr><td>2’b01</td><td>Dyn_cfg0/1/2/3_2</td><td>ETM1 (ARM ETM): interface (single & double packets mode) multiplexed with programmable PL_GPIO (38:14) signals (ref. ETM Dbg6 signal. assessment tab)
JTAG1: connected with main JTAG Interface.</td></tr>
<tr><td>2’b11</td><td>RFU</td><td>Reserved for future use.</td></tr>
<tr><td colspan="3">ETM Dbg6 Signal Assignment Table</td></tr>
<tr><td>Standard IOs</td><td colspan="2">Alternative IOs</td></tr>
<tr><td>PL_GPIO(97)</td><td colspan="2">ARM1_TRCCLK</td></tr>
<tr><td>PL_GPIO(96)</td><td colspan="2">ARM1_TRCPKTA(0)</td></tr>
<tr><td>PL_GPIO(95)</td><td colspan="2">ARM1_TRCPKTA(1)</td></tr>
<tr><td>PL_GPIO(94)</td><td colspan="2">ARM1_TRCPKTA(2)</td></tr>
<tr><td>PL_GPIO(93)</td><td colspan="2">ARM1_TRCPKTA(3)</td></tr>
<tr><td>PL_GPIO(92)</td><td colspan="2">ARM1_TRCPKTB(0)</td></tr>
<tr><td>PL_GPIO(91)</td><td colspan="2">ARM1_TRCPKTB(1)</td></tr>
<tr><td>PL_GPIO(90)</td><td colspan="2">ARM1_TRCPKTB(2)</td></tr>
<tr><td>PL_GPIO(89)</td><td colspan="2">ARM1_TRCPKTB(3)</td></tr>
<tr><td>PL_GPIO(88)</td><td colspan="2">ARM1_TRCSYNCA</td></tr>
</table>

Table 159. DIAG_CFG_CTR register bit assignments (continued)

DIAG_CFG_CTR Register				0x004
Bit	**Name**	**Reset Value**	**Description**	
[05:04]	SOC_dbg6	-	PL_GPIO(87)	ARM1_TRCSYNCB
			PL_GPIO(86)	ARM1_PIPSTATA(0)
			PL_GPIO(85)	ARM1_PIPSTATA(1)
			PL_GPIO(84)	ARM1_PIPSTATA(2)
			PL_GPIO(83)	ARM1_PIPSTATB(0)
			PL_GPIO(82)	ARM1_PIPSTATB(1)
			PL_GPIO(81)	ARM1_PIPSTATB(2)
			PL_GPIO(80)	ARM1_TRCPKTA(4)
			PL_GPIO(79)	ARM1_TRCPKTA(5)
			PL_GPIO(78)	ARM1_TRCPKTA(6)
			PL_GPIO(77)	ARM1_TRCPKTA(7)
			PL_GPIO(76)	ARM1_TRCPKTB(4)
			PL_GPIO(75)	ARM1_TRCPKTB(5)
			PL_GPIO(74)	ARM1_TRCPKTB(6)
			PL_GPIO(73)	ARM1_TRCPKTB(7)
			PL_GPIO(86)	ARM1_PIPSTATA(0)
			PL_GPIO(85)	ARM1_PIPSTATA(1)
			PL_GPIO(84)	ARM1_PIPSTATA(2)
			PL_GPIO(83)	ARM1_PIPSTATB(0)
			PL_GPIO(82)	ARM1_PIPSTATB(1)
			PL_GPIO(81)	ARM1_PIPSTATB(2)
			PL_GPIO(80)	ARM1_TRCPKTA(4)
			PL_GPIO(79)	ARM1_TRCPKTA(5)
			PL_GPIO(78)	ARM1_TRCPKTA(6)
			PL_GPIO(77)	ARM1_TRCPKTA(7)
			PL_GPIO(76)	ARM1_TRCPKTB(4)
			PL_GPIO(75)	ARM1_TRCPKTB(5)
			PL_GPIO(74)	ARM1_TRCPKTB(6)
			PL_GPIO(73)	ARM1_TRCPKTB(7)
[03:00]	RFU	-	Reserved for future use (Write don't care - Read return zeros).	

12.4.5 PLL 1/2_CTR registers

The PLL1/2_CTR are R/W registers which configure the operating mode of the main PLLs.

PLL programming sequence

After reset both PLLs must be firstly configured in normal mode waiting for the PLL lock valid status, than these can be optionally reconfigured in dithered mode through an additional specific programming sequence.

Two different output frequency equations are provided for the above PLL operating mode:

- PLL Normal Mode:

$$F_{out} = \frac{2 \times M_{[158]}}{N} \times \frac{F_{in}}{2^P}$$

- PLL Dithered or fractional-N mode

$$F_{out} = \frac{2 \times M}{256XN} \times \frac{F_{in}}{2^P}$$

Table 160. PLL 1/2_CTR register bit assignments

PLL_CTR Register **PLL2_CTR**				**0x008** **0x014**
Bit	**Name**	**Reset Value**	**Description**	
[31:09]	RFU	-	Reserved for future use (Write don't care - Read return zeros).	
[08:03]	pll_control1	6'h0	**PLL Main Configuration Table**	
			Control Bit	Description
			Pll_control1(8) 1'b0 1'b1	External feedback enable: Internal feedback External feedback (dithered mode)
			Pll_control(7:6) 2'b00 2'b01 1X	Sigma Delta Order: 1st Order 2nd Order N.A. (not applicable for current silicon version)
			Pll_control1(5:4) 2'b00 2'b01 2'b10 2'b11	Dither mode: Normal mode (non dithered) Fractional-N Dithering (double side modulation) Dithering (single side modulation)
			Pll_control1(3) 1'b0 1'b1	PLL sample program parameters: No action. Sample program parameters (.)
[02]	pll_enable	1'h0	Enable PLL: 1'b0: Disable PLL (power-down mode). 1'b1: Enable PLL	

Table 160. PLL 1/2_CTR register bit assignments (continued)

PLL_CTR Register PLL2_CTR			0x008 0x014
Bit	**Name**	**Reset Value**	**Description**
[01]	pll_resetn	1'h0	PLL soft reset command: 1'b0: PLL active reset command. 1'b1: PLL reset enable.
[00]	pll_lock	1'h0	PLL Lock Status (RO); field meaningful when PLL is configured in normal mode: 1'b0: PLL unlock status. 1'b1: PLL lock active status.

12.4.6 PLL1/2_FRQ registers

The PLL1/2_FRQ are R/W registers used to configure the PLL VCO frequency operating mode. The register bit assignments is given in the next table.

Table 161. PLL1/2_FRQ register bit assignments

PLL1_FRQ Register PLL2_FRQ			0x00C 0x018
Bit	**Name**	**Reset Value**	**Description**
[31:16]	pll_fbkdiv_M	16'h A600 (.)	M[15:0]: PLL feedback divisor values; when PLL is configured in normal mode only M[15:8] upper byte is considered. Two different equations are provided for the VCO frequency definition which must be programmed within range from 200 MHz min. to 800 MHz max as detailed below: PLL Normal mode configuration $f_{VCD} = 2 \bullet f_{ref} \bullet M_{[15:8]}$ M[15:8] can assume the following range of values: $4 < M < 17$ with $200 < f_{VCD} < 800$ MHz; f_{ref}24 MHz. PLL dithered or fractional-N mode configurations: $f_{VCD} = \frac{2 \bullet f_{ref}}{256} \bullet M$ [15:0] can assume the following range of values: $1066 < M < 4266$ with $200 < f_{VCD} < 800$ MHz; f_{ref}24 MHz.
[15:11]	RFU	-	Reserved for future use (Write don't care - Read return zeros).

Table 161. PLL1/2_FRQ register bit assignments (continued)

<table>
<tr><td colspan="8">PLL1_FRQ Register
PLL2_FRQ</td><td>0x00C
0x018</td></tr>
<tr><td>Bit</td><td colspan="2">Name</td><td>Reset Value</td><td colspan="5">Description</td></tr>
<tr><td rowspan="10">[10:08]</td><td colspan="2" rowspan="10">pll_postdiv_P</td><td rowspan="10">3’h1</td><td colspan="5">Post divider (P) table</td></tr>
<tr><td colspan="5">P(2:0): PLL post-divisor values in 1:32 in 2’powers (ref. Post Divider table)</td></tr>
<tr><td colspan="2">Pdiv2</td><td>Pdiv1</td><td>Pdiv0</td><td>Division factor</td></tr>
<tr><td colspan="2">0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td colspan="2">0</td><td>0</td><td>1</td><td>2</td></tr>
<tr><td colspan="2">0</td><td>1</td><td>0</td><td>4</td></tr>
<tr><td colspan="2">0</td><td>1</td><td>1</td><td>8</td></tr>
<tr><td colspan="2">1</td><td>0</td><td>0</td><td>16</td></tr>
<tr><td colspan="2">1</td><td>0</td><td>1</td><td>32</td></tr>
<tr><td colspan="2">1</td><td>1</td><td>0</td><td>32</td></tr>
<tr><td colspan="2">1</td><td>1</td><td>1</td><td>32</td></tr>
<tr><td rowspan="13">[07:00]</td><td colspan="2" rowspan="13">pll_prediv_N</td><td rowspan="13">8’h0C</td><td colspan="5">N(7:0): PLL pre-divisor programmable value from 1 to 255 (ref. Pre-divisor table)
The reference clock fref should be within the range below:
$1MHz <= f_{re(f <= 40Mhz)}$
The reference clock value is given from the following formula:
$f_{ref} = \frac{f_{osci}}{N}$</td></tr>
<tr><td colspan="5">Pre-divider (N) table</td></tr>
<tr><td>div 7</td><td>div 6</td><td>div 5</td><td>div 4</td><td>div 3</td><td>div 2</td><td>div 1</td><td>div 0</td><td>Div fact.</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>~</td><td>~</td><td>~</td><td>~</td><td>~</td><td>~</td><td>~</td><td>~</td><td>~</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>254</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>255</td></tr>
</table>

12.4.7 PLL1/2_MOD registers

The PLL1/2_MOD is R/W registers which configure the dithering modulation parameters. The register bit assignments is given in the next table.

Table 162. PLL1/2_MOD register bit assignments

PLL1_MOD Register PLL2_MOD			0x010 0x01C
Bit	**Name**	**Reset Value**	**Description**
[31:29]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[28:16]	pll_modperiod	13'h0	MP(12:0) PLL modulation wave parameters: Modulation rate f_{mod} depends from reference clock $f_{ref;}$ and modulation period mp as detailed in the next formula : $f_{mod}(KHz) = \frac{f_{ref(KHz)}}{4 \bullet mp}$ Example: If f_{ref} = 24000 kHz and f_{mod} = 100 kHz the modulation period register will be mp=60. Any changes in the reference clock results in changes in the modulation frequency. The maximum modulation frequency that can pass through the filter is 100 kHz.
[15:00]	pll_slope	16'h0	SR(15:0) PLL slope modulation wave parameters: The slope modulation rate reflects the modulation-depth (md) in respect to the nominal frequency of the un-dithered clock as shown in the next formula: $sr = \frac{2^{17}}{f_{ref}^2} \bullet md \bullet f_{VCD} \bullet f_{mod}$ Where sr in the actual value of the slope register. Example: If md=2.5% and f_{VCD} = 576 MHz and f_{mod} = 100 kHz with f_{ref} = 24 MHz, (using the simplified formula) it results: $sr = \frac{2^8}{mp} \bullet md \bullet M$ $sr = \frac{256 \bullet 0 \bullet 025 \bullet 3072}{60} = 327 = 0 \times 0147$

12.4.8 PLL_CLK_CFG register

The PLL_CLK_CFG is an R/W register used to configure the input source clock for all the internal PLLs. The register bit assignments is given in the next table.

Table 163. PLL_CLK_CFG register bit assignments

PLL_CLK_CFG Register			0x020	
Bit	**Name**	**Reset Value**	**Description**	
[31]	RFU	-	Reserved for future use (Write don't care - Read return zeros).	
[30:28]	mctr_clk_sel	3'h0	MPMC memory controller DDR_CLK configuration.	
			3'b000	Synch mode: core clock provided from PLL1: 1:1 for DDRCORE_CLK the reference frequency is HCLK. DDR_CLK = HCLK.
			3'b001	Synch mode: core clock provided from PLL1: 2:1 for DDRCORE_CLK the reference frequency is 2x HCLK. DDR_CLK = 2 x HCLK. *Note: Ratio 2:1 must also be be set in the ahbX_fifo_type_reg parameter (see Table 151 in Section 11)*
			3'b010	Reserved for future use.
			3'b011	Asynch mode: core clock provided from PLL2 >1:1 (clock up to 333 MHz). <1:1 (clock range 100 - 166 MHz).
			3'b1XX	Reserved for future use.
[27]	RFU	-	Reserved for future use (Write don't care - Read return zeros).	
[26-24]	pll2_clk_sel	3'h0	**Auxiliary PLL2 source clock configuration**	
			Control Bit	Description
			3'b000	24 MHz Oscillator (default mode)
			3'b001	Programmable PL_CLK (3) signal.
			3'b010	Reserved for future use
			3'b011	Reserved for future use.
			3'b1XX	Reserved for future use.
[23]	RFU	-	Reserved for future use (Write don't care - Read return zeros.)	

Table 163. PLL_CLK_CFG register bit assignments (continued)

PLL_CLK_CFG Register			0x020	
Bit	**Name**	**Reset Value**	**Description**	
[22:20]	pll1_clk_sel	3'h0	**Main PLL1 source clock configuration table**	
			Control Bit	Description
			3'b000	24 MHz Oscillator (default mode)
			3'b001	Programmable PL_CLK (4) signal.
			3'b01X	Reserved for future use.
			3'b1XX	Reserved for future use.
[19]	mem_dll_lock	-	Memory DLL lock; this field reflects the current value of memory controller DLL lock signal (RO): 1'b0: DLL unlock status (for interrupt capability ref. SYSERR_CFG_CTR register description). 1'b1: DLL active lock.	
[18]	usb_pll_lock	-	USB PLL3 llock; this field reflects the current value of USB PLL3 llock signal (RO): 1'b0: USB PLL3 unlock status (for interrupt capability ref. SYSERR_CFG_CTR register description). 1'b1: PLL3 active lock.	
[17]	sys_pll2_lock	-	Auxillary System PLL2 lock; this field reflects the current value of System PLL2 lock signal (RO): 1'b0: PLL2 unlock status (for interrupt capability ref. SYSERR_CFG_CTR register description). 1'b1: Pll2 active lock. This field should be ignored when PLL2 is programmed in dithering mode.	
[16]	sys_pll1_lock	-	Main System PLL1 lock; this field reflects the current value of the System PLL1 lock signal (RO): 1'b0: PLL1 unlock status (for interrupt capability ref. SYSERR_CFG_CTR register description). 1'b1: Pll1 active lock. This field should be ignored when PLL1is programmed in dithering mode.	
[15:03]	RFU	-	Reserved for future use (Write don't care - Read return zeros).	
[02]	pll3_enb_clkout	1'h0	Enable USB PLL3 clock output probing; this functionality is used to check the internal PLL3 clock integrity: 1'b0: Disable clock probing (normal mode). 1'b1: PLL3 clock out (48 MHz) multiplexed on basGPIO(2) signal.	

Table 163. PLL_CLK_CFG register bit assignments (continued)

PLL_CLK_CFG Register			0x020
Bit	**Name**	**Reset Value**	**Description**
[01]	pll2_enb_clkout	1'h0	Enable PLL2 clock output probing; this functionality is used to check the internal PLL2 clock integrity: 1'b0: Disable clock probing (normal mode). 1'b1: PLL2 clock out (clk1 x 1/8) multiplexed on basGPIO(1) signal.
[00]	pll1_enb_clkout	1'h0	Enable PLL1 clock output probing; this functionality is used to check the internal PLL1 clock integrity: 1'b0: Disable clock probing (normal mode). 1'b1: PLL1 clock out (clk1 x 1/8) multiplexed on basGPIO(0) signal

12.4.9 CORE_CLK_CFG register

The CORE_CLK_CFG is an R/W register used to configure the internal platform clock domains. The register bit assignments is given in the next table.

Table 164. CORE_CLK_CFG register bit assignments

CORE_CLK_CFG Register			0x024		
Bit	**Name**	**Reset Value**	**Description**		
[31:22]	RFU	-	Reserved for future use (Write don't care - Read return zeros).		
[21:20]	Osci24_div-ratio	2'h0	**OSCI24 divider config. table**		
			Control bit	**Ratio**	**Description**
			2'b00	1:2	24 MHz to divider out ratio.
			2'b01	1:4	24 MHz to divider out ratio.
			2'b10	1:16	24 MHz to divider out ratio.
			2'b11	1:32	24 MHz to divider out ratio.
[19]	Osci24_div_en	1'h0	When set the 24 MHz Oscillator clock, used in SLOW and DOZE mode for the AMBA subsystem, is divided by a prescaler. The prescaler division factor can be set through osci24_div_ratio field.		
[18]	ras_synt34_clksel	1'h0	Current field selects the RAS clock synthesizer Synt-3 and Synt-4 input source clock (ref. Auxiliary clock synthesizer register description): 1'b0: Clock synthesizer input freq. Fin = PLL1 output clock (333 MHz). 1'b1: Clock synthesizer input freq. Fin = PLL2 output clock (programmable value)		
[17:12]	RFU	-	Reserved for future use (Write don't care - Read return zeros).		

Table 164. CORE_CLK_CFG register bit assignments (continued)

<table>
<tr><td colspan="4">CORE_CLK_CFG Register</td><td>0x024</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="2">Description</td></tr>
<tr><td rowspan="7">[11:10]</td><td rowspan="7">hclk_divsel</td><td rowspan="7">2’h0</td><td colspan="2">PLL1_clkout to HCLK clock ratio definition (ref. next table)</td></tr>
<tr><td colspan="2">PLL1_clkout to HCLK configuration table</td></tr>
<tr><td>Control bit</td><td>Ratio</td><td>Description</td></tr>
<tr><td>2’b00</td><td>1:1</td><td>Hclk to Pll1_clkout ratio.</td></tr>
<tr><td>2’b01</td><td>1:2</td><td>Hclk to Pll1_clkout ratio.</td></tr>
<tr><td>2’b10</td><td>1:3</td><td>Hclk to Pll1_clkout ratio.</td></tr>
<tr><td>2’b11</td><td>1:4</td><td>Hclk to Pll1_clkout ratio.</td></tr>
<tr><td rowspan="7">[09:08]</td><td rowspan="7">pclk_ratio_lwsp</td><td rowspan="7">2’h0</td><td colspan="2">Low speed subsystem PCLK clock ratio divider (ref. next table)</td></tr>
<tr><td colspan="2">HCLK to PCLK clock ratio configuration table</td></tr>
<tr><td>Control bit</td><td>Ratio</td><td>Description</td></tr>
<tr><td>2’b00</td><td>1:1</td><td>Hclk to Pclk clock ratio.</td></tr>
<tr><td>2’b01</td><td>1:2</td><td>Hclk to Pclk clock ratio.</td></tr>
<tr><td>2’b10</td><td>1:3</td><td>Hclk to Pclk clock ratio.</td></tr>
<tr><td>2’b11</td><td>1:4</td><td>Hclk to Pclk clock ratio.</td></tr>
<tr><td>[07:06]</td><td>RFU</td><td>-</td><td colspan="2">Reserved for future use (Write don’t care - Read return zeros).</td></tr>
<tr><td>[05:04]</td><td>pclk_ratio_basc</td><td>2’h0</td><td colspan="2">Basic subsystem PCLK clock ratio divider (ref. HCLK to PCLK clock ratio configuration table)</td></tr>
<tr><td>[03:02]</td><td>RFU</td><td>-</td><td colspan="2">Reserved for future use (Write don’t care - Read return zeros).</td></tr>
<tr><td>[01:00]</td><td>pclk_ratio_arm1</td><td>2’h0</td><td colspan="2">ARM subsystem PCLK clock ratio divider (ref. HCLK to PCLK clock ratio configuration table).</td></tr>
</table>

12.4.10 PRPH_CLK_CFG register

The PRPH_CLK_CFG is an R/W register used to configure the peripheral source clock definition. The register bit assignments is given in next table.

Table 165. PRPH_CLK_CFG register bit assignments

PRPH_CLK_CFG Register			0x028
Bit	**Name**	**Reset Value**	**Description**
[31:18]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[17]	gptmr3_freez	1'h0	General purpose timer-3 clock enable 1'b0: enable clock 1;'b1: disable clock
[16]	gptmr2_freez	1'h0	General purpose timer-2 clock enable 1'b0: enable clock 1;'b1: disable clock
[15:14]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[13]	gptmr1_freez	1'h0	General purpose time-1 clock enable 1'b0: enable clock 1;'b1: disable clock
[12]	gptrmr3_clksel	1'h0	GPT3 General purpose timer 3 source clock selection 1'b0: 48 MHz (default clock) 1'b1: Clock prescaler (PRSC3_CLK_CFG)n
[11]	gptmr2_clksel	1'h0	GPT2 General purpose timer 2 source clock selection 1'b0: 48 MHz (default clock) 1'b1: Clock prescaler (PRSC2_CLK_CFG)
[10:09]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[08]	gptmr1_clksel	1'h0	GPT1 General purpose timer 1 source clock selection 1'b0: 48 MHz (default clock) 1'b1: Clock prescaler (PRSC1_CLK_CFG).
[07]	rtc_disable	1'h1	Real Time Clock enable. 1'b0: RTC clock enable (to be enabled to set 32 kHz as the input clock source in DOZE mode). 1'b1: Disable RTC clock (disable 32 kHz as the input clock source in DOZE mode)
[06:05]	irda_clksel	2'h0	IrDA source clock selection 2'b00: 48 MHz (default clock) 2'b01: IrDA clock synthesizer *Section 12.4.15: Auxiliary clock synthesizer registers* 2'b10: External PL_CLK (3) signal. 2'b11: Reserved.
[04]	uart_clksel	1'h0	UART0 source clock selection 1'b0: 48 MHz (default clock) 1'b1:UART0 Clock Synthesizer *Section 12.4.15: Auxiliary clock synthesizer registers*

Table 165. PRPH_CLK_CFG register bit assignments (continued)

PRPH_CLK_CFG Register			0x028
Bit	**Name**	**Reset Value**	**Description**
[03:02]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[01]	plltimeen	1'h1	Enable PLL1 timer: this functionality replace PLL lock signals and it's used to control the switch transition from slow to normal operating mode when System controller PLL1 timeout event expires: 1'b0: Disable PLL1 timer functionality. 1'b1: Enable PLL1 timer switching transition; set from Processor to switch into the normal operating frequency either after the initialization sequence complete or to restore the normal operating condition from a dynamic power down sequence (power save).
[00]	xtaltimeen	1'h0	Enable Xtal timer: this functionality enables an auxiliary timer to control the switch transition from doze to slow operating mode when system controller Xtal timeout event expires: 1'b0: Disable Xtal timer functionality: the switch transition is controlled from macro-oscillator clock enable signal. 1'b1: Enable Xtal timer; set from Processor to ensure the oscillator output clock stable before to enter in slow operating mode.

12.4.11 PERIP1_CLK_ENB register

The PERIP1_CLK_ENB is an R/W register which controls the peripheral clock enable functionality. The register bit assignments is given in the next table.

Table 166. PERIP1_CLK_ENB register bit assignments

PERIP1_CLK_ENB Register			0x02C
Bit	**Name**	**Reset Value**	**Description**
[31]	C3_clock_enb	1'h1	1'b0: Disable C3 clock 1'b1: Enable C3 clock
[30]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[29]	ddr_core_enb	1'h1	DDR memory controller core clock enable; functionality asserted setting '0' the PERIPH1_CLK_ENB[27] after a previous write with PERIPH1_CLK_ENB[29,27]=01: 1'b0: Disable DDR core clock gating functionality. 1'b1: Enable DDR core clock gating functionality.
[28]	RFU	-	Reserved for future use (Write don't care - Read return zeros)

Table 166. PERIP1_CLK_ENB register bit assignments (continued)

PERIP1_CLK_ENB Register			0x02C
Bit	**Name**	**Reset Value**	**Description**
[27]	ddr_clkenb	1'h1	1'b0: Disable DDR memory controller core clock. 1'b1: Enable DDR memory controller core clock. Note: Command allowed when ddr_core_enb bit is active high.
[26]	usbh_clock	1'h1	1'b0: Used to disable USB ehci host reset. 1'b1: Used to enable USB ehci host reset.
[25]	usbh1_clkenb	1'h0	1'b0: Disable USB host clock. 1'b1: Enable USB host clock.
[24]	usbdev_clkenb	1'h0	1'b0: Disable USB device clock. 1'b1: Enable USB device clock.
[23]	MAC_clkenb	1'h0	1'b0: Disable MAC Ethernet clock. 1'b1: Enable MAC Ethernet clock.
[22]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[21]	smi_clkenb	1'h1	1'b0: Disable serial Flash controller clock. 1'b1: Enable serial Flash controller clock.
[20]	rom_clkenb	1'h1	1'b0: Disable ROM controller clock. 1'b1: Enable ROM controller clock.
[19]	DMA_clkenb	1'h0	1'b0: Disable DMA controller clock. 1'b1: Enable DMA controller clock.
[18]	GPIO_clkenb	1'h0	1'b0: Disable GPIO clock. 1'b1: Enable GPIO clock.
[17]	rtc_clkenb	1'h0	1'b0: Disable real time controller clock. 1'b1: Enable real time controller clock.
[16]	RFU	-	-
[15]	adc_clkenb	1'h0	1'b0: Disable ADC controller clock. 1'b1: Enable ADC controller clock.
[14:13]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[12]	GPT3 clkenb	1'h0	1'b0: Disable GPT 3 of basic subsystem clock. 1'b1: Enable GPT 3 of basic subsystem clock.
[11]	GPT2 clkenb	1'h0	1'b0: Disable GPT 2 of basic subsystem clock. 1'b1: Enable GPT 2 of basic subsystem clock.
[10]	firda_clkenb	1'h0	1'b0: Disable IrDA clock. 1'b1: Enable IrDA clock.
[09]	RFU		
[08]	jpeg_clkenb	1'h0	1'b0: Disable JPEG codec clock. 1'b1: Enable JPEG codec clock.

Table 166. PERIP1_CLK_ENB register bit assignments (continued)

PERIP1_CLK_ENB Register			0x02C
Bit	**Name**	**Reset Value**	**Description**
[07]	i2c_clkenb	1'h0	1'b0: Disable I^2C clock. 1'b1: Enable I^2C clock.
[06]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[05]	ssp_clkenb	1'h0	1'b0: Disable SPI clock. 1'b1: Enable SPI clock.
[04]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[03]	uart_clkenb	1'h1	1'b0: Disable UART0 clock. 1'b1: Enable UART0 clock.
[02]	RFU		Reserved for future use
[01]	arm_clkenb	1'h1	1'b0: Disable ARM subsystem clock. 1'b1: Enable ARM subsystem clock. Note: Command allowed when arm_enb bit is active high.
[00]	arm_enb	1'h0	ARM clock enable; functionality asserted setting '0' the PERIPH1_CLK_ENB[1] after a previous write with PERIPH1_CLK_ENB[1,0]=01: 1'b0: Disable ARM clock gating functionality. 1'b1: Enable ARM clock gating functionality.

12.4.12 RAS_CLK_ENB register

The RAS_CLK_ENB is an R/W register which controls the internal programmable logic clock enable functionality. The register bit assignments are given in the next table.

Table 167. RAS_CLK_ENB register bit assignments

RAS_CLK_ENB Register			0x034
Bit	**Name**	**Reset Value**	**Description**
[31:16]	reserved	-	Reserved for future use (Write don't care - Read return zeros).
[15]	pl_gpck4_clkenb	1'h0	1'b0: Disable PL_CLK(4) external clock signal. 1'b1: Enable PL_CLK(4) external clock signal.
[14]	pl_gpck3_clkenb	1'h0	1'b0: Disable PL_CLK(3) external clock signal. 1'b1: Enable PL_CLK(3) external clock signal.
[13]	pl_gpck2_clkenb	1'h0	1'b0: Disable PL_CLK(2) external clock signal. 1'b1: Enable PL_CLK(2) external clock signal.
[12]	pl_gpck1_clkenb	1'h0	1'b0: Disable PL_CLK(1) external clock signal. 1'b1: Enable PL_CLK(1) external clock signal.

Table 167. RAS_CLK_ENB register bit assignments (continued)

RAS_CLK_ENB Register			0x034
Bit	**Name**	**Reset Value**	**Description**
[11]	ras_synt4_clkenb	1'h0	1'b0: Disable internal synthesizer-4 source clock. 1'b1: Enable internal synthesizer-4 source clock.
[10]	ras_synt3_clkenb	1'h0	1'b0: Disable internal synthesizer-3 source clock. 1'b1: Enable internal synthesizer-3 source clock.
[09]	ras_synt2_clkenb	1'h0	1'b0: Disable internal synthesizer-2 source clock. 1'b1: Enable internal synthesizer-2 source clock.
[08]	ras_synt1_clkenb	1'h0	1'b0: Disable internal synthesizer-1 source clock. 1'b1: Enable internal synthesizer-1 source clock.
[07]	pll2_clkenb	1'h0	1'b0: Disable PLL2 source clock. 1'b1: Enable PLL2 source clock.
[06]	RFU		
[05]	clk48M_clkenb	1'h0	1'b0: Disable 48 MHz internal source clock. 1'b1: Enable 48 MHz internal source clock.
[04]	Clk24M_clkenb	1'h0	1'b0: Disable 24 MHz external source clock signal. 1'b1: Enable 24 MHz external source clock signal.
[03]	clk32K_clkenb	1'h0	1'b0: Disable 32 kHz external source clock signal. 1'b1: Enable 32 kHz external source clock signal.
[02]	pclkappl_clkenb	1'h0	1'b0: Disable internal PCLK (APB application Subsystem) source clock. 1'b1: Enable internal PCLK (APB application Subsystem) source clock.
[01]	pll1_clkenb	1'h0	1'b0: Disable PLL1 source clock. 1'b1: Enable PLL1 source clock.
[00]	hclk_clkenb	1'h0	1'b0: Disable internal AHB HCLK source clock. 1'b1: Enable internal AHB HCLK source clock.

12.4.13 PRSC1/2/3_CLK_CFG register

The PRSC1/2/3_CLK_CFG are three RW registers used to configure the timer pre scalar frequencies. The output frequency is given from the following expressions:

$$F_{out} = \frac{F_{in}}{2^{(N+1)} \times (M+1)}$$

with M < 4096, N < 16; Fin = (PLL1 out frequency) 333 MHz. F_{out} Max 83 MHz.

The register bit assignments is detailed in the next table.

Table 168. PRSC1/2/3_CLK_CFG register bit assignments

PRSC1_CLK_CFG Register PRSC2_CLK_CFG PRSC3_CLK_CFG			0x044 0x048 0x04C
Bit	**Name**	**Reset Value**	**Description**
[31:16]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[15:12]	presc_n	4'h0	N (3:0) constant factor division value: N < 16.
[11:00]	presc_m	12'h0	M (11:0) constant division value: M < 4096.

12.4.14 AMEM_CFG_CTRL register

The AMEM_CFG_CTRL is an R/W register which configures and controls the asynchronous/synchronous memory port-1 source clock definition.

The output frequency originated from the x/y clock synthesizer is given from the next equation:

$$F_{out} = \left(Fin \times \frac{X}{Y}\right) / 2$$

with Y < 256; X < Y/2; Fin = (ref. amem_synt_enb source clock definition).

The register bit assignments is detailed in the next table.

Table 169. AMEM_CFG_CTRL register bit assignments

AMEM_CFG_CTRL Register			0x050
Bit	**Name**	**Reset Value**	**Description**
[31:24]	amem_xdiv	8'h0	X(7:0) clock synthesizer constant division: X < Y/2.
[23:16]	amem_ydiv	8'h0	Y(7:0) clock synthesizer constant division: Y<256.
[15]	amem_rst	1'h0	Memory port-1 soft reset command: 1'b0: Disable soft reset. 1'b1: Active soft reset command.
[14:05]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[04]	amem_synt_enb	1'h0	Enable memory port-1 clock synthesizer: 1'b0: Disable memory clock synthesizer; memory clock is provided in agree with the amem_clksel source clock definitions. 1'b1: Enable memory clock synthesizer; memory clock is provided from clock synthesizer logic (ref. amen Fout equation)

Table 169. AMEM_CFG_CTRL register bit assignments (continued)

AMEM_CFG_CTRL Register			0x050
Bit	**Name**	**Reset Value**	**Description**
[03:01]	amem_clk_sel	3'h0	Memory port-1 source clock definition (ref. next table) Memory port2 source clock configuration table Control Bit: Description 3'b000: HCLK (synchronous operating mode) (.) 3'b001: PLL1(clock synthesizer should be enable). 3'b010: PLL2 (clock synthesizer should be enable). 3'b011: Ras_clk (programmable logic output clock). *Note: This clock bypass the memory clock synthesizer logic.* 3'b100-111: Reserved (RFU)
[00]	amem_clk_enb	1'h0	Memory port-1 clock gating functionality: 1'b0: Disable memory clock. 1'b1: Enable memory clock.

12.4.15 Auxiliary clock synthesizer registers

The Auxiliary clock synthesizers are a group of R/W registers which provide an auxiliary source clock for some internal target devices: IrDA, UART, MII, and the RAS IPs.

If y is integer multiple of x, the clock generated will have less jitter.

The output frequency originated from every clock synthesizer is given from the following equations:

$$F_{out}1 = \left(Fin \times \frac{X}{Y}\right)/2$$

$$F_{out}2 = \left(Fin \times \frac{X}{Y}\right)$$

with Y < 4096; X < Y/2; Fin = (ref. Clock synthesizer input frequency table)

For further details, please refer to *Chapter 11: Clock & reset system* .

The clock synthesizer input frequency is detailed in the next table:

Table 170. Clock Synthesizer input frequency

Clock synthesizer input frequency			
Clock synthesizer	**Src. Clk1 PLL1**	**Src. Clk2 PLL2**	**Description**
IRDA	*		Clock provided from Pll1_clkout
UART	*		Clock provided from Pll1_clkout
MAC		*	Programmable source clock (Ref. MAC_CFG_REG register description).

Table 170. Clock Synthesizer input frequency (continued)

Clock synthesizer input frequency			
Clock synthesizer	**Src. Clk1 PLL1**	**Src. Clk2 PLL2**	**Description**
RAS1	*		Clock provided from Pll1_clkout
RAS2	*		Clock provided from Pll1_clkout
RAS3	*	*	Source clock selected from 'ras_synt34_clksel' register field
RAS4	*	*	Source clock selected from 'ras_synt34_clksel' register field

The register bit assignments is given in the next table.

Table 171. Auxiliary clock synthesizer register bit assignments

Reserved **IRDA_CLK_SYNT_CFG** **UART0_CLK_SYNT_CFG** **MAC_CLK_SYNT_CFG** **RAS_CLK_SYNT1_CFG** **RAS_CLK_SYNT2_CFG** **RAS_CLK_SYNT3_CFG** **RAS_CLK_SYNT4_CFG**			**0x054 to 0x05C** **0x060** **0x064** **0x068** **0x06C** **0x070** **0x074** **0x078**
Bit	**Name**	**Reset Value**	**Description**
[31]	synt_clk_enb	1'h0	Enable clock synthesizer functionality (.) 1'b0: Disable clock synthesizer. 1'b1: Enable clock synthesizer.
[30]	synt_clkout_sel	1'h0	Output Clock Synthesizer selection: 1'b0: Output frequency derived from F_{out} 1 equation. 1'b1: Output frequency derived from F_{out}2 equation.
[29:28]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[27:16]	synt_xdiv	12'h0	X_(11:0) clock synthesizer constant division: X<Y/2
[15:12]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[11:00]	synt_ydiv	12'h0	Y_(11:0) clock synthesizer constant division: Y < 4096

12.4.16 Soft reset control

12.4.17 PERIP1_SOF_RST register

The PERIP1_SOF_RST is an R/W register used to control the peripheral soft reset functionality. The register bit assignments is given in the next table.

Table 172. PERIP1_SOF_RST register bit assignments

PERIP1_SOF_RST Register			0x038
Bit	**Name**	**Reset Value**	**Description**
[31]	C3_reset	1'h1	1'b0: Disable C3 reset. 1'b1: Active C3 reset.
[30]	RFU	-	-
[29]	ddr_core_enbr	1'h0	DDR memory controller reset enable; functionality asserted setting '0' the PERIPH1_LOC_RST [27] after a previous write with PERIPH1_LOC_RST [29,27]=11: 1'b0: Disable DDR core soft reset command. 1'b1: Enable DDR core soft reset command.
[28]	ram_swrst	1'h1	1'b0: Disable Basic subsystem RAM reset. 1'b1: Active Basic subsystem RAM reset command.
[27]	ddr_swrst	1'h0	1'b0: Disable DDR core controller reset. 1'b1: Active DDR core controller reset. Note: Command allowed when ddr_core_enbr bit is active high.
[26]	usbh1_ehci_swrst	1'h1	1'b0: Disable USB ehci host reset. 1'b1: Active USB ehci host reset.
[25]	usbh1_ohci_swrst	1'h1	1'b0: Disable USB ohci host reset. 1'b1: Active USB ohci host reset.
[24]	usbdev_swrst	1'h1	1'b0: Disable USB device reset. 1'b1: Active USB device reset.
[23]	MAC_swrst	1'h1	1'b0: Disable MAC Ethernet reset. 1'b1: Active MAC Ethernet reset.
[22]	RFU	1'h1	Reserved for future use (Write don't care - Read return zeros).
[21]	smi_swrst	1'h0	1'b0: Disable serial Flash controller reset. 1'b1: Active serial Flash controller reset.
[20]	rom_swrst	1'h0	1'b0: Disable ROM controller reset. 1'b1: Active ROM controller reset.
[19]	DMA_swrst	1'h1	1'b0: Disable DMA controller reset. 1'b1: Active DMA controller reset.
[18]	gpio_swrst	1'h1	1'b0: Disable GPIO reset. 1'b1: Active GPIO reset.
[17]	rtc_swrst	1'h1	1'b0: Disable real time controller reset. 1'b1: Active real time controller reset.
[16]	RFU		Reserved for future use.
[15]	adc_swrst	1'h1	1'b0: Disable ADC controller reset. 1'b1: Active ADC controller reset.
[14]	RFU	1'h1	Reserved for future use (Write don't care - Read return zeros).
[13]	RFU		Reserved for future use.

Table 172. PERIP1_SOF_RST register bit assignments (continued)

PERIP1_SOF_RST Register			0x038
Bit	**Name**	**Reset Value**	**Description**
[12]	gptm3_swrst	1'h1	1'b0: Disable general purpose timer-3 reset. 1'b1: Active general purpose timer-3 reset.
[11]	gptm2_swrst	1'h1	1'b0: Disable general purpose timer-2 reset. 1'b1: Active general purpose timer-2 reset.
[10]	firda_swrst	1'h1	1'b0: Disable irda reset. 1'b1: Active irda reset.
[09]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[08]	jpeg_swrst	1'h1	1'b0: Disable JPEG codec reset. 1'b1: Active JPEG codec reset.
[07]	i2c_swrst	1'h1	1'b0: Disable I^2C reset. 1'b1: Active I^2C reset.
[06]	RFU		Reserved for future use.
[05]	ssp_swrst	1'h1	1'b0: Disable SPI reset. 1'b1: Active SPI reset.
[04]	RFU	1'h1	Reserved for future use (Write don't care - Read return zeros).
[03]	uart_swrst	1'h0	1'b0: Disable UART reset. 1'b1: Active UART reset.
[02]	RFU	1'h1	Reserved for future use (Write don't care - Read return zeros).
[01]	arm1_swrst	1'h0	1'b0: Disable ARM subsystem reset. 1'b1: Active ARM subsystem reset Note: Command allowed when arm1_enbr bit is active high.
[00]	arm1_enbr	1'h0	Arm1 reset enable; functionality asserted setting '0' the PERIPH1_LOC_RST[1] after a previous write with PERIPH1_LOC_RST [1,0]=11: 1'b0: Disable ARM soft reset command. 1'b1: Enable ARM soft reset command.

12.4.18 RAS_SOF_RST register

The RAS_SOF_RST is an R/W register which controls the internal programmable logic soft reset functionality. The register bit assignments is given in the next table.

Table 173. RAS_SOF_RST register bit assignments

RAS_SOF_RST Register			0x040
Bit	**Name**	**Reset Value**	**Description**
[31:16]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[15]	pl_gpck4_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[14]	pl_gpck3_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[13]	pl_gpck2_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[12]	pl_gpck1_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[11]	ras_synt4_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[10]	ras_synt3_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[09]	ras_synt2_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[08]	ras_synt1_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[07]	pll2_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[06]	clk125M_swrst	1'h1	This bit is always 1 since GBIT Ethernet is not supported
[05]	clk48M_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[04]	Clk24M_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[03]	Clk32K_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[02]	pclkappl_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[01]	pll1_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.
[00]	hclk_swrst	1'h1	1'b0: Disable reset command. 1'b1: Active reset command.

12.4.19 SoC configuration basic parameters

12.4.20 ICM1-8_ARB_CFG register

The ICM1-8_ARB_CFG is a group of R/W registers which configure the embedded interconnection matrix arbitration protocol and the priority level of each masters; the next table shows the relations from all ICMs and their correspondent logic domains.

Table 174. Interconnection matrix

Interconnection Matrix		
ICM properties		**Logic domain**
ICM number	**ICM Master number**	
ICM-1	3	Low speed subsystem
ICM-2	3	Application subsystem
ICM-3	2	Basic subsystem
ICM-4	2	High speed subsystem
ICM-5	2	Memory controller port-2
ICM-6	2	RAS_F port
ICM-7	4	Memory controller port-3
ICM-8	2	Memory controller port-4

The register bit assignments is given in the next table.

Table 175. ICM 1-9_ARB_CFG register bit assignments

ICM1_ARB_CFG Register	**0x07C**
ICM2_ARB_CFG	**0x080**
ICM3_ARB_CFG	**0x084**
ICM4_ARB_CFG	**0x088**
ICM5_ARB_CFG	**0x08C**
ICM6_ARB_CFG	**0x090**
ICM7_ARB_CFG	**0x094**
ICM8_ARB_CFG	**0x098**

Bit	Name	Reset Value	Description
[31]	mtx_arb_type	1'h0	Interconnect matrix arbitration Protocol definition: 1'b0: Fixed priority arbitration type; the arbitration policy is done in agree with the priority level definition of each master (level 0 is the highest priority). 1'b1: Round robin arbitration type.
[30:28]	mxt_rndrb_pry_lyr	3'h0	This field specifies the priority starting level (from o to 7) used from round robin arbitration protocol. This field is not relevant in case of fixed arbitration scheme.

Table 175. ICM 1-9_ARB_CFG register bit assignments (continued)

ICM1_ARB_CFG Register	**0x07C**
ICM2_ARB_CFG	**0x080**
ICM3_ARB_CFG	**0x084**
ICM4_ARB_CFG	**0x088**
ICM5_ARB_CFG	**0x08C**
ICM6_ARB_CFG	**0x090**
ICM7_ARB_CFG	**0x094**
ICM8_ARB_CFG	**0x098**

Bit	Name	Reset Value	Description
[27:24]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[23:21]	mtx_fix_pry_lyr 7	3'h0	Reserved field not applicable for current silicon version.
[20:18]	mtx_fix_pry_lyr 6	3'h0	Reserved field not applicable for current silicon version.
[17:15]	mtx_fix_pry_lyr 5	3'h0	Reserved field not applicable for current silicon version.
[14:12]	mtx_fix_pry_lyr 4	3'h0	Reserved field not applicable for current silicon version.
[11:09]	mtx_fix_pry_lyr 3	3'h0	Master layer-3 fixed priority number level (from 0 to 7). This field is relevant only for ICM7_ARB_CFG registers. (ref. Fixed priority number level definition table). (.)
[08:06]	mtx_fix_pry_lyr 2	3'h0	Master layer-1 fixed priority number level (from 0 to 7) (ref. Fixed priority number level definition table). (.)
[05:03]	mtx_fix_pry_lyr 1	3'h0	Master layer-1 fixed priority number level (from 0 to 7) (ref. Fixed priority number level definition table). (.)

Table 175. ICM 1-9_ARB_CFG register bit assignments (continued)

ICM1_ARB_CFG Register	**0x07C**
ICM2_ARB_CFG	**0x080**
ICM3_ARB_CFG	**0x084**
ICM4_ARB_CFG	**0x088**
ICM5_ARB_CFG	**0x08C**
ICM6_ARB_CFG	**0x090**
ICM7_ARB_CFG	**0x094**
ICM8_ARB_CFG	**0x098**

Bit	Name	Reset Value	Description	
[02:00]	mtx_fix_pry_lyr 0	3'h0	Master layer-0 fixed priority number level (from 0 to 7); (ref. next table). (.)	
			Fixed priority level definition table	
			Control Bit	**Description**
			3'b000	Priority level 0 (highest)
			3'b001	Priority level 1
			3'b010	Priority level 2
			3'b011	Priority level 3
			3'b100	Priority level 4
			3'b101	Priority level 5
			3'b110	Priority level 6
			3'b111	Priority level 7 (lowest)

Note: 1 Field ignored in case of round-robin arbitration type.

2 In case more masters share the same priority level the lowest master number is granted.

12.4.21 DMA_CHN_CFG register

The DMA_CHN_CFG is an R/W register which configures the DMA channels assignment scheme among different requester agents. Two basic assignment schemes are supported for current silicon version:

- DMA_Sch_0: Core logic domain
- DMA_Sch_1: RAS domain

The register bit assignment is given in the next table.

Table 176. DMA_CHN_CFG register bit assignment

DMA_CHN_CFG Register					0x0A0
Bit	**Name**	**Reset Value**	**Description**		
			DMA channel configuration scheme: this field configures each DMA channel assignment. Please refer to *Table 588: RAS DMA configuration* for the Sch_1 channel assignments.(the configuration value '10' and '11' are reserved and not applicable for current silicon version).		
			CHAN	**(Sch_0)00**	**(Sch_1)01**
[31:30]	dma_cfg_chan 15	2'h0	15	FROM_JPEG	RAS_7 Tx
[29:28]	dma_cfg_chan 14	2'h0	14	TO_JPEG	RAS_7 Rx
[27:26]	dma_cfg_chan 13	2'h0	13	ADC	RAS_6 Tx
[25:24]	dma_cfg_chan 12	2'h0	12	IrDA Rx/Tx	RAS_6 Rx
[23:22]	dma_cfg_chan 11	2'h0	11	I^2C Tx	RAS_5 Tx
[21:20]	dma_cfg_chan 10	2'h0	10	I^2C Rx	RAS_5 Rx
[19:18]	dma_cfg_chan 09	2'h0	09	SSP0_Tx	RAS_4 Tx
[17:16]	dma_cfg_chan 08	2'h0	08	SSP0_Rx	RAS_4 Rx
[15:14]	dma_cfg_chan 07	2'h0	07	Reserved	RAS_3 Tx
[13:12]	dma_cfg_chan 06	2'h0	06	Reserved	RAS_3 Rx
[11:10]	dma_cfg_chan 05	2'h0	05	Reserved	RAS_2 Tx
[09:08]	dma_cfg_chan 04	2'h0	04	Reserved	RAS_2 Rx
[07:06]	dma_cfg_chan 03	2'h0	03	UART0 Tx	RAS_1 Tx
[05:04]	dma_cfg_chan 02	2'h0	02	UART0 Rx	RAS_1 Rx
[03:02]	dma_cfg_chan 01	2'h0	01	Reserved	RAS_0 Tx
[01:00]	dma_cfg_chan 00	2'h0	00	Reserved	RAS_0 Rx

12.4.22 USB2_PHY_CFG register

The USB2_PHY_CFG is a R/W register which configures the USB2 triple Phy Basic parameters. The register bit assignments is given in the next table.

Table 177. USB2_PHY_CFG register bit assignments

USB2_PHY_CFG Register			0x0A4
Bit	**Name**	**Reset Value**	**Description**
[31:04]	RFU	-	Reserved for future use.
[03]	usbh_overcur	1'h0	USB host over-current: enable USB controller to enter in power down state when an electrical overcurrent condition is detected on the corresponding USB bus: To check 1'b0: Disable functionality 1'b1: Enable over-current detection functionality.
[02:01]	RFU	-	Reserved for future use.
[00]	PLL_pwdn	1'h0	This bit controls the state of PLL blocks when in Suspend mode 1'b0: PLL blocks powered up during Suspend mode 1'b1: PLL blocks powered down during Suspend mode

12.4.23 MAC_CFG_CTR register

The MAC_CFG_CTR is an R/W register which configures the MAC Ethernet internal source clock. The register bit assignments is given in the next table.

Table 178. MAC_CFG_CTR register bit assignments

<table>
<tr><td colspan="4">MAC_CFG_CTR Register</td><td>0x0A8</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="2">Description</td></tr>
<tr><td>[31:05]</td><td>RFU</td><td>-</td><td colspan="2">Reserved for future use (Write don't care - Read return zeros).</td></tr>
<tr><td>[04]</td><td>MAC_synt_enb</td><td>1'h0</td><td colspan="2">MAC GMII/MII clock synthesizer enable:
1'b0: Disable MAC clock synthesizer: GMII/MII clock is provided in agree with the MAC_clk_sel source clock definitions.
1'b1: Enable MAC clock synthesizer: GMII/MII clock is provided from clock synthesizer logic (ref. MAC_CLK_SYNT_CFG register description).(1)</td></tr>
<tr><td rowspan="7">[03:02]</td><td rowspan="7">MAC_clk_sel</td><td rowspan="7">2'h0</td><td colspan="2">MAC internal source clock definition(1)</td></tr>
<tr><td colspan="2">MII source clock definition table</td></tr>
<tr><td>Control Bit</td><td>Source clock</td><td>Description</td></tr>
<tr><td>2'b00</td><td>External</td><td>MII_txclk25' signal</td></tr>
<tr><td>2'b01</td><td>Internal</td><td>PLL2 output clock</td></tr>
<tr><td>2'b10</td><td>External</td><td>24 MHz oscillator</td></tr>
<tr><td>2'b11</td><td>-</td><td>Reserved</td></tr>
</table>

Table 178. MAC_CFG_CTR register bit assignments (continued)

MAC_CFG_CTR Register			0x0A8
Bit	**Name**	**Reset Value**	**Description**
[01]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[00]	mili_reverse	1'h0	MII normal/reverse mode configuration type: 1'b0: MII normal mode (external Eth. PHY connection): both Txclk and Rxclk bidirectional signals are configured with input direction and the MII clocks are provided from the external Phy. 1'b1: MII reverse mode (MII to MII direct connection): both Txclk and RXclk bidirectional signals are configured with output direction and the MII clock are provided from the internal logic.

1. MII frequency definition should be compliant with IEEE-803.3 std: MII Txclk/Rxclk 25/2.5 MHz.

12.4.24 Special configuration parameters

12.4.25 Powerdown_CFG_CTR register

The POWERDOWN_CFG_CTR is an R/W register which configures the interrupt wakeup type used in conjunction with the dynamic power down functionality. The register bit assignments is given in the next table.

Table 179. Powerdown_CFG_CTR register bit assignments

Powerdown_CFG_CTR Register			0x0E0
Bit	**Name**	**Reset Value**	**Description**
[31:01]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[00]	wakeup_fiq_enb	1'h0	Wakeup interrupt type (Firq/Irq) definition; this field selects the interrupt type detected from processor-1 to restore the normal operating frequency from the power down state (switch from sleep to doze/low speed operating mode): **1'b0: Irq interrupt type**: the peripheral interrupt requests lines are also used as a wakeup source event increasing the overall interrupt latency time. **1'b1: Firq interrupt type**: single global interrupt request line which ensure both fast recovery time from power down state and the best peripheral interrupt response time since the wakeup SW interrupt service routine is centralized. The wakeup interrupt vector is defined in the processor-1 interrupt table line-15.(1)

1. IRQ interrupt type should be masked before to enter in sleep mode,

12.4.26 COMPSSTL_1V8_CFG/DDR_2V5_COMPENSATION register

The COMPSSTL_1V8_CFG/DDR_2V5_COMPENSATION is R/W registers which configure the internal SSTL compensation cells parameters. The register bit assignments is given in the next table.

Table 180. COMPSSTL_1V8_CFG/DDR_2V5_COMPENSATION register bit assignments

COMPSSTL_1V8_CFG Register			0x0E4
Bit	**Name**	**Reset Value**	**Description**
[31]	TQ	1'h0	It enables IDDq mode.
[30:24]	rasrc	7'h78	Writing code compensation parameter: field sampled from the compensation macro-cell during Read operating mode command (ref. Compensation cell operating mode table).
[23]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[22:16]	nasrc	-	Read code compensation parameter (RO); this field is qualified from 'sts_ok' active high.
[15:05]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[04]	COMPOK	1'h0	Valid code compensation (RO); field actives high in normal mode when the measured code is available on the compensation bus nasrc.
[03]	accurate	1'h1	Compensation cell internal/external reference resistance definition: 1'b0: Internal reference resistor 1'b1: External reference resistor: used to improve the accuracy of compensation code value.
[02]	freeze	1'h0	Freeze command: when high freezes the current calculated value of compensation bus.
[01]	comptq	1'h0	Compensation cell internal command parameter (ref. Compensation cell operating mode table)
[00]	compen	1'h0	Compensation cell internal command parameter (ref. Compensation cell operating mode table).

12.4.27 COMPCOR_3V3_CFG register

The COMPCOR_3V3_CFG is a R/W register which configures the internal CORE compensation cell parameters. The register bit assignments is given in the next table.

Table 181. COMPCOR_3V3_CFG register bit assignments

COMPCOR_3V3_CFG Register			0x0EC
Bit	**Name**	**Reset Value**	**Description**
[31]	TQ	1'h0	It enables IDDq mode.
[30:24]	RASRC	7'h78	Writing code compensation parameter sample from the compensation macro-cell during Read operating mode (ref. Compensation cell operating mode table).
[23]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[16:22]	NASRC	7'h0	Copy of the code on compensation bus (Note that bit 30 of RASRC is mapped on bit 16 and so on)
[15:05]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[04]	COMPOK	1'h0	Valid code compensation (RO); field actives high in normal mode when the measured code is available on the compensation bus nasrc.
[03]	ACCURATE	1'h0	Compensation cell internal/external reference resistance definition: 1'b0: Internal reference resistor 1'b1: External reference resistor: used to improve the accuracy of compensation code value.
[02]	FREEZE	1'h0	Freeze command: when high freezes the current calculated value of compensation bus.
[01]	COMPTQ	1'h1	Compensation cell internal command parameter (ref. Compensation cell operating mode table)
[00]	COM PEN	1'h0	It selects macro_cell operating modes.

12.4.28 DDR_PAD register

The DDR_PAD is a R/W register which configures the SSTL pad internal parameters.The register bit assignments is given in the next table.

Table 182. DDR_PAD register bit assignments

DDR_PAD Register			0x0F0
Bit	**Name**	**Reset Value**	**Description**
[31:19]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[18:15]	DDR_SW-mode[3:0]	4'h0	When '0110' the selection of DDR2/DDR (Low power) is decided by SW. In all other cases, the selection is decided by the value of DDR2_EN pad.

Table 182. DDR_PAD register bit assignments (continued)

<table>
<tr><td colspan="3">DDR_PAD Register</td><td colspan="3">0x0F0</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="3">Description</td></tr>
<tr><td>[14]</td><td>DDR_EN_PAD</td><td>-</td><td colspan="3">It contains the value decided in the external pad DDR2_EN to select DDR (Low Power) or DDR2 (RO)
1’b0: DDR2
1’b1: DDR Low Power</td></tr>
<tr><td>[13]</td><td>REFSSTL</td><td>1’h1</td><td colspan="3">Internal/External SSTL common reference voltage definition:
1’b0: Internal reference voltage
1’b1: External reference voltage to be applied on DDR_MEM_REF signal</td></tr>
<tr><td>[12]</td><td>GATE_OPEN_mode</td><td>1’h1</td><td colspan="3">It selects the internal (1) or external (0) gate open mode. It is connected to stp_asic IP.</td></tr>
<tr><td>[11]</td><td>RFU</td><td>-</td><td colspan="3">Reserved for future use (Write don’t care - Read return zeros).</td></tr>
<tr><td>[10]</td><td>ENZI</td><td>1’h0</td><td colspan="3">Input buffer enable. Active low, connected to ENZI of all the pads.</td></tr>
<tr><td>[09]</td><td>DQS_PDN_sel</td><td>1’h1</td><td colspan="3">DQS Pull down. It is connected to PDCLK of SSTL differential pads.</td></tr>
<tr><td>[08]</td><td>DQS_PU_sel</td><td>1’h0</td><td colspan="3">DQS pull up, it is connected to PUCLK of SSTL different pads.</td></tr>
<tr><td>[07]</td><td>CLK_PDN_sel</td><td>1’h1</td><td colspan="3">Programmable CLK Pull down functionality connected with both PDCLK and PCLKB signals of SSTL different pads (ref. Pull-up/down configuration table)</td></tr>
<tr><td>[06]</td><td>CLK_PU_sel</td><td>1’h0</td><td colspan="3">Programmable CLK Pull up functionality connected with both PUCLK and PUCLKB signal of SSTL different pads (ref. Pull-up/down configuration table)</td></tr>
<tr><td rowspan="7">[05]</td><td rowspan="7">PDN_sel</td><td rowspan="7">1’h1</td><td colspan="3">Enable active Pull Down for SE SSTL pads (ref. next table)</td></tr>
<tr><td colspan="3">Pull/up down configuration table</td></tr>
<tr><td>Pull-up</td><td>Pull-Down</td><td>Description</td></tr>
<tr><td>0</td><td>1</td><td>Pull-up/down not actives</td></tr>
<tr><td>1</td><td>1</td><td>Active pull-up</td></tr>
<tr><td>0</td><td>0</td><td>Active pull-down</td></tr>
<tr><td>1</td><td>0</td><td>Forbidden</td></tr>
<tr><td>[04]</td><td>PU-sel</td><td>1’h0</td><td colspan="3">Pull up activation for SE SSTL pads (ref. Pull-up/down configuration table)</td></tr>
</table>

Table 182. DDR_PAD register bit assignments (continued)

DDR_PAD Register			0x0F0
Bit	**Name**	**Reset Value**	**Description**
[03]	S_W_mode	1'h1	SSTL pad drive strength mode: the overall drive strength picture is detailed here below. 1'b0: Strong drive strength. 1'b1: Weak drive strength. This bit changes the output impedance of the pad.
[02]	PROG_a	1'h0	Combination of these bits selects the speed of operation of PAD, 00->200 MHz, 01->266 MHz, 10->333 MHz, 11->Prohibited.
[01]	PROG_b	1'h1	
[00]	DDR_LOW_POWER_DDR2_mode	1'h0	It selects DDR2(0) or DDR low power (1) mode.

12.4.29 Memory BIST execution control

12.4.30 BIST1_CFG_CTR register

The BIST1_CFG_CTR is an R/W register which configures and controls the internal core memory BIST execution at the functional speed. The register bit assignments is given in the next table.

Table 183. BIST1_CFG_CTR register bit assignments

BIST1_CFG_CTR Register			0x0F4
Bit	**Name**	**Reset Value**	**Description**
[31]	bist1_res_rst	1'h0	Reset status register result (BIST1_STS_RES): 1'b0: Disable reset status. 1'b1: Active reset status.
[30:29}	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[28]	bist1_rst	1'h0	Reset BIST engine collar: 1'b0: Disable reset. 1'b1: Active reset.

Table 183. BIST1_CFG_CTR register bit assignments (continued)

BIST1_CFG_CTR Register								0x0F4
Bit	**Name**	**Reset Value**	**Description**					
			Memory BIST interface command: command code and BIST engine actions are detailed in the next table					
			Memory Bist Command Table					
			Bist command					**Peripherals**
			Tm	Ret	Rbacktx	Iddq	Debug	
[27]	bist1-tm	1'h0						
[26]	bist1_debug	1'h0	0	0	1	0	0	Run BIST
[25]	bist1_ret	1'h0	0	0	0	0	1	Scan collar
[24]	bist1_iddq	1'h0	0	1	0	0	0	Read 0 retention test
			0	1	0	0	1	Read 1 retention test
			0	0	0	1	0	Iddq fill 0
			0	0	0	1	1	Iddq fill 0
			0	0	1	1	0	Stable mode
			0	0	0	0	0	Transparent mode
[23:15]	RFU	-	Rbact reserved command.					

Table 183. BIST1_CFG_CTR register bit assignments (continued)

<table>
<tr><td colspan="4">BIST1_CFG_CTR Register</td><td>0x0F4</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="2">Description</td></tr>
<tr><td rowspan="19">[14:00]</td><td rowspan="19">rbact1 (14:00)</td><td>15’h0</td><td colspan="2">Run BIST execution command (ref. Memory BIST command):
1’b0: Disable BIST command
1’b1: Run BIST command: memory BIST execution can be done either in single or group mode (ref. next table)</td></tr>
<tr><td colspan="3">Run BIST command table</td></tr>
<tr><td>Rbact</td><td>Memory cut</td><td>Peripherals</td></tr>
<tr><td>[14]</td><td>ST_DPHS_2048X32m8_Lb</td><td>Low speed shrd men</td></tr>
<tr><td>[13]</td><td>ST_DPHD_96X128m4_b (HWACC)</td><td>Application subssystem (HWACC)</td></tr>
<tr><td>[12]</td><td>ST_SPREG_384X12m4_L</td><td>JPEG HUFFENC</td></tr>
<tr><td>[11]</td><td>ST_SPREG_416X8m4_L</td><td>JPEG DHTMEM</td></tr>
<tr><td>[10]</td><td>ST_SPREG_256X8m4_L</td><td>JPEG QMEM</td></tr>
<tr><td>[09]</td><td>ST_SPREG_96X11m4_L</td><td>JPEG ZIGRAM_2</td></tr>
<tr><td>[08]</td><td>ST_SPREG_96X11m4_L</td><td>JPEG ZIGRAM_1</td></tr>
<tr><td>[07]</td><td>ST_DPHD_96X11m4_L</td><td>JPEG DCTRAM</td></tr>
<tr><td>[06]</td><td>ST_DPREG16X32m2_b</td><td>JPEG CTRL TX Fifo</td></tr>
<tr><td>[05]</td><td>ST_DPREG_16X32m2</td><td>JPEG CTRL RX Fifo</td></tr>
<tr><td>[04]</td><td>ST_DPREG_1024X35m4</td><td>Mac_rxfifo</td></tr>
<tr><td>[03]</td><td>ST_DPREG_512X35m4</td><td>Mac_txfifo</td></tr>
<tr><td>[02]</td><td>ST_DPHS_1024X36m8_L</td><td>Usb_device</td></tr>
<tr><td>[01]</td><td>RFU (not used)</td><td></td></tr>
<tr><td>[00]</td><td>ST_DPHD_256X32m4_L</td><td>Usb_host</td></tr>
</table>

12.4.31 BIST2_CFG_CTR register

The BIST2_CFG_CTR is an R/W register which configures and controls the RAS sub-group memory BIST execution at the functional speed. The register bit assignments is given in the next table.

Table 184. BIST2_CFG_CTR register bit assignments

<table>
<tr><td colspan="5">BIST2_CFG_CTR Register</td><td>0x0F8</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="3">Description</td></tr>
<tr><td>[31]</td><td>bist2_res_rst</td><td>1’h0</td><td colspan="3">Reset status register result (BIST2_STS_RES)
1’b0: Disable reset
1’b1: Active reset</td></tr>
<tr><td>[30:29]</td><td>RFU</td><td>-</td><td colspan="3">Reserved for future use (Write don’t care - Read return zeros).</td></tr>
<tr><td>[28]</td><td>bist2_rst</td><td>1’h0</td><td colspan="3">Reset BIST engine collar:
1’b0: Disable reset.
1’b1: Active reset</td></tr>
<tr><td>[27]
[26]
[25]
[24]</td><td>bist2_tm
bist2_debug
bist2_ret
bist2_iddq</td><td>1’h0
1’h0
1’h0
1’h0</td><td colspan="3">Memory BIST interface command: command code and BIST engine actions are detailed in the Memory BIST Command Table</td></tr>
<tr><td>[23:04]</td><td>RFU</td><td>-</td><td colspan="3"></td></tr>
<tr><td rowspan="6">[03:00]</td><td rowspan="6">rbact2(03:00)</td><td rowspan="6">4’h0</td><td colspan="3">Run BIST execution command (ref. Memory BIST command):
1’b0: Disable BIST command.
1’b1: Run BIST command: memory BIST execution can be done either in single or group mode (ref. next table)</td></tr>
<tr><td colspan="3">Run BIST command table</td></tr>
<tr><td>Rbact</td><td>Memory Cut</td><td>Peripherals</td></tr>
<tr><td>[03]</td><td>ST_SPREG_2048X32m8_Lb</td><td>RAS Local Data Buffer-0</td></tr>
<tr><td>[02]</td><td>ST_DPHS_1024X32m8_Lb</td><td>RAS Local Data Buffer - 1</td></tr>
<tr><td>[01]</td><td>ST_DPHD_128X8m4_L</td><td>RAS HWACC Data Buffer</td></tr>
<tr><td>[00]</td><td>ST_DPHD_96X128m4_b (RAS)</td><td>RAS HWACC Data Descriptor</td></tr>
</table>

12.4.32 BIST3_CFG_CTR register

The BIST3_CFG_CTR is an R/W register which configures and controls the RAS-2 sub-group memory BIST execution at the functional speed. The register bit assignments is given in the next table.

Table 185. BIST3_CFG_CTR register bit assignments

<table>
<tr><td colspan="5">BIST3_CFG_CTR Register</td><td>0x0FC</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="3">Description</td></tr>
<tr><td>[31]</td><td>bist3_res_rst</td><td>1’h0</td><td colspan="3">Reset status register result (BIST3_STS_RES)
1’b0: Disable reset
1’b1: Active reset.</td></tr>
<tr><td>[30:29]</td><td>RFU</td><td>-</td><td colspan="3">Reserved for future use (Write don’t care - Read return zeros).</td></tr>
<tr><td>[28]</td><td>bist3_rst</td><td>1’h0</td><td colspan="3">Reset BIST engine collar:
1’b0: Disable reset.
1’b1: Active reset.</td></tr>
<tr><td>[27]
[26]
[25]
[24]</td><td>bist3_tm
bist3_debug
bist3_ret
bist3_iddq</td><td>1’h0
1’h0
1’h0
1’h0</td><td colspan="3">Memory BIST interface command:command code and BIST engine actions are detailed in the Memory Bist Command Table</td></tr>
<tr><td>[23:03]</td><td>RFU</td><td>-</td><td colspan="3"></td></tr>
<tr><td rowspan="5">[02:00]</td><td rowspan="5">rbact3(02:00)</td><td rowspan="5">3’h0</td><td colspan="3">Run BIST execution command (ref. Memory BIST command):
1’b0: Disable BIST command.
1’b1: Run BIST command: memory BIST execution can be done either in single or group mode (ref. next table)</td></tr>
<tr><td colspan="3">Run BIST command table</td></tr>
<tr><td>Rbact</td><td>Memory Cut</td><td>Peripherals</td></tr>
<tr><td>[02]</td><td>ST_SPHDL_2048X8m16</td><td>RAS Local Data Buffer - 2</td></tr>
<tr><td>[01]</td><td>ST_SPREG_1024X32m4_Lb</td><td>RAS Local Data Buffer - 3_0</td></tr>
<tr><td>[00]</td><td>ST_SPREG_512X32m4_Lb</td><td>RAS Local Data Buffer - 3_1</td></tr>
</table>

12.4.33 BIST4_CFG_CTR register

The BIST4_CFG_CTR is an R/W register which configures and controls the ARM internal memory BIST execution at the functional speed. The register bit assignments is given in the next table.

Table 186. BIST4_CFG_CTR register bit assignments

BIST4_CFG_CTR Register				0x100
Bit	**Name**	**Reset Value**	**Description**	
[31]	bist4_res_rst	1'h0	Reset status register result (BIST4_STS_RES): 1'b0: Disable reset 1'b1: Active reset	
[30:29]	RFU	-	Reserved for future use (Write don't care - Read return zeros).	
[28]	bist4_rst	1'h0	Reset BIST engine collar. 1'b0: Disable reset. 1'b1: Active reset.	
[27] [26] [25] [24]	bist4_tm bist4_debug bist4_ret bist4_iddq	1'h0 1'h0 1'h0 1'h0	Memory BIST interface command:command code and BIST engine actions are detailed in the Memory BIST Command Table	
[23:08]	RFU	-		
[07:00]	rbact4	8'h0	Run BIST execution command (ref. Memory BIST command): 1'b0: Disable BIST command. 1'b1: Run BIST command execution.	
			RBACT	**Memory Cut**
			[00]	A926CM_rbact_dcache
			[01]	A926CM_rbact_dcache
			[02]	A926CM_rbact_dtag
			[03]	A926CM_rbact_dvalid
			[04]	A926CM_rbact_icache
			[05]	A926CM_rbact_itag
			[06]	A926CM_rbact_ivalid
			[07]	A926CM_rbact_mmu

12.4.34 BIST1_STS_RES register

The BIST1_STS_RES is an RO register which returns the functional BIST execution results for the internal core memory group. The register bit assignments is given in the next table.

Table 187. BIST1_STS_RES register bit assignments

<table>
<tr><td colspan="4">BIST1_STS_RES Register</td><td colspan="3">0x108</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="4">Description</td></tr>
<tr><td>[31]</td><td>bist1_end</td><td>-</td><td colspan="4">End memory BIST 1 execution:
1’b0: BIST execution pending.
1’b1: End memory BIST execution.</td></tr>
<tr><td>[30:24]</td><td>RFU</td><td>-</td><td colspan="4">Reserved for future use (Write don’t care - Read return zeros).</td></tr>
<tr><td>[23:15]</td><td>RFU</td><td>-</td><td colspan="4">Reserved for BIST bad extension field.</td></tr>
<tr><td rowspan="15">[14:00]</td><td rowspan="15">bbad1(14:00)</td><td rowspan="15">-</td><td colspan="4">BIST execution result (BIST bad signal status):
1’b0: BIST execution ok.
1’b1: BIST execution fails (ref. next table)</td></tr>
<tr><td colspan="4">Run BIST status table</td></tr>
<tr><td>Rbact</td><td colspan="2">Memory cut</td><td>Peripherals</td></tr>
<tr><td>[14]</td><td colspan="2">ST_DPHS_2048X32m8_Lb</td><td>Low speed shrd mem</td></tr>
<tr><td>[13]</td><td colspan="2">ST_DPHD_96X128m4_b (HWACC)</td><td>Application subsystem (HWACC)</td></tr>
<tr><td>[12]</td><td colspan="2">ST_SPREG_384X12m4_L</td><td>JPEG HUFFENC</td></tr>
<tr><td>[11]</td><td colspan="2">ST_SPREG_416X8m4_L</td><td>JPEG DHTMEM</td></tr>
<tr><td>[10]</td><td colspan="2">ST_SPREG_256X8m4_L</td><td>JPEG QMEM</td></tr>
<tr><td>[09]</td><td colspan="2">ST_SPREG_96X11m4_L</td><td>JPEG ZIGRAM_2</td></tr>
<tr><td>[08]</td><td colspan="2">ST_SPREG_96X11m4_L</td><td>JPEG ZIGRAM_1</td></tr>
<tr><td>[07]</td><td colspan="2">ST_DPHD_64X15m4_L</td><td>JPEG DCTRAM</td></tr>
<tr><td>[06]</td><td colspan="2">ST_DPREG_16X32m2_b</td><td>JPEG CTRL TX Fifo</td></tr>
<tr><td>[05]</td><td colspan="2">ST_DPREG_16X32m2</td><td>JPEG CTRL RX Fifo</td></tr>
<tr><td>[04]</td><td colspan="2">ST_DPREG_1024X35m4</td><td>Mac_rxfifo</td></tr>
<tr><td>[03]</td><td colspan="2">ST_DPREG_512X35m4</td><td>Mac_txfifo</td></tr>
<tr><td>[02]</td><td colspan="2">ST_DPHS_1024X36m8_L</td><td>Usb_device</td></tr>
<tr><td>[01]</td><td colspan="2">RFU (not used)</td><td></td></tr>
<tr><td>[00]</td><td colspan="2">ST_DPHD_256X32m4_L</td><td>Usb_host</td></tr>
</table>

12.4.35 BIST2_STS_RES register

The BIST2_STS_RES in an RO register which returns the functional BIST execution results for the RAS-1 memory sub-group. The register bit assignments is given in the next table.

Table 188. BIST2_STS_RES register bit assignments

BIST2_STS_RES Register			0x10C
Bit	**Name**	**Reset Value**	**Description**
[31]	bist2_end	-	End memory BIST2 execution: 1'b0: BIST execution pending 1'b1: End memory BIST execution.
[30:24]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[23:15]	RFU	-	Reserved for BIST bad extension field.

Table 188. BIST2_STS_RES register bit assignments (continued)

<table>
<tr><td colspan="4">BIST2_STS_RES Register</td><td>0x10C</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="3">Description</td></tr>
<tr><td rowspan="15">[14:00]</td><td rowspan="15">bbad2(14:00)</td><td rowspan="15">-</td><td colspan="3">BIST execution result (BIST bad signal status):
1’b0: BIST execution ok.
1’b1: BIST execution fails (ref. next table)</td></tr>
<tr><td colspan="3">Bist failure table</td></tr>
<tr><td>Bbad</td><td>Memory Cut</td><td>Peripherals</td></tr>
<tr><td>[14]</td><td>ST_SPREG_2048x32m8_Lb</td><td>Ras buf. Sp4Kx8_2</td></tr>
<tr><td>[13]</td><td>ST_SPREG_2048x32m8_Lb</td><td>Ras buf. Sp8Kx8_1</td></tr>
<tr><td>[12]</td><td>ST_DPHS_1024x32m8_Lb</td><td>Ras buf. Sp4Kx8_4</td></tr>
<tr><td>[11]</td><td>ST_DPHS_1024x32m8_Lb</td><td>Ras buf. Sp4Kx8_3</td></tr>
<tr><td>[10]</td><td>ST_DPHS_1024x32m8_Lb</td><td>Ras buf. Dp4Kx8_2</td></tr>
<tr><td>[09]</td><td>ST_DPHS_1024x32m8_Lb</td><td>Ras buf. Dp4Kx8_1</td></tr>
<tr><td>[08]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_8</td></tr>
<tr><td>[07]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_7</td></tr>
<tr><td>[06]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_6</td></tr>
<tr><td>[05]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_5</td></tr>
<tr><td>[04]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_4</td></tr>
<tr><td>[03]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_3</td></tr>
<tr><td>[02]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_2</td></tr>
<tr><td>[01]</td><td>ST_DPHD_128x8m4_L</td><td>Ras hwacc.SpDp128x8_1</td></tr>
<tr><td>[00]</td><td>ST_DPHD_96x128m4_b</td><td>Ras hwdescr.Dp96x128</td></tr>
</table>

12.4.36 BIST3_STS_RES register

The BIST3_STS_RES is an RO register which returns the functional BIST execution results for the RAS-2 memory sub-group. The register bit assignments is given in the next table.

Table 189. BIST3_STS_RES register bit assignments

<table>
<tr><td colspan="4">BIST3_STS_RES Register</td><td colspan="3">0x110</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="4">Description</td></tr>
<tr><td>[31]</td><td>bist3_end</td><td>-</td><td colspan="4">End memory BIST3 execution:
1’b0: BIST execution pending
1’b1: End memory BIST execution.</td></tr>
<tr><td>[30:14]</td><td>RFU</td><td>-</td><td colspan="4">Reserved for future use (Write don’t care - Read return zeros).</td></tr>
<tr><td rowspan="16">[13:00]</td><td rowspan="16">bbad3(13:00)</td><td rowspan="16">-</td><td colspan="4">BIST execution result (BIST bad signal status):
1’b0: Bist execution ok.
1’b1: Bist execution fails (ref. next table)</td></tr>
<tr><td colspan="4">Bist failure table</td></tr>
<tr><td>Bbad</td><td>Memory Cut</td><td colspan="2">Peripherals</td></tr>
<tr><td>[13]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_8</td></tr>
<tr><td>[12]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_7</td></tr>
<tr><td>[11]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_6</td></tr>
<tr><td>[10]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_5</td></tr>
<tr><td>[09]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_4</td></tr>
<tr><td>[08]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_3</td></tr>
<tr><td>[07]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_2</td></tr>
<tr><td>[06]</td><td>ST_SPHDL_2048 x8m16</td><td colspan="2">Ras buf. Sp2Kx8_1</td></tr>
<tr><td>[05]</td><td>ST_SPREG_1024 x32m4_Lb</td><td colspan="2">Ras buf. Sp4Kx8_2</td></tr>
<tr><td>[04]</td><td>ST_SPREG_1024 x32m4_Lb</td><td colspan="2">Ras buf. Sp4Kx8_1</td></tr>
<tr><td>[03]</td><td>ST_SPREG_512x 32m4_Lb</td><td colspan="2">Ras buf. Sp2Kx8_4</td></tr>
<tr><td>[02]</td><td>ST_SPREG_512x 32m4_Lb</td><td colspan="2">Ras buf. Sp2Kx8_3</td></tr>
<tr><td>[01]</td><td>ST_SPREG_512x 32m4_Lb</td><td colspan="2">Ras buf. Sp2Kx8_2</td></tr>
<tr><td>[00]</td><td>ST_SPREG_512x 32m4_Lb</td><td colspan="2">Ras buf. Sp2Kx8_1</td></tr>
</table>

12.4.37 BIST4_STS_RES register

The BIST4_STS_RES is an RO register which returns the functional BIST execution results for the ARM internal memory pool. The register bit assignments is given in the next table

Table 190. BIST4_STS_RES register bit assignments

BIST4_STS_RES Register			0x114
Bit	**Name**	**Reset Value**	**Description**
[31]	bist4_end	-	End memory BIST4 execution: 1'b0: BIST execution pending 1'b1: End memory BIST execution.
[30:24]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[23:14]	RFU	-	Reserved for BIST bad extension field

Table 190. BIST4_STS_RES register bit assignments (continued)

BIST4_STS_RES Register					0x114
Bit	**Name**	**Reset Value**	**Description**		
[13:00]	bbad4(13:00)	-	BIST execution result (BIST bad signal status): 1'b0: BIST execution ok 1'b1: BIST execution fails (ref. next table)		
			Bist failure table		
			Bbad	**Memory Cut**	**Peripherals**
			[13]	ARM_SPREG_1024x32m8_	Arm ddata Sp1Kx32_4
			[12]	ARM_SPREG_1024x32m8_b	Arm ddata Sp1Kx32_3
			[11]	ARM_SPREG_1024x32m8_b	Arm ddata Sp1Kx32_2
			[10]	ARM_SPREG_1024x32m8_b	Arm ddata Sp1Kx32_1
			[09]	ARM_SPREG_256x88m2_b	Arm dtag Sp256Kx22
			[08]	ARM_SPREG_32x24m2_L	Arm dcvalid Sp32x24
			[07]	ARM_SPREG_128x8m4_bL	Arm dcdirty Sp128x8
			[06]	ARM_SPREG_1024x32m8	Arm iicdata Sp1kx32_4
			[05]	ARM_SPREG_1024x32m8	Arm iicdata Sp1kx32_3
			[04]	ARM_SPREG_1024x32m8	Arm iicdata Sp1kx32_2
			[03]	ARM_SPREG_1024x32m8	Arm icdata Sp1Kx32_1
			[02]	ARM_SPREG_128x88m2_b	Arm itag Sp128x88
			[01]	ARM_SPREG_32x24m2_L	Arm ivalid Sp32x24
			[00]	ARM_SPREG_32x112m2_b	Arm mmu Sp32x112

12.4.38 BIST5_RSLT_REG register (Reserved)

The BIST5_RSLT_REG is an RO register which returns the functional BIST execution results for the ARM internal memory pool. The register bit assignments is given in the next table.

Table 191. BIST5_RSLT_REG register bit assignments

<table>
<tr><td colspan="4">BIST5_RSLT_REG Register</td><td colspan="3">0x118</td></tr>
<tr><td>Bit</td><td>Name</td><td>Reset Value</td><td colspan="4">Description</td></tr>
<tr><td>[31]</td><td>bist5_end</td><td>-</td><td colspan="4">end memory BIST5 execution:
1’b0: BIST execution pending
1’b1: end memory BIST execution.</td></tr>
<tr><td>[30:24]</td><td>RFU</td><td>-</td><td colspan="4">reserved for future use (write don’t care - read return zeros).</td></tr>
<tr><td>[23:20]</td><td>RFU</td><td>-</td><td colspan="4">reserved for BIST bad extension field</td></tr>
<tr><td rowspan="22">[19:00]</td><td rowspan="22">bbad4(19:00)</td><td rowspan="22">-</td><td colspan="4">BIST execution result (BIST bad signal status):
1’b0: BIST execution ok
1’b1: BIST execution fails (ref. next table)</td></tr>
<tr><td colspan="4">Bist failure table</td></tr>
<tr><td>Bbad</td><td>Memory Cut</td><td colspan="2">Peripherals</td></tr>
<tr><td>[19]</td><td>SPUHD1024x32m8_b</td><td colspan="2">Arm ddata Sp1Kx32_3</td></tr>
<tr><td>[18]</td><td>SPUHD1024x32m8_b</td><td colspan="2">Arm ddata Sp1Kx32_2</td></tr>
<tr><td>[17]</td><td>SPUHD1024x32m8_b</td><td colspan="2">Arm ddata Sp1Kx32_1</td></tr>
<tr><td>[16]</td><td>SPUHD1024x32m8_b</td><td colspan="2">Arm ddata Sp1Kx32_0</td></tr>
<tr><td>[15]</td><td>SP_64KUHD_256x22m4</td><td colspan="2">Arm dtag Sp256Kx22_3</td></tr>
<tr><td>[14]</td><td>SP_64KUHD_256x22m4</td><td colspan="2">Arm dtag Sp256Kx22_2</td></tr>
<tr><td>[13]</td><td>SP_64KUHD_256x22m4</td><td colspan="2">Arm dtag Sp256Kx22_1</td></tr>
<tr><td>[12]</td><td>SP_64KUHD_256x22m4</td><td colspan="2">Arm dtag Sp256Kx22_0</td></tr>
<tr><td>[11]</td><td>SPUHD1024x32m8</td><td colspan="2">Arm idata Sp1Kx32_3</td></tr>
<tr><td>[10]</td><td>SPUHD1024x32m8</td><td colspan="2">Arm idata Sp1Kx32_2</td></tr>
<tr><td>[09]</td><td>SPUHD1024x32m8</td><td colspan="2">Arm idata Sp1Kx32_1</td></tr>
<tr><td>[08]</td><td>SPUHD1024x32m8</td><td colspan="2">Arm idata Sp1Kx32_0</td></tr>
<tr><td>[07]</td><td>SP_64KUHD_128x22m4</td><td colspan="2">Arm itag Sp128Kx22_3</td></tr>
<tr><td>[06]</td><td>SP_64KUHD_128x22m4</td><td colspan="2">Arm itag Sp128Kx22_3</td></tr>
<tr><td>[05]</td><td>SP_64KUHD_128x22m4</td><td colspan="2">Arm itag Sp128Kx22_3</td></tr>
<tr><td>[04]</td><td>SP_64KUHD_128x22m4</td><td colspan="2">Arm itag Sp128Kx22_3</td></tr>
<tr><td>[03]</td><td>SP_64KUHD_32x24m2</td><td colspan="2">Arm dvalid Sp32Kx24</td></tr>
<tr><td>[02]</td><td>SP_64KUHD_128x8m2_b</td><td colspan="2">Arm ddirty Sp128Kx8</td></tr>
<tr><td>[01]</td><td>SP_64KUHD_32x24m2</td><td colspan="2">Arm ivalid Sp32Kx24</td></tr>
<tr><td>[00]</td><td>SP_64KUHD_32x112m2_b</td><td colspan="2">ARM MMU Sp32x112</td></tr>
</table>

12.4.39 Diagnostic functionality

12.4.40 SYSERR_CFG_CTR register

The SYSERR_CFG_CTR is an R/W register which configures the SoC internal error detections. The register bit assignments is detailed in the next table.

Table 192. SYSERR_CFG_CTR register bit assignments

SYSERR_CFG_CTR Register			0x11C
Bit	**Name**	**Reset Value**	**Description**
[31:29]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[28]	DMA_err	1'h0	DMA transfer error (RO); detection enable through 'DMA_err_enb' register field set high: 1'b0: No error pending. 1'b1: Active DMA transfer error; asserted when DMA master transaction receives an error response type (for further detail ref. DMA Chapter)
[27]	Mem_err	1'h0	Memory transaction error (RO); detection enable through 'mem_err_enb' register field set high: 1'b0: No error pending. 1'b1: Memory transfer error; asserted from memory controller when one of the following error event is active: A single access outside the defined PHYSICAL memory space. Multiple accesses outside the defined PHYSI-CAL memory space. DRAM initialization completes (no error event). Address cross page boundary. DLL unlock event.
[26] [25] [24]	usbh2_err usbh1_err usbdv_err	1'h0 1'h0 1'h0	USB2 PHY receiver error (RO); detection enable through 'usb_err_enb' register field set high: 1'b0: No error pending. 1'b1: USB2 PHY 'rxerror'; asserted when one of the following error events is active: Bit stuff errors during FS receive operation. Elasticity buffer overrun/under run. Alignment error; EOP not on a byte boundary.
[23]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[22]	arm1_wdg_err	1'h0	Processors watch dog timeout error (RO); detection enable through 'wdg_err_enb' bit set high: 1'b0: No error pending. 1'b1: Active watches dog timeout error; asserted when the arms watch dog timer expires (the ARM watch dog functionality is supplied from Basic subsystem Timer1).

Table 192. SYSERR_CFG_CTR register bit assignments (continued)

SYSERR_CFG_CTR Register			0x11C
Bit	**Name**	**Reset Value**	**Description**
[21:16]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[15] [14] [13] [12]	mem_dll_err usb_pll_err sys_pll2_err sys_pll1_er	1'h0 1'h0 1'h0 1'h0	PLL/DLL unlock error (RO); detection enable through 'pll_err_enb' register field set high: 1'b0: No error pending. 1'b1: Pll/Dll unlock error.
[11]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[10]	DMA_err_enb	1'h0	Enable DMA transfer error interrupt detection: 1'b0: Disable error detection. 1'b1: Enable error detection.
[09]	mem_err_enb	1'h0	Enable Memory transfer error interrupt detection: 1'b0: Disable error detection. 1'b1: Enable error detection.
[08]	usb_err_enb	1'h0	Enable USB2 PHY receive error interrupt detection: 1'b0: Disable error detection. 1'b1: Enable error detection.
[07]	RFU	-	
[06]	wdg_err_enb	1'h0	Enable Watch dog timeout error interrupt detection: 1'b0: Disable error detection. 1'b1: Enable error detection.
[05]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[04]	pll_err_enb	1'h0	Enable PLL/DLL unlock error interrupt detection: 1'b0: Disable error detection. 1'b1: Enable error detection.
[03]	RFU	-	Reserved for future use (Write don't care - Read return zeros).
[02]	int_error	1'h0	SYS_ERROR interrupt request (RO): enabled when 'int_error_enb' is high: 1'b0: No error interrupts pending. 1'b1: Active error interrupt: this bit is the logic or of all enabled error interrupt events.
[01]	int_error_rst	1'h0	Reset error interrupt request: 1'b0: No action. 1'b1: Reset all active error interrupt requests.
[00]	int_error_enb	1'h0	Enable SYS_ERROR interrupt event: 1'b0: Disable error interrupt assertion. 1'b1: Enable error interrupt assertion.

12.4.41 USB_TUN_PRM register

To enable adjusting various USB 2.0 specification-related characteristics, the USB 2.0 nanoPHY provides top level parameter override bits. The USB 2.0 nanoPHY is designed to a default setting of these bits, and you are not expected to have to change these bits from their default setting.

These override bits are not intended for per-part centering or dynamic centering. However, there might be circumstances that require bit settings that are different than the default. If a change is required, statically set these bits to the same value for all product parts. Statistical analysis of the USB 2.0 nanoPHY's silicon characterization will determine whether any of these bits require a different setting other than the default.

Table 193. USB_TUN_PRM register bit assignments

USB0/1/2_TUN_PRM			0x120/4/8
Bit	**Name**	**Reset Value**	**Description**
[31:18]	RFU	-	Reserved for future use (Write don't care - Read return zeros)
[17:15]	COMPDISTUNE[02:00]	3'h4	COMPDISTUNE
[14:12]	SQRXTUNE[2:0]	3'h3	SQRXTUNE
[11:08]	TXFSLSTUNE[3:0]	4'h3	TXFSLSTUNE
[07:04]	TXVREFTUNE[3:0]	4'h8	TXVREFTUNE
[03]	TXPREEMPHASISTUNE	1'h0	TXPREEMPHASISTUNE
[02:01]	TXHSXVTUNE[1:0]	2'h3	TXHSXVTUNE
[00]	TXRISETUNE	1'h0	TXRISETUNE

12.4.42 PLGPIOn_PAD_PRG Registers

These five registers are used to program FAST IO pad's Drive strength, Pull up, pull down status and Slew parameters. Depending upon the value of the above parameters following tables govern their behaviour.

Table 194. Drive selection

DRV[0]	DRV[1]	OUTPUT DRIVE
0	0	4mA
0	1	6mA
1	0	8mA
1	1	12mA

Table 195. Pull Up and Pull Down Selection

PUP	PDN	Comment
0	0	Not Allowed
0	1	Pull-Up Activated
1	0	Pull-Down Activated
1	1	Pull-Up/Pull-Down deactivated

Table 196. Slew selection

SL	Slew level
0	Nominal
1	Fast

In general for programming purposes the slew and drive strength of Four pads are shared by a bit as specified in table below with some exception as specified in the table. For Pull Up and Pull down same rule follows but with some exception as specified below in the tables.

Table 197. PLGPIO0_PAD_PRG register bit assignments

PLGPIO0_PAD_PRG			0x130
Bit	**Name**	**Reset Value**	**Description**
[31]	PDN_UART	1'h0	Pull down control for UART (Pads 2,3)
[30]	PUP_UART	1'h1	Pull up control for UART (Pads 2,3)
[29]	PDN_5	1'h1	Pull down control for pads 20,21,22,23
[28]	PUP_5	1'h0	Pull up control for pads 20,21,22,23
[27:26]	DRV_5[1:0]	2'h0	Drive strength control for pads 20,21,22,23
[25]	SLEW_5	1'h0	Slew control for pads 20,21,22,23
[24]	PDN_4	1'h1	Pull down control for pads 16,17,18,19
[23]	PUP_4	1'h0	Pull up control for pads 16,17,18,19
[22:21]	DRV[1:0]	2'h0	Drive strength control for pads 16,17,18,19
[20]	SLEW_4	1'h0	Slew control for pads 16,17,18,19
[19]	PDN_3	1'h1	Pull down control for pads 12,13,14,15
[18]	PUP_3	1'h0	Pull up control for pads 12,13,14,15
[17:16]	DRV_3[1:0]	2'h0	Drive strength control for pads 12,13,14,15
[15]	SLEW_3	1'h0	Slew control for pads 12,13,14,15
[14]	PDN_2	1'h1	Pull down control for pads 8,9
[13]	PUP_2	1'h0	Pull Up control for pads 8,9
[12:11]	DRV_2[1:0]	2'h0	Drive Strength control for pads 8,9,10,11
[10]	SLEW_2	1'h0	Slew control for pads 8,9,10,11
[09]	PDN_1	1'h1	Pull Down control for pads 6,7

Table 197. PLGPIO0_PAD_PRG register bit assignments (continued)

PLGPIO0_PAD_PRG			0x130
Bit	**Name**	**Reset Value**	**Description**
[08]	PUP_1	1'h0	Pull up control for pads 6,7
[07:06]	DRV_1[1:0]	2'h0	Drive strength control for pads 4,5,6,7
[05]	SLEW_1	1'h0	Slew control for pads 4,5,6,7
[04]	PDN_0	1'h1	Pull Down Control for pads 0,1
[03]	PUP_0	1'h0	Pull Up control for pads 0,1
[02:01]	DRV_0[1:0]	2'h0	Drive strength control for pads 0,1,2,3
[00]	SLEW_0	1'h0	Slew control for pads 0,1,2,3

Table 198. PLGPIO1_PAD_PRG register bit assignments

PLGPIO1_PAD_PRG			0x134
Bit	**Name**	**Reset Value**	**Description**
[31]	PDN_I2C	1'h1	Pull down control for I2C (Pads 4,5)
[30]	PUP_12C	1'h0	Pull up control for I2C (Pads 4,5)
[29]	PDN_11	1'h1	Pull down control for pads 45, 46, 47
[28]	PUP_11	1'h0	Pull up control for pads 45,46,47
[27:26]	DRV_11[1:0]	2'h0	Drive strength control for pads 44,45,46,47
[25]	SLEW_11	1'h0	Slew control for pads 44,45,46,47
[24]	PDN_10	1'h0	Pull down control for pads 40,41,42,43,44
[23]	PUP_10	1'h1	Pull up control for pads 40,41,42,43,44
[22:21]	DRV_10[1:0]	2'h0	Drive strength control for pads 40,41,42,43
[20]	SLEW_10	1'h0	Slew control for pads 40,41,42,43
[19]	PDN_9	1'h0	Pull down control for pads 37,38,39
[18]	PUP_9	1'h1	Pull up control for pads 37,38,39
[17:16]	DRV_9[1:0]	2'h0	Drive strength control for pads 36,37,38,39
[15]	SLEW_9	1'h0	Slew control for pads 36,37,38,39
[14]	PDN_8	1'h1	Pull down control for pads 32,33,34,35,36
[13]	PUP_8	1'h0	Pull up control for pads 32,33,34,35,36
[12:11]	DRV_8[1:0]	2'h0	Drive strength control for pads 32,33,34,35
[10]	SLEW_8	1'h0	Slew control for pads 32,33,34,35
[09]	PDN_7	1'h1	Pull down control for pads 28,29,30,31
[08]	PUP_7	1'h0	Pull up control for pads 28,29,30,31
[07:06]	DRV_7[1:0]	2'h0	Drive strength control for pads 28,29,30,31
[05]	SLEW_7	1'h0	Slew control for pads 28,29,30,31

Table 198. PLGPIO1_PAD_PRG register bit assignments (continued)

PLGPIO1_PAD_PRG			0x134
Bit	**Name**	**Reset Value**	**Description**
[04]	PDN_6	1'h1	Pull down control for pads 24,25,26,27
[03]	PUP_6	1'h0	Pull Up control for pads 24,25,26,27
[02:01]	DRV_6[1:0]	2'h0	Drive Strength control for pads 24,25,26,27
[00]	SLEW_6	1'h0	Slew control for pads 24, 25,26,27

Table 199. PLGPIO2_PAD_PRG register bit assignments

PLGPIO2_PAD_PRG			0x138
Bit	**Name**	**Reset Value**	**Description**
[31]	PDN_ETHERNET	1'h0	Pull down control for Ethernet (Pads 10,11)
[30]	PUP_ETHERNET	1'h1	Pull up control for Ethernet (Pads 10,11)
[29]	PDN_17	1'h1	Pull down control for pads 68,69,70,71
[28]	PUP_17	1'h0	Pull up control for pads 68,69,70,71
[27:26]	DRV_17[1:0]	2'h0	Drive strength control for pads 68,69,70,71
[25]	SLEW_17	1'h0	Slew control for pads 68,69,70,71
[24]	PDN_16	1'h1	Pull down control for pads 64,65,66,67
[23]	PUP_16	1'h0	Pull up control for pads 64,65,66,67
[22:21]	DRV_16[:0]	2'h0	Drive strength control for pads 64,65,66,67
[20]	SLEW_16	1'h0	Slew control for pads 64,65,66,67
[19]	PDN_15	1'h1	Pull down control for pads 60,61,62,63
[18]	PUP_15	1'h0	Pull up control for pads 60,61,62,63
[17:16]	DRV_15[1:0]	2'h0	Drive strength control for pads 60,61,62,63
[15]	SLEW_15	1'h0	Slew control for pads 60,61,62,63
[14]	PDN_14	1'h1	Pull down control for pads 56,57,58,59
[13]	PUP_14	1'h0	Pull up control for pads 56,57,58,59
[12:11]	DRV_14[1:0]	2'h0	Drive strength control for pads 56,57,58,59
[10]	SLEW_14	1'h0	Slew control for pads 56,57,58,59
[09]	PDN_13	1'h1	Pull down control for pads 52,53,54,55
[08]	PUP_13	1'h0	Pull up control for pads 52,53,54,55
[07:06]	DRV_13[1:0]	2'h0	Drive strength control for pads 52,53,54,55
[05]	SLEW_13	1'h0	Slew control for pads 52,53,54,55
[04]	PDN_12	1'h1	Pull Down control for pads 48,49,50,51
[03]	PUP_12	1'h0	Pull Up control for pads 48,49,50,51

Table 199. PLGPIO2_PAD_PRG register bit assignments

PLGPIO2_PAD_PRG			0x138
Bit	**Name**	**Reset Value**	**Description**
[02:01]	DRV_12[1:0]	2'h0	Drive Strength control for pads48,49,50,51
[00]	SLEW_12	1'h0	Slew control for pads 48,49,50,51

Table 200. PLGPIO3_PAD_PRG register bit assignments

PLGPIO3_PAD_PRG			0x13C
Bit	**Name**	**Reset Value**	**Description**
[31:30]	RFU	-	Reserved for future use (Read return zeros)
[29]	PDN_23	1'h1	Pull down control for pads 92,93,94,95
[28]	PUP_23	1'h0	Pull up control for pads 92,93,94,95
[27:26]	DRV_23[1:0]	2'h0	Drive strength control for pads 92,93,94,95
[25]	SLEW_23	1'h0	Slew control for pads 92,93,94,95
[24]	PDN_22	1'h1	Pull down control for pads 88,89,90,91
[23]	PUP_22	1'h0	Pull up control for pads 88,89,90,91
[22:21]	DRV_22[1:0]	2'h0	Drive strength control for pads 88,89,90,91
[20]	SLEW_22	1'h0	Slew control for pads 88,89,90,91
[19]	PDN_21	1'h1	Pull down control for pads 84,85,86,87
[18]	PUP_21	1'h0	Pull up control for pads 84,85,86,87
[17:16]	DRV_21[1:0]	2'h0	Drive strength control for pads 84,85,86,87
[15]	SLEW_21	1'h0	Slew control for pads 84,85,86,87
[14]	PDN_20	1'h1	Pull down control for pads 80,81,82,83
[13]	PUP_20	1'h0	Pull up control for pads 80,81,82,83
[12:11]	DRV_20[1:0]	2'h0	Drive strength control for pads 80,81,82,83
[10]	SLEW_20	1'h0	Slew control for pads 80,81,82,83
[09]	PDN_19	1'h1	Pull down control for pads 76,77,78,79
[08]	PUP_19	1'h0	Pull up control for pads 76,77,78,79
[07:06]	DRV_19[1:0]	2'h0	Drive strength control for pads 76,77,78,79
[05]	SLEW_19	1'h0	Slew control for pads 76,77,78,79
[04]	PDN_18	1'h1	Pull Down control for pads 72,73,74,75
[03]	PUP_18	1'h0	Pull Up control for pads 72,73,74,75
[02:01]	DRV_18[1:0]	2'h0	Drive Strength control for pads 72,73,74,75
[00]	SLEW_18	1'h0	Slew control for pads 72,73,74,75

Table 201. PLGPIO4_PAD_PRG register bit assignments

PLGPIO4_PAD_PRG			0x140
Bit	**Name**	**Reset Value**	**Description**
[31:25]	RFU	-	Reserved for future use (Read return zeros)
[24]	PDN_CLK4	1'h1	Pull down control for pads CLK4
[23]	PUP_CLK4	1'h0	Pull up control for pads CLK4
[22:21]	DRV_CLK4[1:0]	2'h0	Drive strength control for pads CLK4
[20]	SLEW_CLK4	1'h0	Slew control for pads CLK4
[19]	PDN_CLK3	1'h1	Pull down control for pads CLK3
[18]	PUP_CLK3	1'h0	Pull up control for pads CLK3
[17:16]	DRV_CLK3[1:0]	2'h0	Drive strength control for pads CLK3
[15]	SLEW_CLK3	1'h0	Slew control for pads CLK3
[14]	PDN_CLK2	1'h1	Pull down control for pads CLK2
[13]	PUP_CLK2	1'h0	Pull up control for pads CLK2
[12:11]	DRV_CLK2[1:0]	2'h0	Drive strength control for pads CLK2
[10]	SLEW_CLK2	1'h0	Slew control for pads CLK2
[09]	PDN_CLK1	1'h1	Pull down control for pads CLK1
[08]	PUP_CLK1	1'h0	Pull up control for pads CLK1
[07:06]	DRV_CLK1[1:0]	2'h0	Drive strength control for pads CLK1
[05]	SLEW_CLK1	1'h0	Slew control for pads CLK1 & pads 96,97
[04]	PDN_24	1'h1	Pull Down control for pads 96,97
[03]	PUP_24	1'h0	Pull up control for pads 96,97
[02:01]	DRV_24[1:0]	2'h0	Drive Strength control for pads 96,97
[00]	RFU		Reserved for future purpose

12.5 Miscellaneous register global space

12.5.1 Overview

The global space controls the following functionalities:

- General purpose input signals:
  - General status/command interface received from programmable logic.
  - Registered input mail box data.
- General purpose output signals:
  - General output command interface.
  - Programmable logic configuration extension.
  - Registered output mail box data.

12.5.2 Miscellaneous register global space address map

Next table shows the miscellaneous global register map.

Table 202. Miscellaneous global space registers overview

Miscellaneous Global Space Register Map		Base Address: 0xFCA8.0000	
Register Name	**Alias-1 Offset 0x0.8000**	**Alias-2 Offset 0x1.8000**	**Type**
	Register Displacement (single region)		
RAS_GPP1_IN	0x00		R/W
RAS_GPP2_IN	**0x04**		R/W
RAS_GPP1_OUT	0x08		R/W
RAS_GPP2_OUT	0x0C		R/W
RAS_GPP_EXT_IN	0x10		R/W
RAS_GPP_EXT_OUT	0x14		R/W
Reserved	0x18		R/W
Reserved	0x1C		R/W

12.5.3 RAS1/2_GPP_INP register

The RAS1/2_GPP_INP is a group of RO general purpose input registers used to pass different kind of information from the reconfigurable logic array toward the internal core logic. The register bit assignments is detailed in the next two tables.

Table 203. RAS_GPP1_IN register bit assignments

RAS_GPP1_IN Register			0x000
Bit	**Name**	**Reset Value**	**Description**
[31:00]	gpp1_in[31:00]	-	General purpose input register (RO) which return the current value of the programmable logic GPP_INP (31:00) signals.

Table 204. RAS_GPP2_IN register bit assignments

RAS_GPP2_IN Register			0x004
Bit	**Name**	**Reset Value**	**Description**
[31:00]	gpp2_in[31:00]	-	General purpose input register (RO) which return the current value of the programmable logic GPP_INP (63:32) signals.

Table 205. RAS_GPP_EXT_IN register bit assignments

RAS_GPP_EXT_IN Register			0x010
Bit	**Name**	**Reset Value**	**Description**
[07:00]	Gpp_ext_in[07:00]	-	General purpose input register (RO) which return the current value of the programmable logic signals.

12.5.4 RAS1/2_GPP_OUT register

The RAS1/2_GPP_OUT are a group of R/W general purpose output registers used to pass different kind of data/command from the internal core logic to reconfigurable logic array. The register bit assignments is given in the next two tables.

Table 206. RAS_GPP1_OUT register bit assignments

RAS_GPP1_OUT Register			0x008
Bit	**Name**	**Reset Value**	**Description**
[31:00]	gpp1_out[31:00]	32'h0	General purpose output register direct controls the programmable logic GPP_OUT(31:00) signals 1'b0: Force the corresponding bit signal low. 1'b1: Force the corresponding bit signal high. General purpose output command field.

Table 207. RAS_GPP2_OUT register bit assignments

RAS_GPP2_OUT Register			0x00C
Bit	**Name**	**Reset Value**	**Description**
[31:00]	gpp2_out[31:00]	32'h0	General purpose output register direct controls the programmable logic GPP_OUT(63:32) signals 1'b0: Force the corresponding bit signal low. 1'b1: Force the corresponding bit signal high. General purpose output command field.

Table 208. RAS_GPP_EXT_OUT register bit assignments

RAS_GPP_EXT_OUT Register			0x014
Bit	**Name**	**Reset Value**	**Description**
[07:00]	Gpp_ext_out[07:00]	7'h0	General purpose output register direct controls the programmable logic GPP_EXT_OUT(07:00) signals 1'b0: Force the corresponding bit signal low. 1'b1: Force the corresponding bit signal high. General purpose output command field.

13 LS_Synchronous serial peripheral (SSP)

13.1 Overview

Within its low speed connectivity, the device provides one ARM PrimeCell® synchronous serial port (SSP) block that offers a master or slave interface to enables synchronous serial communication with slave or master peripherals

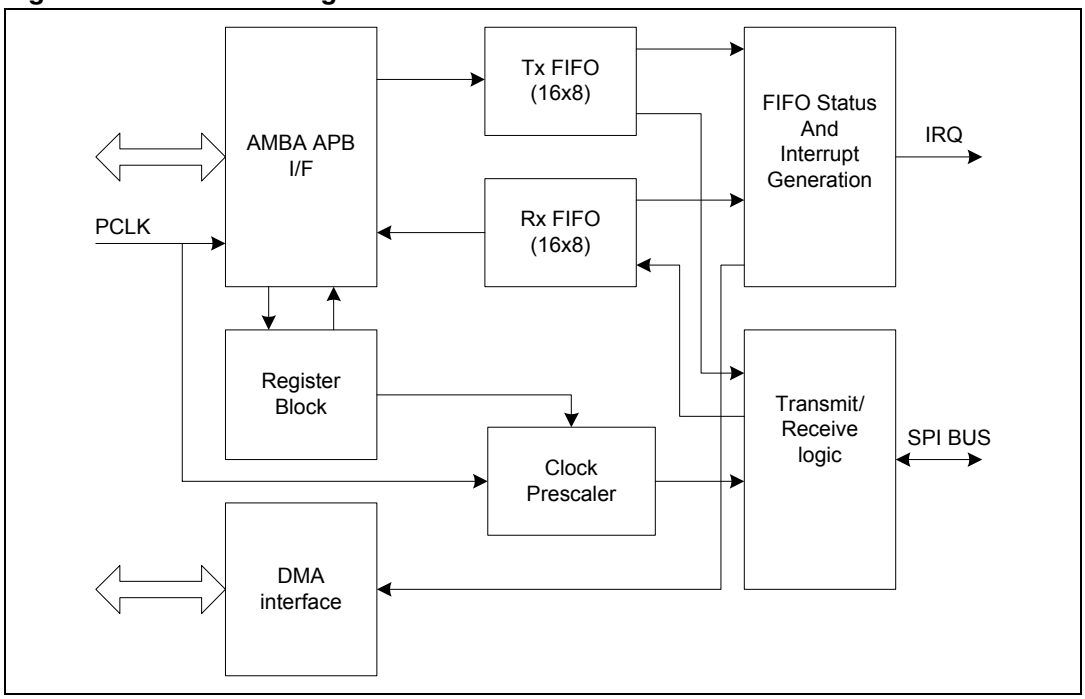
Main features of the SSP are:

- Master or slave operation.
- Programmable clock bit rate and prescale.
- Separate transmit and receive first-in, first-out memory buffers, 16 bits wide, 8 locations deep.
- Programmable choice of interface operation, SPI, Microwire, or TI synchronous serial.
- Programmable data frame size from 4 to 16 bits.
- Independent masking of transmit FIFO, receive FIFO, and receive overrun interrupts.
- Internal loopback test mode available.
- Support for direct memory access (DMA).

13.2 Block diagram

Figure 22 shows the block diagram of SPI controller.

Figure 22. SPI block diagram



13.3 Signal interfaces

The SSP directly interfaces with the signals summarized in *Table 209*

Table 209. SSP signal interface

Group	Signal name	Direction	Size (bit)	Description
Global	CLK	Input	1	Main SSP clock input.
	nRST	Input	1	SSP reset signal.
Interrupts	TXINTR	Output	1	Transmit FIFO service request interrupt.
	RXINTR	Output	1	Receive FIFO service request interrupt.
	RORINTR	Output	1	Receive overrun interrupt.
	RTINTR	Output	1	Receive timeout interrupt.
	INTR	Output	1	SSP interrupt. This interrupt is an OR of the four individual interrupts TXINTR, RXINTR, RORINTR and RTINTR.
DMA Interface	TXDMASREQ	Output	1	Transmit DMA single request.
	TXDMABREQ	Output	1	Transmit DMA burst request.
	RXDMASREQ	Output	1	Receive DMA single request.
	RXDMABREQ	Output	1	Receive DMA burst request.
	TXDMACLR	Input	1	DMA request clear. Asserted by DMA controller to clear the transmit request signal.
	RXDMACLR	Input	1	DMA request clear. Asserted by DMA controller to clear the receive request signal.
PAD control	FSSOUT	Output		SSP frame or slave select (master mode).
	CLKOUT	Output		SSP clock output (Master mode).
	TXD	Output		Transmit data output.
	OE	Output		Output enable signal.
	FSSIN	Input		SSP frame input (slave mode).
	CLKIN	Input		SSP clock input (slave mode).
	RXD	Input		Receive data input.
APB Slave	-	Input/Output	-	See AMBA Specification.

13.4 Main functions description

13.4.1 APB slave interface

The AMBA APB interface generates read and write decodes for accesses to status and control registers, and transmit and receive FIFO memories.

The AMBA APB is a local secondary bus that provides a low-power extension to the higher bandwidth AMBA advanced high-performance bus (AHB) within the AMBA system

hierarchy. The AMBA APB groups narrow-bus peripherals to avoid loading the system bus and provides an interface using memory-mapped registers, which are accessed under programmed control.

13.4.2 Register block

The register block stores data written or to be read across the AMBA APB interface.

13.4.3 Clock prescaler

When configured as a master, an internal prescaler, comprising two free-running re-loadable serially linked counters, is used to provide the serial output clock CLKOUT.

You can program the clock prescaler, through the SSPCPSR register, to divide CLK by a factor of 2 to 254 in steps of two. By not utilizing the least significant bit of the SSPCPSR register, division by an odd number is not possible and this ensures a symmetrical (equal mark space ratio) clock is generated.

The output of the prescaler is further divided by a factor of 1 to 256, through the programming of the SSPCR0 control register, to give the final master output clock CLKOUT.

13.4.4 Transmit FIFO

The common transmit FIFO is a 16 bit wide, 8-locations deep, first-in, first-out memory buffer. CPU data written across the AMBA APB interface are stored in the buffer until read out by the transmit logic.

When configured as a master or a slave parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master respectively, through the SSPTXD pin.

13.4.5 Receive FIFO

The common receive FIFO is a 16 bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface are stored in the buffer until read out by the CPU across the AMBA APB interface.

When configured as a master or slave, serial data received through the `SSPRXD` pin is registered prior to parallel loading into the attached slave or master receive FIFO respectively.

13.4.6 Transmit and receive logic

When configured as a master, the clock to the attached slaves is derived from a divided down version of CLK through the prescaler operations described previously. The master transmit logic successively reads a value from its transmit FIFO and performs parallel to serial conversion on it. Then the serial data stream and frame control signal, synchronized to CLKOUT, are output through the TXD pin to the attached slaves. The master receive logic performs serial to parallel conversion on the incoming synchronous SSPRXD data stream, extracting and storing values into its receive FIFO, for subsequent reading through the APB interface.

When configured as a slave, the SSPCLKIN clock is provided by an attached master and used to time its transmission and reception sequences. The slave transmit logic, under control of the master clock, successively reads a value from its transmit FIFO, performs

parallel to serial conversion, then output the serial data stream and frame control signal through the slave SSPTXD pin. The slave receive logic performs serial to parallel conversion on the incoming SSPRXD data stream, extracting and storing values into its receive FIFO, for subsequent reading through the APB interface.

13.4.7 Interrupt generation logic

The PrimeCell SSP generates four individual maskable, active HIGH interrupts.

A combined interrupt output is also generated as an OR function of the individual interrupt requests.

You can use the single combined interrupt with a system interrupt controller that provides another level of masking on a per-peripheral basis. This allows use of modular device drivers that always know where to find the interrupt source control register bits.

The individual interrupt requests could also be used with a system interrupt controller that provides masking for the outputs of each peripheral. In this way, a global interrupt controller service routine would be able to read the entire set of sources from one wide register in the system interrupt controller. This is attractive where the time to read from the peripheral registers is significant compared to the CPU clock speed in a real-time system.

The peripheral supports both the above methods.

The transmit and receive dynamic data-flow interrupts, TXINTR and RXINTR, are separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels.

13.4.8 DMA interface

This block manages the DMA interface. It can work in single transfer mode or in burst transfer mode.

Refer to the *Chapter 19: BS_DMA controller* to have more detail on this Interface.

13.4.9 Synchronizing registers and logic

The SSP supports both asynchronous and synchronous operation of the clocks, PCLK and SSPCLK. Synchronization registers and hand shaking logic have been implemented, and are active at all times. This has a minimal impact on performance or area. Synchronization of control signals is performed on both directions of data flow, which is from the PCLK to the SSPCLK domain and from the SSPCLK to the PCLK domain.

13.5 SSP operation

In the following sections are described the operation of the SSP block.

13.5.1 Configuring the SSP

Following reset, the SSP logic is disabled and must be configured when in this state.

Control registers SSPCR0 and SSPCR1 need to be programmed to configure the peripheral as a master or slave operating under one of the following protocols:

- Motorola SPI
- Texas Instruments SSI
- National Semiconductor.

The bit rate, derived from the APB clock (PCLK), requires the programming of the clock prescale register SSPCPSR. (Refer to the miscellaneous registers description for the PCLK frequency).

Defining the chip select

Four chip select lines are available to verify the real availability of the external signal but only one can be operational at time. The selection of the active one is done using the upper two GPIO lines. The FSSOUT will be connected to the external CSx according to the *Table 210*

Table 210. External CS selection

GPIO[7]	GPIO[6]	CSx
0	0	CS1
0	1	CS2
1	0	CS3
1	1	CS4

13.5.2 Enable SSP operation

You can either prime the transmit FIFO, by writing up to eight 16 bit values when the SSP is disabled, or allow the transmit FIFO service request to interrupt the CPU. Once enabled, transmission or reception of data begins on the transmit (SSPTXD) and receive (SSPRXD) pins.

Clock ratios

There is a constraint on the ratio of the frequencies of PCLK to SSPCLK. The frequency of SSPCLK must be less than or equal to that of PCLK. This ensures that control signals from the SSPCLK domain to the PCLK domain are certain to get synchronized before one frame duration:

```
FSSPCLK <= FPCLK
```

In the slave mode of operation, the SSPCLKIN signal from the external master is double synchronized and then delayed to detect an edge. It takes three SSPCLKs to detect an edge on SSPCLKIN. SSPTXD has less setup time to the falling edge of SSPCLKIN on which the master is sampling the line. The setup and hold times on SSPRXD with reference to SSPCLKIN must be more conservative to ensure that it is at the right value when the actual sampling occurs within the SSPMS. To ensure correct device operation, SSPCLK must be at least 12 times faster than the maximum expected frequency of SSPCLKIN.

The frequency selected for SSPCLK must accommodate the desired range of bit clock rates. The ratio of minimum SSPCLK frequency to SSPCLKOUT maximum frequency in the case of the slave mode is 12 and for the master mode it is two.

To generate a maximum bit rate of 1.8432 Mbps in the Master mode, the frequency of SSPCLK must be at least 3.6864 MHz. With an SSPCLK frequency of 3.6864 MHz, the SSPCPSR register has to be programmed with a value of two and the SCR[7:0] field in the SSPCR0 register needs to be programmed as zero.

To work with a maximum bit rate of 1.8432 Mbps in the slave mode, the frequency of SSPCLK must be at least 22.12 MHz. With an SSPCLK frequency of 22.12 MHz, the SSPCPSR register can be programmed with a value of 12 and the SCR[7:0] field in the SSPCR0 register can be programmed as zero. Similarly the ratio of SSPCLK maximum frequency to SSPCLKOUT minimum frequency is 254 x 256.

The minimum frequency of SSPCLK is governed by the following equations, both of which have to be satisfied:

- FSSPCLK(min) => 2 x FSSPCLKOUT(max) [for master mode]
- FSSPCLK(min) => 12 x FSSPCLKIN(max) [for slave mode].

The maximum frequency of SSPCLK is governed by the following equations, both of which have to be satisfied:

- FSSPCLK(max) <= 254 x 256 x FSSPCLKOUT(min) [for master mode]
- FSSPCLK(max) <= 254 x 256 x FSSPCLKIN(min) [for slave mode]

13.5.3 Programming the SSPCR0 control register

The SSPCR0 register is used to:

- Program the serial clock rate
- Select one of the three protocols
- Select the data word size (where applicable).

The Serial Clock Rate (SCR) value, in conjunction with the SSPCPSR clock prescale divisor value (CPSDVSR), is used to derive the SSP transmit and receive bit rate from the external SSPCLK.

The frame format is programmed through the FRF bits and the data word size through the DSS bits.

Bit phase and polarity, applicable to Motorola SPI format only, are programmed through the SPH and SPO bits.

13.5.4 Programming the SSPCR1 control register

The SSPCR1 register is used to:

- Select master or slave mode
- Enable a loop back test feature
- Enable the SSP peripheral.

To configure the SSP as a master, clear the SSPCR1 register master or slave selection bit (MS) to 0, which is the default value on reset.

Setting the SSPCR1 register MS bit to 1 configures the SSP as a slave. When configured as a slave, enabling or disabling of the SSP SSPTXD signal is provided through the SSPCR1 slave mode SSPTXD output disable bit (SOD). This can be used in some multi-slave environments where masters might parallel broadcast.

To enable the operation of the PrimeCell SSP set the Synchronous Serial Port Enable (SSE) bit to 1.

Bit rate generation

Dividing down the input clock SSPCLK derives the serial bit rate. The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in SSPCPSR. The clock is further divided by a value from 1 to 256, which is 1 + SCR, where SCR is the value programmed in SSPCR0.

The frequency of the output signal bit clock SSPCLKOUT is:

$$SSPCLKOUT = \frac{(FSSPCLK)}{[CPSDVR \cdot (1 + SCR)]}$$

For example, if SSPCLK is 3.6864 MHz, and CPSDVSR = 2, then SSPCLKOUT has a frequency range from 7.2 kHz to 1.8432 MHz.

13.5.5 Frame format

Each data frame is between 4 and 16 bits long depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Motorola SPI
- National semiconductor microwire.

For all three formats, the serial clock (SSPCLKOUT) is held inactive while the SSP is idle, and transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSPCLKOUT is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Motorola SPI and National Semiconductor Microwire frame formats, the serial frame (SSPFSSOUT) pin is active LOW, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the SSPFSSOUT pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSP and the off-chip slave device drive their output data on the rising edge of SSPCLKOUT, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the National Semiconductor Microwire format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8 bit control message is transmitted to the off-chip slave. During this transmission no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8 bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

13.6 Programming model

13.6.1 External pin connections

Table 211. External pin connection

Signal	Ball
CLK	C2
MOSI	B2
MISO	B1
CS1	D3
CS2	E8
CS3	D8
CS4	C8

13.6.2 Register map

The SSP can be fully configured by programming its registers which can be accessed through the APB slave interface at the following base address:

Table 212. SSP registers summary

Name	Offset	Type	Width (bit)	Reset value	Description
SSPCR0	0x000	R/W	16	16'h0	Control register 0
SSPCR1	0x004	R/W	4	4'h0	Control register 1
SSPDR	0x008	R/W	16	-	Receive FIFO (read) and transmit FIFO (write) data register
SSPSR	0x00C	RO	5	5'h3	Status register
SSPCPSR	0x010	R/W	8	8'h0	Clock prescale register
SSPIMSC	0x014	R/W	4	4'h0	Interrupt mask set and clear register
SSPRIS	0x018	RO	4	4'h8	Raw interrupt status register
SSPMIS	0x01C	RO	4	4'h0	Masked interrupt status register
SSPICR	0x020	WO	4	4'h0	Interrupt clear register
SSPDMACR	0x024	R/W	2	2'h0	DMA control register
Reserved	0x028 to 0xFDC	-	-	-	Reserved
SSPPeriphID0	0xFE0	RO	8	8'h22	Peripheral identification register bits 7:0
SSPPeriphID1	0xFE4	RO	8	8'h10	Peripheral identification register bits 15:8
SSPPheriphID2	0xFE8	RO	8	8'h4	Peripheral identification register bits 23:16
SSPPheriphID3	0xFEC	RO	8	8'h0	Peripheral identification register bits 31:24

Table 212. SSP registers summary (continued)

Name	Offset	Type	Width (bit)	Reset value	Description
SSPCellID0	0xFF0	RO	8	8'hD	PrimeCell identification register bits 7:0
SSPCellID1	0xFF4	RO	8	8'hF0	PrimeCell identification register bits 15:8
SSPCellID2	0xFF8	RO	8	8'h5	PrimeCell identification register bits 23:16
SSPCellID3	0xFFC	RO	8	8'hB1	PrimeCell identification register bits 31:24

13.6.3 Register description

13.6.4 SSPCR0 register

SSPCR0 is control register 0 and contains five bit fields that control various functions within the SSP.

Table 213. SSPCR0 register bit assignments

Bit	Name	Type	Description
[15:08]	SCR	R/W	Serial clock rate. The value SCR is used to generate the transmit and receive bit rate of the SSP. The bit rate is: PCLK CPSDVR * (1 + SPR) Where CPSDVSR is an even value from 2 to 254, programmed through the SSPCPSR register and SCR is a value from 0 to 255.
[07]	SPH	R/W	CLKOUT phase (applicable to Motorola SPI frame format only). See Motorola SPI frame format on page xxx
[06]	SPO	R/W	CLKOUT polarity (applicable to Motorola SPI frame format only). See Motorola SPI frame format on page
[05:04]	FRF	R/W	Frame format: 2'b00 = Motorola SPI frame format 2'b01 = TI synchronous serial frame format 2'b10 = National microwire frame format 2'b11 = Reserved, undefined operation
[03:00]	DSS	R/W	Data size select: 4'b0000 = Reserved, undefined operation 4'b0001 = Reserved, undefined operation 4'b0010 = Reserved, undefined operation 4'b0011 = 4 bit data 4'b0100 = 5 bit data ……. 4'b1110 = 15 bit data 4'b1111 = 16 bit data

13.6.5 SSPCR1 register

SSPCR1 is the control register 1 and contains four different bit fields, which control various functions within the SSP.

Table 214. SSPCR1 register bit assignments

Bit	Name	Type	Description
[15:04]	-	-	Reserved. Read unpredictable, should be written as 0
[03]	SOD	R/W	Slave-mode output disable. This bit is relevant only in the slave mode (MS=1). In multiple-slave systems, it is possible for an SSP master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto its serial output line. In such systems the RXD lines from multiple slaves could be tied together. To operate in such systems, the SOD bit can be set if the SSP slave is not supposed to drive the SSPTXD line. 1'b0 = SSP can drive the TXD output in slave mode. 1'b1 = SSP must not drive the TXD output in slave mode.
[02]	MS	R/W	Master or slave mode select. This bit can be modified only when the SSP is disabled (SSE=0): 1'b0 = device configured as master (default) 1'b1 = device configured as slave.
[01]	SSE	R/W	Synchronous serial port enable: 1'b0 = SSP operation disabled 1'b1 = SSP operation enabled.
[00]	LBM	R/W	Loop back mode: 1'b0 = Normal serial port operation enabled 1'b1 = Output of transmit serial shifter is connected to input of receive serial shifter internally.

13.6.6 SSPDR register

SSPDR is the data register and is 16 bits wide. When SSPDR is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the PrimeCell SSP receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When SSPDR is written to, the entry in the transmit FIFO (pointed to by the write pointer), is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the SSPTXD pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the PrimeCell SSP is programmed for National Microwire frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when SSE is set to zero. This allows the software to fill the transmit FIFO before enabling the PrimeCell SSP. *Table 215* shows the bit assignments for SSPDR.

Table 215. SSPDR register bit assignments

Bit	Name	Type	Description
[15:00]	DATA	R/W	Transmit/receive FIFO: Read = Receive FIFO Write = Transmit FIFO You must right justify data when the SSP is programmed for a data size that is less then 16 bits. Unused bits are ignored by transmit logic. The receive logic automatically right justifies.

13.6.7 SSPSR register

SSPSR is a read only register that contains bits that indicates the FIFO fill status and the SSP busy status.

Table 216. SSPSR register bit assignments

Bit	Name	Type	Description
[15:05]	-	-	Reserved, read unpredictable, should be written as 0
[04]	BSY	RO	SSP busy flag: 1'b0 = SSP is idle 1'b1 = SSP is currently transmitting or receiving a frame
[03]	RFF	RO	Receive FIFO Full: 1'b0 = receive FIFO is not full 1'b1 = Receive FIFO is full
[02]	RNE	RO	Receive FIFO not empty: 1'b0 = Receive FIFO is empty 1'b1 = receive FIFO is not empty
[01]	TNF	RO	Transmit FIFO not full: 1'b0 = Transmit FIFO is full 1'b1 = transmit FIFO is not full
[00]	TFE	RO	Transmit FIFO empty: 1'b0 = Transmit FIFO is not empty 1'b1 = transmit FIFO is empty

13.6.8 SSPCPSR register

SSPCPSR is the clock prescale register and specifies the division factor by which the input SSPCLK must be internally divided before further use.

The value programmed into this register must be an even number between 2 to 254. The least significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least significant bit as zero. The SSPCPSR bit assignments are given in *Table 217*.

Table 217. SSPCPSR register bit assignments

Bit	Name	Type	Description
[15:08]	-	-	Reserved, read unpredictable, must be written as 0.
[07:00]	CPSDVSR	R/W	Clock prescale divisor. Must be an even number from 2 to 254, depending on the frequency of SSPCLK. The least significant bit always returns zero on reads.

13.6.9 SSPIMSC register

The SSPIMSC register is the interrupt mask set or clear register. It is a read/write register. On a read this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask. All the bits are cleared to 0 when reset. The SSPIMSC bit assignments are given in *Table 218*.

Table 218. SSPIMSC register bit assignments

Bit	Name	Type	Description
[15:04]	-	-	Reserved, read as 0, do not modify.
[03]	TXIM	R/W	Transmit FIFO interrupt mask: 1'b0 = Tx FIFO half empty or less condition interrupt is masked. 1'b1 = Tx FIFO half empty or less condition interrupt is not masked
[02]	RXIM	R/W	Receive FIFO interrupt mask: 1'b0 = Rx FIFO half full or less condition interrupt is masked 1'b1 = Rx FIFO half full or less condition interrupt is not masked.
[01]	RTIM	R/W	Receive timeout interrupt mask: 1'b0 = Rx FIFO not empty and no read prior to timeout period interrupt is masked 1'b1= Rx FIFO not empty and no read prior to timeout period interrupt is not masked.
[00]	RORIM	R/W	Receive overrun interrupt mask: 1'b0 = Rx FIFO written to while full condition interrupt is masked 1'b1= Rx FIFO written to while full condition interrupt is not masked.

13.6.10 SSPRIS register

The SSPRIS register is the raw interrupt status register. It is a read-only register. On a read this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect. The SSPRIS bit assignments are given in *Table 219*.

Table 219. SSPRIS register bit assignments

Bit	Name	Type	Description
[15:04]	-	-	Reserved, read as 0, do not modify.
[03]	TXRIS	RO	Gives the raw interrupt state (prior to masking) of the SSPTXINTR interrupt

Table 219. SSPRIS register bit assignments (continued)

Bit	Name	Type	Description
[02]	RXRIS	RO	Gives the raw interrupt state (prior to masking) of the SSPRXINTR interrupt
[01]	RTRIS	RO	Gives the raw interrupt state (prior to masking) of the SSPRTINTR interrupt
[00]	RORRIS	RO	Gives the raw interrupt state (prior to masking) of the SSPRORINTR interrupt

13.6.11 SSPMIS Register

The SSPMIS register is the masked interrupt status register. It is a read-only register. On a read this register gives the current masked status value of the corresponding interrupt. A write has no effect. The SSPMIS bit assignments are given in *Table 220*.

Table 220. SSPMIS register bit assignments

Bit	Name	Type	Description
[15:04]	-	-	Reserved, read as 0, do not modify
[03]	TXMIS	RO	Gives the transmit FIFO masked interrupt state (after masking) of the SSPTXINTR interrupt.
[02]	RXMIS	RO	Gives the transmit FIFO masked interrupt state (after masking) of the SSPRXINTR interrupt.
[01]	RTMIS	RO	Gives the transmit FIFO masked interrupt state (after masking) of the SSPRTINTR interrupt.
[00]	RORMIS	RO	Gives the transmit FIFO masked interrupt state (after masking) of the SSPRORINTR interrupt.

13.6.12 SSPICR register

The SSPICR register is the interrupt clear register and is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect. The SSPICR bit assignments are given in *Table 221*.

Table 221. SSPICR register bit assignments

Bit	Name	Type	Description
[15:02]	-	-	Reserved, read as 0, do not modify.
[01]	RTIC	WO	Clear the SSPRTINTR interrupt.
[00]	RORIC	WO	Clear the SSPRORINTR interrupt.

13.6.13 SSPDMACR register

The SSPDMACR register is the DMA control register. It is a read/write register. All the bits are cleared to 0 on reset. The SSPDMACR bit assignments are given in *Table 222*.

Table 222. SSPDMACR register bit assignments

Bit	Name	Type	Description
[15:02]	-	-	Reserved, read as 0, do not modify.
[01]	TXDMAEn	R/W	If this bit is set to 1, DMA for the transmit FIFO is enabled.
[00]	RXDMAEn	R/W	If this bit is set to 1, DMA for the receive FIFO is enabled.

13.6.14 PHERIPHID0 register

Table 223. PHERIPHID0 register bit assignments

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PartNumber0	RO	These bits read back as 0x22

13.6.15 PHERIPHID1 register

Table 224. PHERIPHID1 register bit assignment

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:04]	Designer0	RO	These bits read back as 0x1
[03:00]	PartNumber1	RO	These bits read back as 0x0

13.6.16 PHERIPHID2 register

Table 225. PHERIPHID2 register bit assignments

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:04]	Revision	RO	These bits read back as 0x0
[03:00]	Designer1	RO	These bits read back as 0x4

13.6.17 PHERIPHID3 register

Table 226. PHERIPHID3 register bit assignments

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	Configuration	RO	These bits read back as 0x00

13.6.18 PCELLID0 register

Table 227. PCELLID0 register bit assignments

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLID0	RO	These bits read back as 0x0D

13.6.19 PCELLID1 register

Table 228. PCELLID1 register bit assignment

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLID1	RO	These bits read back as 0xF0

13.6.20 PCELLID2 register

Table 229. PCELLID2 register bit assignment

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLID2	RO	These bits read back as 0x05

13.6.21 PCELLID3 register

Table 230. PCELLID3 register bit assignment

Bit	Name	Type	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLID3	RO	These bits read back as 0xB1

13.7 Interrupts

There are five interrupts generated by the SSP. Four of these are individual, maskable, active HIGH interrupts:

- SSPRXINTR - SSP receive FIFO service interrupt request.
- SSPTXINTR - SSP transmit FIFO service interrupt request.
- SSPRORINTR - SSP receive overrun interrupt request
- SSPRTINTR - SSP time out interrupt request.

The fifth is a combined single interrupt SSPINTR.

You can mask each of the four individual maskable interrupts by setting the appropriate bits in the SSPIMSC register. Setting the appropriate mask bit HIGH enables the interrupt.

Provision of the individual outputs as well as a combined interrupt output, allows use of either a global interrupt service routine, or modular device drivers to handle interrupts.

The transmit and receive dynamic dataflow interrupts SSPTXINTR and SSPRXINTR have been separated from the status interrupts, so that data can be read or written in response to just the FIFO trigger levels.

The status of the individual interrupt sources can be read from SSPRIS and SSPMIS registers.

13.7.1 SSPRXINTR

The receive interrupt is asserted when there is four or more valid entries in the receive FIFO.

13.7.2 SSPTXINTR

The transmit interrupt is asserted when there are four or less valid entries in the transmit FIFO. The transmitter interrupt SSPTXINTR is not qualified with the SSP enable signal, which allows operation in one of two ways. Data can be written to the transmit FIFO prior to enabling the PrimeCell SSP and the interrupts. Alternatively, the SSP and interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.

13.7.3 SSPRORINTR

The receive overrun interrupt SSPORINTR is asserted when the FIFO is already full and an additional data frame is received, causing an overrun of the FIFO. Data is over-written in the receive shift register, but not the FIFO.

13.7.4 SSPRTINTR

The receive timeout interrupt is asserted when the receive FIFO is not empty and the SSP has remained idle for a fixed 32 bit period. This mechanism ensures that the user is aware that data is still present in the receive FIFO and requires servicing. This interrupt is deasserted if the receive FIFO becomes empty by subsequent reads, or if new data is received on SSPRXD. It can also be cleared by writing to the RTIC bit in the SSPICR register.

13.7.5 SSPINTR

The interrupts are also combined into a single output SSPINTR, that is an OR function of the individual masked sources. You can connect this output to the system interrupt controller to provide another level of masking on an individual per-peripheral basis. The combined SSP interrupt is asserted if any of the four individual interrupts above are asserted and enabled.

14 BS_System controller

14.1 Overview

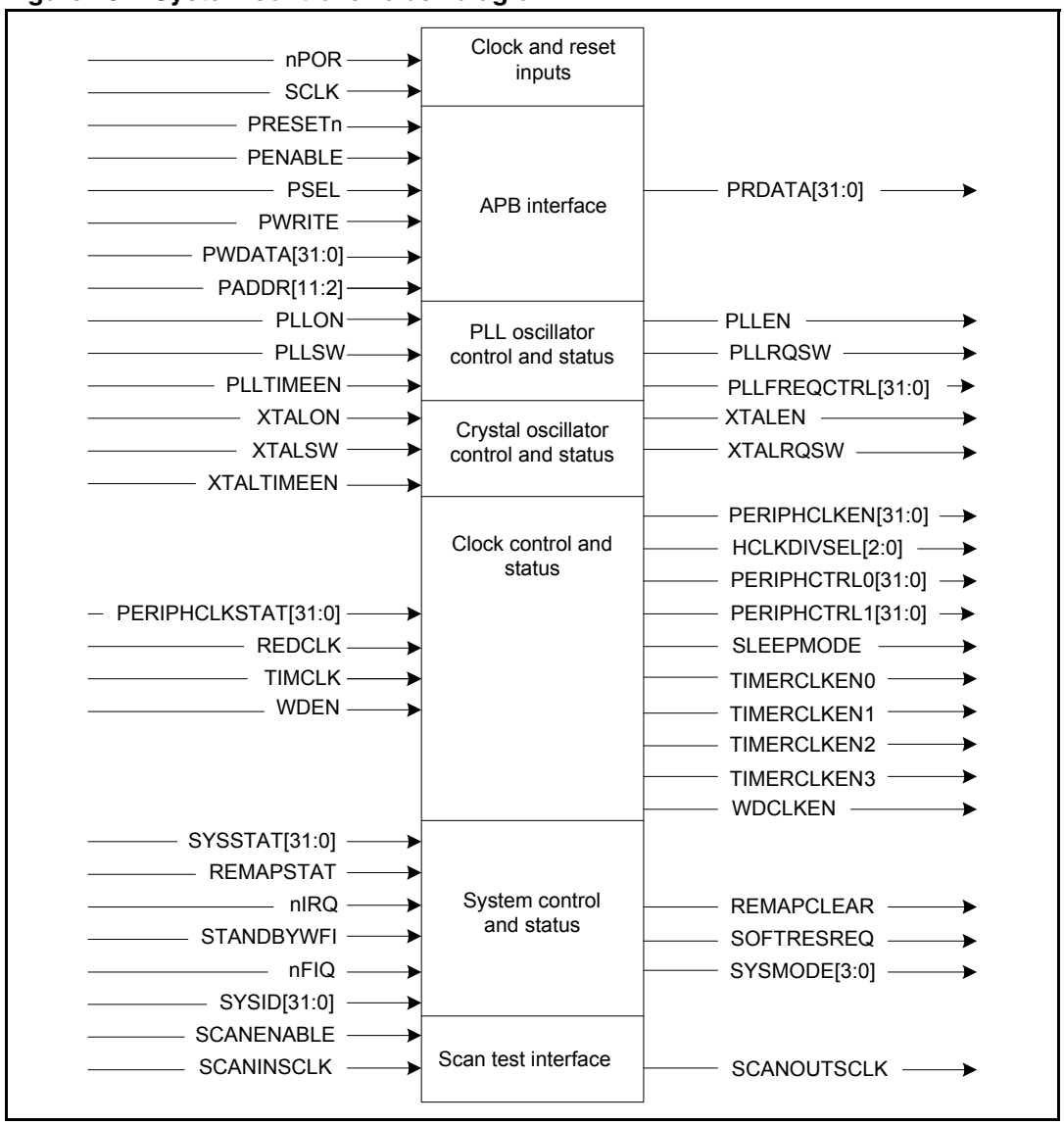
Within its Basic Subsystem, the device provides a System Controller, which is used to supply an interface to control the operation of the overall system.

Main features of the System Controller are listed below:

- Provides a system mode control state machine
- Integrates crystal and PLL control
- Defines the system response to interrupts
- Implements soft reset generation
- Generates Watchdog module clock enable

14.2 Block diagram

Figure 23. System controller block diagram



14.3 Main function description

14.3.1 System mode control

Here there is a description of how the system controller changes between modes: DOZE, SLEEP, SLOW and NORMAL.

The state machine is controlled using three mode control bits (ModeCtrl[2:0]) in the system control register (SCCTRL), which define the required system operating mode. The mode control bits control the modes:

- 1xx If the most significant bit (MSB) is set then the system moves into NORMAL mode.
- 01x If the MSB is not set and the next MSB is set then the system moves into SLOW mode.
- 001 If only the least significant bit (LSB) is set then the system moves into DOZE mode.
- 000 If none of the mode control bits are set the system moves into SLEEP mode.

When the required operating mode has been defined in the system mode control register the system mode control state machine moves to the required operating mode without further software interaction.

The current system mode is output on the SYSMODE[3:0] bus and can also be read back by the processor using the ModeStatus bit in the SCCTRL register.

If the nPOR input is activated, the state machine and the required operating mode in the system control register are set to DOZE. If the PRESETn input is activated, the system mode control state machine does not change mode but the required operating mode is set to DOZE in the system control register.

SLEEP mode

During this mode the system clocks, HCLK and CLK, are disabled and the System Controller clock SCLK is driven from a low speed oscillator (nominally 32 768 Hz). When either a FIQ or an IRQ interrupt is activated (through the VIC) the system moves into the DOZE mode. Additionally, the required operating mode in the system control register automatically changes from SLEEP to DOZE.

DOZE mode

During this mode the system clocks, HCLK and CLK, and the System Controller clock SCLK are driven from the output of crystal oscillator (24 MHz) or low frequency oscillator (32 kHz). The System Controller moves into SLEEP mode from DOZE mode only when none of the mode control bits are set and the processor is in Wait-for-interrupt state. If SLOW mode or NORMAL mode is required the system moves into the XTAL control transition state to initialize the crystal oscillator.

SLOW mode

During this mode, both the system clocks and the System Controller clock are driven from the output of the crystal oscillator.

If NORMAL mode is required, the system moves into the “PLL control” transition state. If neither the SLOW nor the NORMAL mode control bits are set, the system moves into the “Switch from XTAL” transition state.

NORMAL mode

In NORMAL mode, both the system clocks and the System Controller clock are driven from the output of PLL.

If the NORMAL mode control bit is not set, then the system moves into the “Switch from PLL” transition state.

XTAL control transition state, XTAL CTL

XTAL control transition state is used to initialize the crystal oscillator. While in this state, both the system clocks and the System Controller clock are driven from a low-frequency oscillator.

The system moves into the Switch to XTAL transition state when the crystal oscillator output is stable. This is indicated when either the XTAL timeout defined in the XTAL control register expires (when the XTALTIMEEN input is valid) or by the XTALON input being set to logic ‘1’.

Switch to XTAL transition state, SW TO XTAL

Switch to XTAL transition state is used to initiate the switching of the system clock source from the slow speed oscillator to the crystal oscillator. The system moves into the SLOW mode when the XTALSW input is set to logic ‘1’, to indicate that the clock switching is complete.

Switch from XTAL transition state (SW from XTAL)

The “Switch from XTAL” transition state is entered when moving from SLOW to DOZE mode. It is used to initiate the switching of the system clock source from the crystal oscillator to the low speed oscillator. The system moves into the DOZE mode when the XTALSW input is set to PLL control transition state (PLL CTL)

The “PLL control” transition state is used to initialize the PLL. While in this state, both the system clocks and the System Controller clock are driven by the output of the crystal oscillator.

The system moves then into the “Switch to PLL” transition state when the PLL timeout defined in the PLL control register (SCPLLCTRL register) expires (when the PLLTIMEEN input is invalid) and if the PLLON input is set to logic1.

Switch to PLL transition state (SW to PLL)

The “Switch to PLL” transition state is used to initiate the switching of the system clock source from the crystal oscillator to the PLL output.

The system moves then into the NORMAL mode when the PLLSW input is set to logic ‘1’ to indicate that the clock switching is complete.

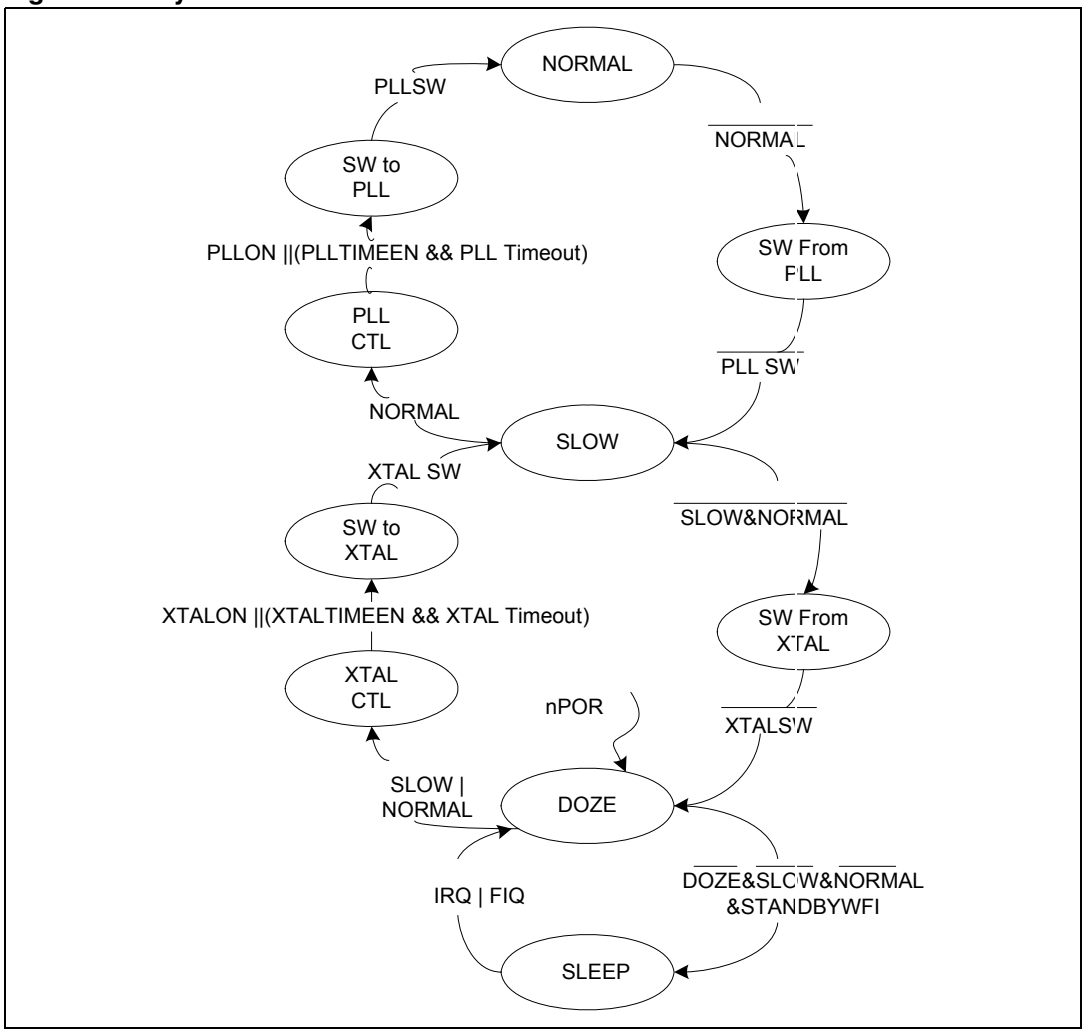
Switch from PLL transition state (SW from PLL)

The “Switch from PLL” transition state is entered when moving from NORMAL to SLOW mode. It is used to initiate the switching of the clock sources from the PLL to the crystal oscillator output.

The system moves then into the SLOW mode when the PLLSW input is set to logic ‘0’ to indicate that the clock switching is complete.

14.3.2 System control state machine

Figure 24. System mode control state machine



Crystal oscillator and PLL control

The system control state machine (*Section 14.3.2: System control state machine*) can also be used to control the crystal oscillator and the PLL.

Nevertheless, software can be used to override control of the crystal and PLL by using the crystal control register (SCXTALCTRL, *Section 14.4.7*) and the PLL control register (SCPLLCTRL).

PLL frequency control

To define the frequency of the clock generated from the PLL, the system controller provides a PLL frequency control register (SCPLLFCTRL, *Section 14.3.5*).

14.3.3 Interrupt response mode

To enable the best possible response to interrupts, the present mode bits can be override in the system control register after an interrupt has been generated. This enables, for example, the state machine to move from the DOZE to the NORMAL mode after an interrupt.

The interrupt response functionality is controlled by the interrupt mode control register (SCIMCTRL, *Section 14.4.5*), which defines if the functionality has been enabled, the mode of operation that is required following an interrupt, what type of interrupt is permitted to enable the interrupt response mode, an interrupt mode status bit and clear mechanism.

Note: It is not possible for the interrupt response mode to slow the system operating speed, for example, changing mode from NORMAL to SLOW.

The interrupt response mode is cleared by writing a 1‘b0 to the interrupt mode control register SCIMCTRL. Following a power-on reset, the interrupt response mode is disabled.

14.3.4 Reset control

The reset control is used to request a soft reset to be generated by asserting the SOFTRESREQ output for a single SCLK cycle when any value is written to the reset status register.

14.3.5 Core clock control

To enable the software to control the relative frequency of the core clock (CLK) and the bus clock (HCLK), the system controller provides access to the HCLKDIVSEL[2:0] output through the system control register (namely the 3 bit field HCLKDivSel of the SCCTRL register, *Section 14.4.3*).

These output signals are intended for use by the clock generation logic to control the generation of the CLK/HCLK clock source and the HCLKEN input. To prevent spurious change of the CLK/HCLK clock ratio, the HCLKDIVSEL output is only allowed to change when the system mode control state machine is in a stable state, that is, the actual system mode matches the required system mode.

14.3.6 Watchdog module clock enable generation

Enable signals are generated by the system controller to allow the watchdog module to be clocked at a rate that is independent of the system clock `SCLK`. In particular, the enable signals are generated by sampling a free-running, constant frequency input clock and generating an active high pulse for a single SCLK clock cycle on each rising edge of the input clock.

The supported module enable signals are:

- WDCLKEN for the watchdog module clock enable.

The enable signal for the watchdog module is generated from the REFCLK input, as defined in the system control register (SCCTRL, *Section 14.4.3*).

Additionally, to enable the watchdog module to be clocked directly at the system clock rate, it is also possible to selectively force the enable outputs high. The watchdog clock enable output can be forced inactive by deasserting the WDEN input (for example, the WDEN input can be used to disable the watchdog timer when the processor core is in a debug state).

14.4 Programming model

14.4.1 Register map

The system controller can be fully configured by programming its registers which can be accessed at the base address 0xFCA0_0000

System controller registers can be logically arranged in two main groups:

- Control and status registers, CSRs (listed in *Table 231*), for system controller configuration,
- Identification registers (listed in *Table 232*), namely twelve 8 bit RO registers (which can be treated as three 32 bit registers) reporting system information and system controller-specific information. Refer to ARM technical documentation for further details.

Note: In addition to reserved locations within the CSRs address space (Table 231), offset addresses from 0xF00 to 0xFDC are reserved for test purposes. All these locations must not be used during normal operation.

Table 231. System controller control and status registers summary

Name	Offset	Width [bit](1) (2)	Type	Reset value	Description
SCCTRL	0x000	24	RW	24'h000009	System control
SCSYSSTAT	0x004	32	RW	-	System status
SCIMCTRL	0x008	8	RW	8'h00	Interrupt mode control
SCIMSTAT	0x00C	1	RW	1'h0	Interrupt mode status
SCXTALCTRL	0x010	19	RW	19'h0	Crystal control
SCPLLCTRL	0x014	28	RW	28'h0	PLL control
-	0x018 to 0xEDC	-	-	-	Reserved

1. This value represents the actual number of used bits, being reserved the others to 32.

Table 232. System controller identification registers summary

Name	Offset	Width[bit]	Type	Reset Value	Description
SCSYSID0	0xEE0	8	RO	8'h00	System identification
SCSYSID1	0xEE4	8	RO	8'h00	
SCSYSID2	0xEE8	8	RO	-	
SCSYSID3	0xEEC	8	RO	-	
-	0xEF0 to 0xEFC	-	-	-	Reserved

Table 232. System controller identification registers summary (continued)

Name	Offset	Width[bit]	Type	Reset Value	Description
SCPeriphID0	0xFE0	8	RO	8'h10	Peripheral identification
SCPeriphID1	0xFE4	8	RO	8'h18	
SCPeriphID2	0xFE8	8	RO	8'h04	
SCPeriphID3	0xFEC	8	RO	8'h00	
SCPCellID0	0xFF0	8	RO	8'h0D	Identification Registes
SCPCellID1	0xFF4	8	RO	8'hF0	
SCPCellID2	0xFF8	8	RO	8'h05	
SCPCellID3	0xFFC	8	RO	8'hB1	

14.4.2 Register description

14.4.3 SCCTRL register

The SCCTRL (control) is a RW register which is used to define the required operation of the system controller. The SCCTRL bit assignments are given in *Table 233*.

Table 233. SCCTRL register bit assignments

Bit	Name	Reset value	Description
[31:24]	Reserved	-	Read: undefined. Write: should be zero
[23]	WDogEnOv	1'h0	Watchdog enable override This bit allows to control the watchdog enable output signal (*Section 14.3.6*), according to the encoding: 1'b0 = Derived from REFCLK clock source, as defined in *Section 14.3.6* 1'b1 = Forced high
[22:21]	Reserved	-	Read: undefined. Write: should be zero
[20]	TimerEn2Ov	1'h0	Timer enable 2, override
[19]	TimerEn2Sel	1'h0	Timer enable 2, timing reference select
[18]	TimerEn1Ov	1'h0	Timer enable 1, override If set, this bit forces high the timer enable output signal. Otherwise (bit cleared), the enable output signal follows the rules defined in *Section 14.3.6*
[17]	TimerEn1Sel	1'h0	Timer enable 1, timing reference select This bit allows to select the reference clock for the timer enable signals (*Section 14.3.6*), according to the encoding: 1'b0 = REFCLK 1'b1 = TIMCLK
[16]	TimerEn0Ov	1'h0	Timer enable 0, override

Table 233. SCCTRL register bit assignments (continued)

Bit	Name	Reset value	Description
[15]	TimerEn0Sel	1'h0	Timer enable 0, timing reference select
[14:12]	HCLKDivSel	3'h0	Control the HCLKDIVSEL output
[11:10]	Reserved	-	Read: undefined. Write: should be zero
[09]	RemapStat	1'h0	Remap status This bit is used to return the value of the REMAPSTAT input.
[08]	RemapClear	1'h0	Remap clear request This bit is used to control the REMAPCLEAR output. Setting this bit indicates that the memory remap will be cleared.
[07]	Reserved	-	Read: undefined. Write: should be zero
[06:03]	ModeStatus	4'h01	Mode status bitsThis 4 bit field returns the current operation mode as defined by the system controller state machine (*Section 14.3.1: System mode control*), according to the encoding: 4'b0000 = SLEEP 4'b0001 = DOZE (reser value) 4'b0010 = SLOW 4'b0011 = XTAL CTL 4'b0100 = NORMAL 4'b0101 = Not used 4'b0110 = PLL CTL 4'b0111 = Not used 4'b1000 = Not used 4'b1001 = SW from XTAL 4'b1010 = SW from PLL 4'b1011 = SW to XTAL 4'b1100 = Not used 4'b1101 = Not used 4'b1110 = SW to PLL 4'b1111 = Not used
[02:00]	ModeCtrl	3'h01	Mode control bits This 3 bit field defines the required operation mode (*Section 14.3.1: System mode control*), according to the encoding (x is don't care): 3'b000 = SLEEP 3'b001 = DOZE (reset value) 3'b01x = SLOW 3'b1xx = NORMAL

14.4.4 SCSYSSTAT register

Writing any value to the SCSYSSTAT (system status) 32 bit RW register causes the SOFTRESREQ output (soft reset request) to pulse high for a single clock cycle.

14.4.5 SCIMCTRL register

The SCIMCTRL (interrupt mode control) is a RW register which is used to enable and control the operation of the system controller when an interrupt has been generated. The SCIMCTRL bit assignments are given in *Table 234*.

Table 234. SCIMCTRL register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined. Write: should be zero
[07]	InMdType	1'h0	Interrupt mode type This bit is used to define which type of interrupt can cause the system to enter interrupt mode, according to the encoding: 1'b0 = FIQ 1'b1 = FIQ and IRQ
[06:04]	Reserved	-	Read: undefined. Write should be zero.
[03:01]	ItMdCtrl	3'h0	Interrupt mode control bits This 3 bit field defines the slowest operating mode that must be requested when in interrupt mode.
[00]	ItMdEn	1'h0	Interrupt mode enable This bit is used to enable the interrupt mode, according to the encoding: 1'b0 = Disabled 1'b1 = Entered when an interrupt becomes active

14.4.6 SCIMSTAT register

The SCIMSTAT (interrupt mode status) is a RW register which is used to monitor and control the system controller interrupt mode. The SCIMSTAT bit assignments are given in *Table 235*.

Table 235. SCIMSTAT register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read: undefined. Write: should be zero
[00]	ItMdStat	1'h0	Interrupt mode status This bit is used to enable the interrupt mode, according to the encoding: 1'b0 = Not active. 1'b1 = Active. This bit can be directly written to enable software control of the interrupt mode logic.

Note: The interrupt mode must be cleared manually when the interrupt service routine has completed executing.

14.4.7 SCXTALCTRL register

The SCXTALCTRL (crystal control) is a RW register which is used to directly control the crystal oscillator used to generate the system clock SCLK in both SLOW and NORMAL mode (*Section 14.3.1: System mode control*). The SCXTALCTRL bit assignments are given in *Table 236*.

Table 236. SCXTALCTRL register bit assignments

Bit	Name	Type	Reset value	Description
[31:19]	Reserved	-	-	Read: undefined. Write: should be zero
[18:03]	XtalTime	RW	16'h0	Crystal timeout count This value is used to define the number of slow speed oscillator cycles permitted for the crystal oscillator output to settle after being enabled. The timeout is given by: 65536 – XtalTime.
[02]	XtalStat	RO	1'h0	Crystal status bit This RO bit returns the value on the XTALON input signal.
[01]	XtalEn	RW	1'h0	Crystal enable bit This bit is used to directly control the XTALEN output when the crystal control override is enabled (XtalOver bit set to 'b1 in this register).
[00]	XtalOver	RW	1'h0	Crystal control override If set, this bit enables the crystal control signals (from system controller) to be placed under direct software control, rather than being controlled by the system mode control state machine.

14.4.8 SCPLLCTRL register

The SCPLLCTRL (PLL Control) is a RW register which allows the system controller to directly control the PLL. The SCPLLCTRL bit assignments are given in *Table 237*.

Table 237. SCPLLCTRL register bit assignments

Bit	Name	Type	Reset value	Description
[31:28]	Reserved	-	-	Read: undefined. Write: should be zero
[27:03]	PllTime	RW	25'h0	PLL timeout count This value is used to define the number of crystal oscillator cycles permitted for the PLL output to settle after being enabled.The timeout value is given by: 33554432 – PllTime.

Table 237. SCPLLCTRL register bit assignments (continued)

Bit	Name	Type	Reset value	Description
[02]	PllStat	RO	1'h0	PLL status bit This RO bit returns the value on the PLLON input signal.
[01]	PllEn	RW	1'h0	PLL enable bit This bit is used to directly control the PLLEN output when the PLL control override is enabled (PllOver bit set to 'b1 in this register).
[00]	PllOver	RW	1'h0	PLL control override If set, this bit enables the PLL control signals (from the system controller) to be placed under direct software control, rather than being controlled by the system mode control state machine.

15 BS_Serial memory interface

15.1 Overview

SPEAr300 provides a serial memory interface (SMI), acting as an AHB slave interface (32, 16 or 8 bit) to SPI-compatible off-chip memories. SMI allows the CPU to use these serial memories for both data storage and code execution.

Main features of SMI are:

- Supports a group of SPI-compatible Flash and EEPROM devices, namely:
  - STMicroelectronics M25Pxxx, M45Pxxx,
  - STMicroelectronics M95xxx, except M95040, M95020 and M95010,
  - ATMEL AT25Fxx,
  - YMC Y25Fxx,
  - SST SST25LFxx.
- Acts always as a SPI master and up to 2 SPI slave memory devices are supported (through as many chip select signals), with up to 16 MB address space each.
- The SMI clock signal (SMICLK) is generated by SMI and input to all slaves.
- SMICLK can be up to 50 MHz in fast read mode (or 20 MHz in normal mode), and it can be controlled by a programmable 7 bits prescaler allowing then 127 different clock frequencies.

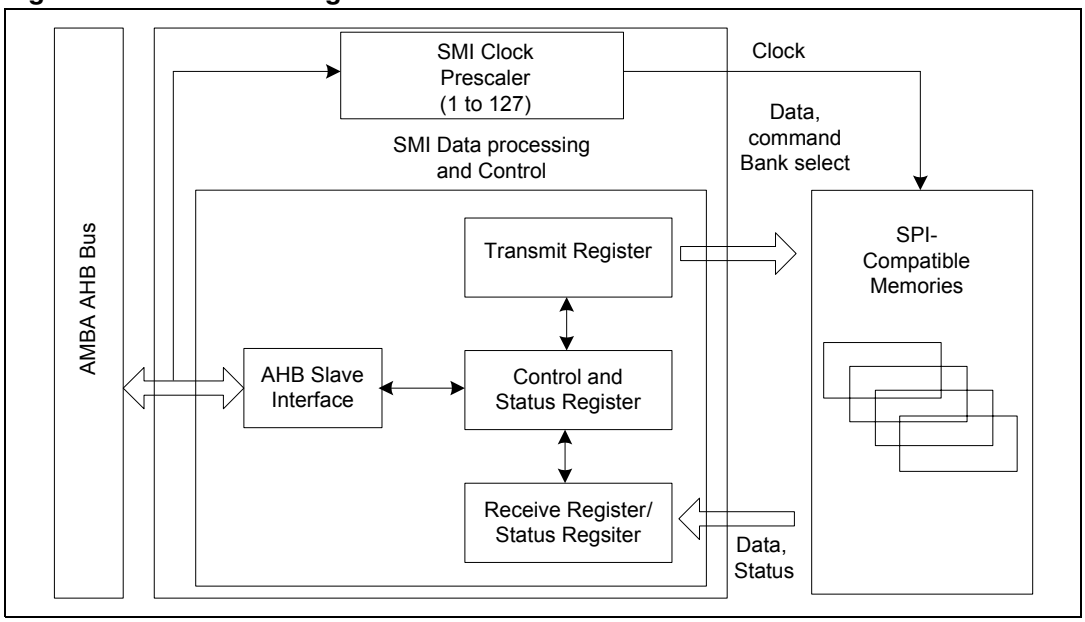
15.2 Block diagram

Figure 25 shows the block diagram of SMI.

SMI consists of two main functions which are detailed in the following sections:

- The Clock Prescaler (*Section 15.3.1*).
- The Data Processing and Control (*Section 15.3.2*).

Figure 25. SMI block diagram



15.3 Main functions description

15.3.1 Clock prescaler

The SMI clock prescaler block allows to set-up the memory clock SMI_CK using the AHB clock HCLK, as detailed in *Section 15.6*.

15.3.2 Data processing and control

The SMI data processing and control block represents the logic controlling the transfer of data between SPI-compatible off-chip memory and AHB bus. Transfer rules through both AHB-to-SMI and SMI-to-memory interfaces. Different data transfer mode between SPI-compatible off-chip memory and AHB bus are detailed in *Section 15.5*.

AHB-to-SMI interface

Acting as an AHB slave interface, the SMI is accessed by AHB master through AHB bus. The following rules apply to this interface:

- Endianness is fixed to little-endian.
- SPLIT / RETRY responses from AHB slave (that is., the SMI) are not supported.
- Size of data transfers to external serial memories can be byte, half-word or word (that is, 8, 16 or 32 bit).
- Size of data transfers to SMI registers must be 32 bits.
- Read requests: all types of BURST defined by AHB protocol are supported (single, wrapping and incrementing). Please note that wrapping bursts take more time than incrementing bursts, as there is a break in the address increment.

Note: If EEPROMs are used instead of Flash memories, a read request address should be (ADDRESS + 1), being ADDRESS the actual target address to be read.

- Write requests: wrapping bursts are not supported, causing an ERROR response on HRESP sent back by SMI to AHB master.
- Bursts must not cross bank boundaries.
- In case of BUSY transfer, the SMI is held until BUSY is inactive.

SMI-to-MEMORY interface

Acting as a SPI master, the SMI supports a synchronous full-duplex data link with its up to 2 SPI slaves (that is, the serial memory devices).

It follows that each SPI slave must agree with the communication protocol fixed by SMI in terms of clock polarity (CPOL) and clock phase (CPHA), specifically CPOL = 1 and CPHA = 1 (that is, clock idles high and data are shifted in and out on the rising edge of the clock).

Prior to any operation involving a SPI-compatible off-chip memory device, the related SPI slave must be selected by SMI (through chip select), then a 1-byte instruction must be sent by SMI to the selected memory. The set of instructions supported by SMI is given in *Table 238*.

Table 238. SMI supported instruction

Opcode	Description
8'h03	Read data bytes.
8'h0B	Read data bytes at high speed.
8'h05	Read status register.
8'h06	Write enable.
8'h02	Page program.
8'hAB	Release from deep power-down.

15.4 Operation modes

SMI is allowed to run in two distinct operation modes:

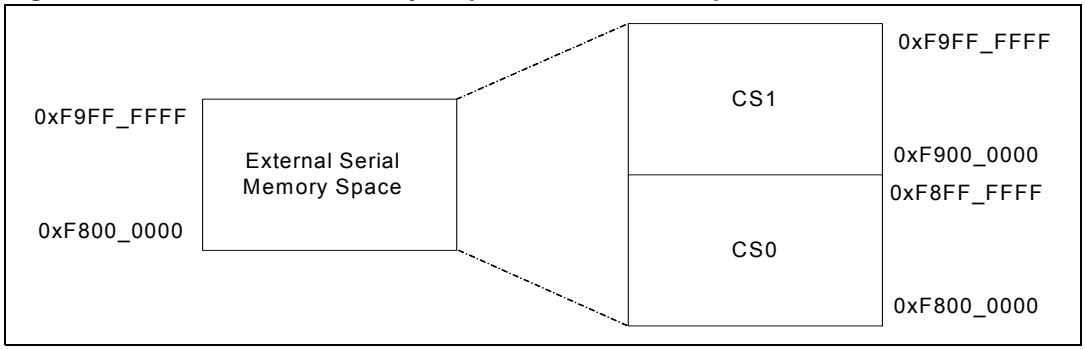
- Hardware mode (detailed in *Section 15.4.1*), clearing the SW bit in the SMI_CR1 register, *Section 15.8.3*.
- Software mode (detailed in *Section 15.4.2*), setting the SW bit in the SMI_CR1 register. Moreover, in both operation modes SMI can work:
- Either in normal mode, clearing the FAST bit in the SMI_CR1 register, with a frequency up to 20 MHz (19 MHz at power-on).
- Or in fast mode, setting the FAST bit in the SMI_CR1 register, with a frequency ranging from 20 MHz to 50 MHz.

15.4.1 Hardware mode

Hardware mode is intended to allow SMI to perform read/write requests from any AHB master, which can directly access the external serial memory.

In particular, external serial memory is mapped in AHB address space as shown in *Figure 26*,

Figure 26. External SPI memory map in AHB address space



In this mode, both the transmit register SMI_TR (*Section 15.8.6*) and the receive register SMI_RR (*Section 15.8.7*) must not be accessed. They are actually in charge of the SMI state machine to communicate with the selected external memory device, whenever an AHB master reads or writes an address into the memory.

At power-on reset, the SMI operates in hardware mode (allowing then boot phase from external memory, as explained in *Section 15.7*).

15.4.2 Software mode

Software mode is intended to allow any AHB master to access external serial memory by programming the internal SMI registers and reading them for memory replies. In this mode, direct transfer to/from external memory – that is, bypassing SMI registers - are not permitted to an AHB master.

In particular, software mode is used both to transfer any data or commands from transmit register (SMI_TR, *Section 15.8.6*) to external serial memory, and to read data directly in the receive register (SMI_RR, *Section 15.8.7*). The transfer actually starts setting the dedicated SEND bit in the SMI_CR2 register (*Section 15.8.4*).

Besides in software mode, application code being executed by the CPU cannot be fetched from external memory, because incompatibility between software and hardware mode. The code must be either hosted by internal memory or previously loaded from external memory while SMI is in hardware mode.

For example, software mode is used to erase Flash memories before writing. Indeed, memory erasing cannot be performed in Hardware mode due to incompatibilities in Flash devices from different vendors.

15.5 Data transfers

15.5.1 Read request

A read request from an AHB master to external SPI memory is served only if SMI is in hardware mode, and write burst mode is not enabled (*Section 15.5.3*). Otherwise, an error flag is set (ERF1 flag in the SMI_SR register, *Section 15.8.5*) and an ERROR response is sent back to the AHB master.

When a read request occurs in normal mode (that is, frequency up to 20 MHz), the SMI sends the following data sequence to the bank selected by the AHB address bits [24.25].

- Read data bytes instruction opcode (8'h03)
- 3- or 2-byte address (depending on the ADD_LENGTH bit of SMI_CR1 register, *Section 15.8.3*) from the MSB to the LSB,
- Then, clock is sent to the memory until the end of burst requested by the AHB master.

In contrast, when a read request occurs in fast mode (that is, frequency ranging from 20 MHz to 50 MHz), the following sequence is sent:

- Read data bytes at high speed instruction opcode (8'h0B),
- 3- or 2-byte address (depending on the ADD_LENGTH bit of SMI_CR1 register, *Section 15.8.3*) from the MSB to the LSB,
- 1 dummy byte (8'h00),
- Then, clock is sent to the memory until the end of burst requested by the AHB master.

The external memory bank remains selected as long as

- There is no external memory address jump,
- No new commands are sent to the SMI (such as write enable request, read status register command set, software mode or write burst mode, write request, bank disable, prescaler configuration change),
- No memory access error occurs.

Note: The memory bank also remains selected when the address rolls over from 0x00FF_FFFF to 0x0000.0000 within the same bank.

15.5.2 Write request

A write request from AHB master to external SPI memory is served only if SMI is in Hardware mode, otherwise an error flag is set (ERF1 flag in the SMI_SR register, *Section 15.8.5*). Wrapping bursts are not allowed as long as external SPI memories don't support them, and an ERROR response is sent back to the AHB master.

When a write request occurs, this request is forwarded to external memory only if both following conditions are fulfilled:

- At first, selected bank is in Write mode (corresponding bit in WM field of SMI_SR register is flagged). Otherwise, a dedicated error flag is set (the ERF2 flag in the SMI_SR register) and an ERROR response is sent back to the AHB master.

Note: To enable write mode, select the memory bank using the BS bits in the SMI_CR2 register (Section 15.8.4), and then set the WEN bit in the SMI_CR1 register (Section 15.8.3).

- Then, no write in progress. The WIP bit of external memory status register in the SMI_SR register (bit [0]) must be cleared (see *Section 15.8.5*). If this condition is not met, AHB is stalled until WIP = 1'b0.

When the 2 conditions above are met, the following data sequence is sent to the bank selected by AHB address bits [24-25]:

- Page program instruction opcode (8'h02, *Table 238*),
- 3- or 2-byte address (depending on the ADD_LENGTH bit of SMI_CR1 register, *Section 15.8.3*) from the MSB to the LSB,
- Then, all data bytes (from bit 7 to bit 0) are transferred, starting with address given in previous step and incrementing it to the last depending on the size of the Write request.

After a Write request has been sent, the WM bit in SMI_SR register is cleared and the read status register instruction (opcode 8'h05, *Table 238*) is automatically sent to this bank until no write in progress (WIP =1'b0).

Note: 1 Write capability must be used only if write in progress / busy bit of the external memory status register is located in bit 0. Otherwise the system will become locked.

2 When memory programming is finished, the WCF bit in the SMI_SR register is set and an interrupt is generated if the enabling WCIE bit in the SMI_CR1 register is set.

3 In order to send a Write request to a bank other than the one under programming, the software must wait for WIP = 1'b1, otherwise the error ERF2 would be generated due to non incrementing address. The bank under programming phase must not be disabled in order to write to another one.

15.5.3 Write burst mode

Write burst mode (WBM bit set in SMI_CR1 register, *Section 15.8.3*) enables to keep selected the external SPI memory after an AHB write request (see above). In this case, the next AHB write request should point to the next incremented address and should have the same size (byte, half-word or word). Otherwise, an error flag is set (ERF2 flag in the SMI_SR register, *Section 15.8.5*) and an ERROR response is sent back to the AHB master.

Note: A memory access error (ERF1 or ERF2) results in both release of chip select and start of the external memory page program.

Disabling the write burst mode (that is, clearing the WBM bit in SMI_CR1 register), the next incrementing AHB write request should be sent to external memory if it occurs before the end of the previous serial transfer. Otherwise, an error flag is set (ERF2 flag in the SMI_SR register) and an ERROR response is sent back to the AHB master.

Consequently, it is mandatory to enable write burst mode in order to perform several Write requests which are not sent in the same AHB incrementing burst. If WBM is cleared and no other write request occurs, the external memory selection is released after sending the data and the external memory page program cycle starts.

Note: Read requests to external memory are not allowed in write burst mode, otherwise an error flag is set (ERF1 flag in the SMI_SR register) and an ERROR response is sent back to the AHB master.

The external SPI memory is released by either disabling write burst mode (clearing WBM bit) or disabling the bank, and the external memory page program cycle starts. If bank is enabled, read status register instruction is automatically sent to this bank until WIP = 1'b0. (see *Section 15.8.5*

15.5.4 Read while write mode

If a read request occurs for the bank which is in programming phase, the AHB bus is stalled until no write in progress (WIP = 1'b0). (please refer to *Section 15.8.5* for major detail on WIP bit).

If a read request occurs for another bank, the read status register sequence is stopped, then the read request is served and, finally, the read status register sequence is sent again to the memory bank being programmed.

It follows that during a read while write, the selected external SPI memory is released after the read command, in order to send the read status register sequence.

15.5.5 Erase and write status register

In case of serial Flash, an erase may be necessary before writing. Due to incompatibility between different serial Flash vendors, erase and write status register can be done only in software mode.

It is mandatory to send previously the write enable instruction through Software mode only (that is, setting the WEN bit in SMI_CR2 register, *Section 15.8.4*), in order to avoid corruption of the WM bit in the SMI_SR register (*Section 15.8.5*). Indeed, the end of either internal Flash erase or write status register cannot be checked by hardware mode, preventing generation of write complete interrupt. On the other hand, WIP bit can be checked by continuously sending a read status register command.

15.6 Timings

The memory clock (`SMI_CK`) is generated by SMI through its programmable prescaler unit (*Section 15.3.1*), as depicted in *Figure 25*.

The incoming AHB bus frequency fAHB (`HCLK signal`) is divided by the value stored in the PRESC field of SMI_CR1 register (*Section 15.8.3*), resulting in the SMI clock frequency f_{SMI_CK}:

- $f_{SMI\_CK} = f_{AHB}$ / (PRESC value)

that is,

- $t_{SMI\_CK} = t_{AHB}$ * (PRESC value)

being t_{SMI_CK} and t_{AHB} the clock period of the SMI clock and the AHB bus, respectively.

Note: If PRESC is an even value, high time and low time of SMI clock are both equal to half a t_{SMI_CK}. In contrast, in case PRESC is an odd value:

$t_{SMI\_CK,}$ high= [(PRESC - 1) / 2]

t_{SMI_CK}, low = [(PRESC + 1) / 2]

15.6.1 Latencies

Assuming that SMI is not busy by now, the nominal latency for a 32 bit single read to a non-incrementing serial Flash address, is:

- 73 t_{AHB} maximum, if PRESC = 1 (that is, $t_{AHB} = t_{SMI\_CK}$).
- (68 t_{SMI_CK} + 5 t_{AHB}) maximum, if PRESC > 1 (that is, $t_{AHB} \neq t_{SMI\_CK}$, and specifically $t_{SMI\_CK} > t_{AHB}$),

taking into account up to 9 clock periods in addition to 64 clock periods required to both send command to serial Flash memory (1-byte opcode + 3-bytes address) and receive back 32 bits.

- Besides, under the same assumption, the nominal latency for a 32 bit single write to a non-incrementing serial Flash address is:
- 5 t_{AHB} maximum, if PRESC = 1 (that is, $t_{AHB} = t_{SMI\_CK}$).
- (2 t_{SMI_CK} + 3 t_{AHB}) maximum, if PRESC > 1 (that is, $t_{AHB} \neq t_{SMI\_CK}$, and specifically $t_{SMI\_CK} > t_{AHB}$).

In case of AHB read burst transfers, the maximum latency for all transfers after the first is the same as data size, that is (32 t_{SMI_CK}) for a word transfer, (16 tS_{MI_CK}) for a half-word and (8

t_{SMI_CK}) for a byte, because of no mandatory extra commands (instruction opcode and address).

Moreover, for AHB write burst transfers, the maximum latency for the 2nd transfer is (data size + opcode + address bytes)

whereas, it is the same as data size for following transfers.

- However, nominal latency can be increased by:
- SMI transfer on going (read, write, read status register command or write enable command),
- Deselect time programming (field TCS in SMI_CR1 register), which adds (TCS + 1) · SMI_CK periods,
- Busy / Idle transfer on AHB bus,
- Fast Read (*Section 15.5.1*) which adds 1 dummy byte,
- Hold programming (field HOLD in SMI_CR1 register, *Section 15.8.3*),
- Boot delay time (*Section 15.7*),
- Frequency change,
- Programming on-going.

15.7 How to boot from external memory

The device allows an external boot from a serial Flash only located at bank0 (which is enabled after power-on reset). During the boot phase, the following instructions sequence is automatically sent to bank0:

- Release from deep power-down (opcode 8'hAB, *Table 238*), in order to be able to boot on this bank even if it was in deep power-down mode.
- 29 µs delay to ensure bank0 is successfully released.
- Read status register (opcode 8'h05), in order to check that bank0 is neither in write nor in erase cycle.
- Read data bytes (opcode 8'h03) at memory start location (that is, 32'h0) with a 19 MHz clock frequency.

Note: 1 All memory banks other than bank0 are disabled at reset and they must be enabled by setting dedicated BE bits in SMI_CR1 register (Section 15.8.3) before they can be accessed.

2 If an AHB request occurs while either the WEN bit or the RSR bit (both in SMI_CR2 register, Section 15.8.4) is set, the on-going command is first finished before the request from AHB is sent to the memory.

15.8 Programming model

15.8.1 External pin connection

Table 239. External pin connection

Signal	Ball
SMI_DATAIN	M13
SMI_DATAOUT	M14
SMI_CLK	N17
SMI_CS_0	M15
SMI_CS_1	M16

The SMI can be fully configured by programming a set of 32 bit wide registers (listed in *Table 240*) which can be accessed at the base address 0xFC00_0000

Note: All transfer to and from these registers must be 32 bit wide only. Any attempt to access with a different size will result an ERROR response.

Table 240. SMI registers summary

Name	Offset	Reset value	Description
SMI_CR1	0x000	32'h51	SMI control register 1.
SMI_CR2	0x004	32'h0	SMI control register 2.
SMI_SR	0x008	32'h0	SMI status register.
SMI_TR	0x00C	32'h0	SMI transmit register.
SMI_RR	0x010	32'h0	SMI receive register.

15.8.2 Register description

15.8.3 SMI_CR1 register

The SMI_CR1 (control register 1) is a RW register which is able (together with coupled SMI_CR2, *Section 15.8.4*) to configure the behavior of SMI. The SMI_CR1 bit assignments are given in *Table 241*.

Table 241. SMI_CR1 register bit assignments

Bit	Name	Reset value	Type	Description
[31:30]	Reserved	-	-	Read: undefined. Write: should be zero.
[29]	WBM	1'h0	RW	Write burst mode. Setting this bit, the write burst mode is enabled and selected external memory device remains active after an AHB write request. In contrast (bit cleared, default), selected memory device is released.

Table 241. SMI_CR1 register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[28]	SW	1'h0	RW	Software mode. Setting this bit, the software operation mode of SMI is enabled (*Section 15.4*), otherwise (bit cleared, default), the hardware operation mode is enabled.
[27:26]	ADD_LENGTH	2'h0	RW	Address length. This is a 2 bit field where each bit is associated to a specific external memory bank, specifically the LSB (bit [24]) refers to bank0. In particular, each bit states the length of the address following the instruction opcode issued by SMI to the relevant bank, according to encoding: 1'b0 = 3 bytes (default) 1'b1 = 2 bytes (for EEPROM compliance)
[25:24]	-	-	-	Not Used
[23:16]	HOLD	8'h0	RW	Clock hold period selection. This 8 bit field states the hold period (where SMI_CK is stopped while chip select remains active) between bytes as an integer number of SMI_CK periods (t_{SMI_CK}, *Section 15.6*).
[15]	FAST	1'h0	RW	Fast read mode. This bit provides for mode selection during reading operation. As specified in *Section 15.4*, setting this bit a clock frequency up to 50 MHz is available, otherwise (bit cleared) it is reduced to 20 MHz.
[14:08]	PRESC	7'h0	RW	Prescaler value. This 7 bit field allows to set the prescaler value used to generate the SMI_CK clock by adjusting the AHB bus fequency, as detailed in *Section 15.6*. Note: The SMI_CK frequency is actually changed after the completion of ongoing transfer.
[07:04]	TCS	4'h5	RW	Deselect time. This 4 bit field enables to configure the deselect time, that is the minimum interval lasting between release of chip select signal and next selection. That is, chip select signal remains released (not selected) for at least (TCS + 1) SMI_CK clock periods. Actual deselect time at power-on reset depends on TCS reset value (4'h5) and it is limited by the SMI_CK frequency at power-on reset, that is 19 MHz, resulting in t_{SMI_CK} = 52.6 ns. It follows that, at reset, t_{cs} = (5+1) · 52.6 ns = 316 ns. Note: FAST and TCS fields must be written at the same time as PRESC. All these values are taken into account after the completion of the ongoing transfer. Any check of the consistency among these three values has to be done by software.

Table 241. SMI_CR1 register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[03:02]				Not used
[01:00]	BE	2'h1	RW	Bank enable. This is a 2 bit field where each bit is associated to a specific external memory bank, specifically the LSB (bit [0]) refers to bank0. Setting a bit, the relevant memory bank is enabled. At power-on reset, all banks are disabled except bank0 (reset value 0x1) to allow booting from external memory, as explained in *Section 15.7*. Note: If any AHB master makes a request on a disabled bank (relevant bit cleared in BE field), an ERROR response is sent back to AHB master. In contrast, write enable, read status register and send commands are not sent if the bank is disabled, without any error message.

15.8.4 SMI_CR2 register

The SMI_CR2 (Control register 2) is a RW register which is able (together with coupled SMI_CR1, *Section 15.8.3*) to configure the behavior of SMI. The SMI_CR2 bit assignments are given in *Table 242*.

Table 242. SMI_CR2 register bit assignments

Bit	Name	Reset value	Type	Description
[31:14]	Reserved	-	-	Read: undefined. Write: should be zero.
[13:12]	BS	2'h0	RW	Bank select. This 2 bit field allows to select the external memory bank, according to encoding: 2'b00 = Bank0 2'b01 = Bank1 2'b10 = Not Implemented 2'b11 = Not Implemented Note: Only one bank can be accessed at a time, and the BS value is latched at the beginning of transfer.
[11]	WEN	1'h0	RW	Write enable command. Setting this bit, a write enable command is sent to the memory bank selected by the BS field. The WEN bit is then directly cleared by hardware as soon as the write enable command has been successfully sent. A write of 1'b0 has no effect. Note: The WEN bit must be used in software mode to send either a write or an erase command.

Table 242. SMI_CR2 register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[10]	RSR	1'h0	RW	Read status register command. Setting this bit, a read status register command is sent to the memory bank selected by the BS field. Result from memory is then loaded into the SMSR field of SMI_SR register (*Section 15.8.5*). The RSR bit is then directly cleared by hardware as soon as the read status register command has been successfully completed. A write of 1'b0 has no effect.
[09]	WCIE	1'h0	RW	Write complete interrupt enable. Setting this bit, it allows to enable the issue of an interrupt request when write complete event occurs. This event also results in setting the write complete flag (WCF) in the SMI_SR register (*Section 15.8.5*).
[08]	TFIE	1'h0	RW	Transfer finished interrupt enable Setting this bit, it allows to enable the issue of an interrupt request when software transfer complete event occurs. This event also results in setting the transfer finished flag (TFF) in the SMI_SR register (*Section 15.8.5*).
[07]	SEND	1'h0	RW	Send command. Setting this bit, the transfer to external memory starts according to data format defined by both REC_LENGTH and TRA_LENGTH fields of this register. A write of 1'b0 has no effect. Note: The WEN bit can be set by software (only if Software mode is enabled), and it is cleared by hardware only.
[06:04]	REC_LENGTH	3'h0	RW	Reception length. This 3 bit field is used to specify the number of bytes to be received from external memory, following a send command (setting SEND bit).
[03]	Reserved	-	-	Read: undefined. Write: should be zero.
[02:00]	TRA_LENGTH	3'h0	RW	Transmission length. This 3 bit field is used to specify the number of bytes to be transmitted to external memory, following a send command (setting SEND bit). Note: The REC_LENGTH and TRA_LENGTH fields must be set by software, and their values are latched at the beginning of software transfer.

Note: Interrupt request issued (IRQ9), will be the OR of the events enabled by WCIE and TFIE fields (See the Section 8.4: Interrupt connection table in the VIC chapter).

15.8.5 SMI_SR register

The SMI_SR (Status register) is a RO register which allows to retrieve the current status of SMI. The SMI_SR bit assignments are given in *Table 243*.

Table 243. SMI_SR register bit assignments

Bit	Name	Reset value	Type	Description
[31:16]	Reserved	-	-	Read: undefined.
[15:14]	WM	2'h0	RO	Write mode for selected bank. This 2 bit field report the write mode (*Section 15.5.2*) status for the four supported memory banks. Each bit is associated to a single bank (specifically the LSB, bit [12], refers to Bank0). A bit is set in case related bank is in write mode, that is, when a write enable command – opcode 8'h06 – is sent to the relevant memory bank. Note: The WM field is not cleared by instructions sent in software mode.
[13:12]	-	-	-	Not Used.
[11]	ERF1	1'h0	RO	Error flag 1: forbidden access. This bit is used to issue error flags concerning access to external memory. Specifically, if set ERF1 marks forbidden access to memory, that is: read/write access requested on disabled bank, read/write access requested in software mode, or read requests in write burst mode (bit WBM set in SMI_CR1 register, *Section 15.8.3*).
[10]	ERF2	1'h0	RO	Error flag 2: forbidden Write request. This bit is used to issue error flags concerning access to external memory. Specifically, if set ERF2 marks specific forbidden write request, that is: write requests when out of write mode (bit WM cleared in this register for relevant bank), size changed between two consecutive write requests, or address is not incremented. Note: Setting either ERF1 or ERF2, an ERROR response is sent back to AHB master on HRESP.
[09]	WCF	1'h0	RO	Write complete flag. This bit is set in case of write completion, that is when the WIP bit of SMSR is set to 1'b0 (stating the end of programming). After a write instruction, a read status register command (opcode 8'h05) is performed by hardware.

Table 243. SMI_SR register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[08]	TFF	1'h0	RO	Transfer finished flag. This bit is set when transfer with external memory is completed, that is after REC_LENGTH and TRA_LENGTH bytes (set in SMI_CR2 register, *Section 15.8.4*) have been received and transmitted, respectively. Besides, TFF is set when either the read status register (bit RSR in SMI_CR2) or the write enable (bit WEN in SMI_CR2) commands are finished.
[07:00]	SMSR	8'h0	RO	Memory device status register. This 8 bit field is used to store a copy of the external memory status register. This field is updated in 2 distinct ways: at first, when the RSR bit in SMI_CR2 register is set (SMSR is updated after the RSR sequence), and after a write request to a memory bank (SMSR is updated until the write cycle is finished, that is when bit [0] of this field is cleared). Note: This field is refreshed every 8 SMI_CLK periods.

15.8.6 SMI_TR register

The SMI_TR is the transmit register which is used by SMI to send either data or commands to external serial memory. In particular, SMI_TR is a 8 bit barrel shifter, where byte0 is sent first and then 8 bits are shifted before sending byte1 and so on. The SMI_TR bit assignments are given in *Table 244*.

This register can be written in software mode only (bit SW set in SMI_CR1 register, *Section 15.8.3*), and when actual transfer is not yet started (bit SEND cleared in SMI_CR2 register, *Section 15.8.4*).

Note: The SMI_TR is also used in hardware mode, but its content is not kept entering in this mode.

Table 244. SMI_TR register bit assignments

Bit	Name	Reset value	Description
[31:24]	Byte3	8'h00	Transmit register (8 bit barrel shifter).
[23:16]	Byte2	8'h00	
[15:08]	Byte1	8'h00	
[07:00]	Byte0	8'h00	

15.8.7 SMI_RR register

The SMI_RR is the receive register which is used by SMI to receive data from external serial memory. Received bytes from external memory are first placed in byte0, then in other next fields of SMI_RR until byte3. The SMI_RR bit assignments are given in *Table 245*.

This register must be read in software mode (bit SW set in SMI_CR1 register, *Section 15.8.3*) after transfer is finished (bit TFF set in SMI_SR register, *Table 243*), otherwise the register content is not valid.

Note: The SMI_RR is also used in Hardware mode, but its content is not kept entering in this mode.

Table 245. SMI_RR register bit assignments

Bit	Name	Reset value	Description
[31:24]	Byte3	8'h00	Receive register.
[23:16]	Byte2	8'h00	
[15:08]	Byte1	8'h00	
[07:00]	Byte0	8'h00	

16 BS_Watchdog timer

16.1 Overview

Within its basic subsystem, the device provides an ARM watchdog module. It consists of a 32 bit down counter with a programmable time-out interval that has the capability to generate an interrupt and a reset signal on timing out. The watchdog module is intended to be used to apply a reset to a system in the event of a software failure.

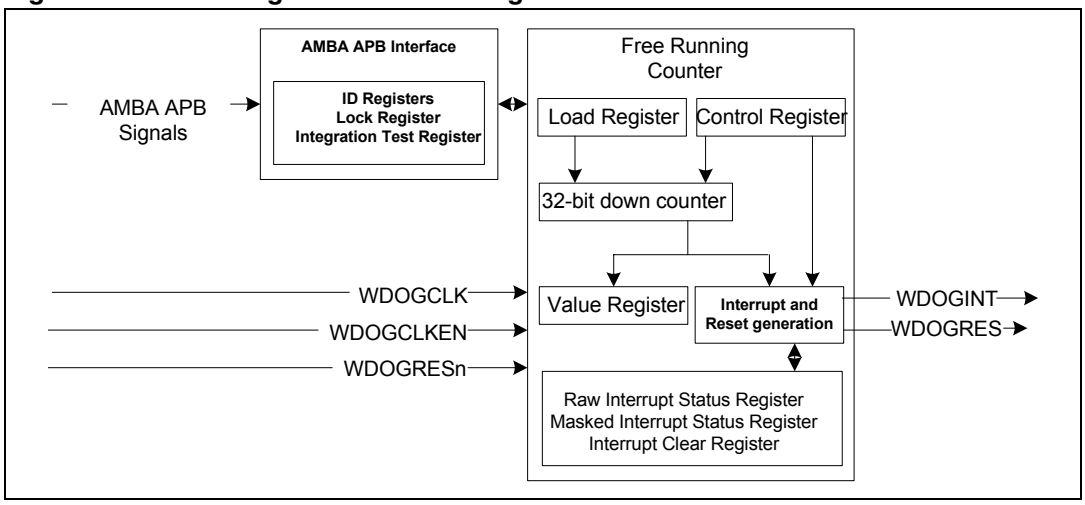
Main features of the Watchdog module are:

- 32 bit down counter with a programmable time-out interval.
- Separate watchdog clock with its clock enable for flexible control of the time-out interval.
- Interrupt output generation on time-out.
- Reset signal generation on time-out, if the interrupt from the previous time-out remains unserviced by software.
- Lock register to protect registers from being altered by runaway software.
- Identification registers that uniquely identify the watchdog module. These can be used by software to automatically configure itself.
- An APB slave allowing access to all registers.

16.2 Block diagram

Figure 27 shows a simplified block diagram of the watchdog module.

Figure 27. Watchdog module block diagram



16.3 Main functions description

16.3.1 AMBA APB interface

The AMBA APB interface block provides an APB slave which allows to accesses to all registers in the watchdog module.

In particular, the lock register (WdogLock, *Section 16.5.9*) controls the enabling of write accesses to all the other registers in order to ensure software cannot unintentionally disable the watchdog module operation.

16.3.2 Free running counter

The free running counter block contains the 32 bit down counter functionality (including related registers, *Section 16.5.2*), and the logic to generate the interrupt and reset signal outputs.

The counter and the interrupt/reset logic are clocked independently, as detailed in *Section 16.4 on page 314*.

16.4 Clock signals

The watchdog module uses two different input clocks:

- The clock of the APB bus (PCLK signal), which is used to clock the Watchdog module registers through APB bus.
- The external WDOGCLK signal which, in conjunction with its clock enable, WGDOGCLKEN, is used to clock the Watchdog module counter and its associated interrupt and reset generation logic. In particular, the watchdog counter only decrements on a rising edge of WDOGCLK when WDOGCLKEN is HIGH.

The following constraints must be observed in the relationship between the two clocks:

- The rising edge of WDOGCLK must be synchronous and balanced with the rising edge of PCLK,
- The WDOGCLK frequency cannot be greater than the PCLK frequency.

From the constraints above and depending on the relationship between WDOGCLK and WDOGCLKEN, the watchdog module counter is decremented on different ways summarized in *Table 246*.

Table 246. Watchdog module counter decremented

Clocks	WDOGCLKEN	Behaviour
WDOGCLK equals PCLK	HIGH	The counter is decremented on every WDOGCLK edge
	Pulsed	The counter is decremented on every second WDOGCLK rising edge
WDOGCLK less than PCLK	HIGH	The counter is decremented on every WDOGCLK rising edge
	Pulsed	The counter is decremented on every second WDOGCLK rising edge

16.5 Programming model

16.5.1 Register map

The watchdog module can be fully configured by programming its 32 bit wide registers which can be accessed at the base address 0xFC88_0000. Watchdog registers can be logically arranged in two main groups:

- control and status registers (listed in *Table 247*), which allow to control the Watchdog module configuration and to get its status.
- identification registers (listed in *Table 248*), namely eight 8 bit RO registers reporting watchdog module-specific information (part number, revision number and so on). Refer to ARM technical documentation for further details.

Table 247. Watchdog control and status registers summary

Name	Offset	Type	Reset value	Description
WdogLoad	0x00	RW	32'hFFFFFFFF	Load register
WdogValue	0x04	RO	32'hFFFFFFFF	Value register
WdogControl	0x08	RW	32'h0	Control register
WdogIntClr	0x0C	WO	-	Interrupt clear register
WdogRIS	0x10	RO	32'h0	Raw interrupt status register
WdogMIS	0x14	RO	32'h0	Masked interrupt status register
-	0x0018 to 0xBFC	-	-	Reserved
WdogLock	0xC00	RW	32'h0	Lock register
-	0xC04 to 0xEFC	-	-	Reserved
-	0xF00	-	-	Reserved (for test purpose only)
-	0xF04	-	-	Reserved (for test purpose only)
-	0xF08-0xFDC	-	-	Reserved

Table 248. Watchdog identification registers summary

Name	Offset	Width	Type	Reset value	Description
WdogPeriphID0	0xFE0	8	RO	8'h05	Peripheral identification registers
WdogPeriphID1	0xFE4	8	RO	8'h18	
WdogPeriphID2	0xFE8	8	RO	8'h14	
WdogPeriphID3	0xFEC	8	RO	8'h00	
WdogPCellID0	0xFF0	8	RO	8'h0D	Identification registers
WdogPCellID1	0xFF4	8	RO	8'hF0	
WdogPCellID2	0xFF8	8	RO	8'h05	
WdogPCellID3	0xFFC	8	RO	8'hB1	

16.5.2 Register description

16.5.3 WdogLoad register

The WdogLoad is a RW register that contains the value from which the counter is to decrement. When this register is written to, the counter is immediately restarted from the new value. The minimum valid value for WdogLoad is 32'h1.

Note: 1 If WdogLoad is set to 32'h0 then an interrupt is generated immediately.

2 The WdogLoad register must be programmed with the desired time-out interval before the watchdog module is enabled (by setting the INTEN bit of the WdogControl register, Section 16.5.5).

16.5.4 WdogValue register

The WdogValue is a RO register that gives the current value of the decrementing counter.

16.5.5 WdogControl register

The WdogControl is a RW register which allows the software to control the watchdog module. The WdogControl register bit assignments are given in *Table 249*.

Table 249. WdogControl register bit assignments

Bit	Name	Reset value	Description
[31:02]	Reserved	-	Read: undefined. Write: should be zero.
[01]	RESEN	1'h0	Enable watchdog module reset output. This bit acts as a mask for the reset output of the watchdog module: it is set to enable the reset, and it is cleared to disable the reset. Note: If enabled (RESEN set to 1'b1), the reset output is asserted if the interrupt (raised when the counter reaches zero) is not cleared by software (writing any value to WdogIntClr register, *Section 16.5.6*) before the counter next reaches zero. After reset, the counter stops.
[00]	INTEN	1'h0	Enable the interrupt event. Setting this bit, the counter and the interrupt are enabled. In this case, the counter is re-loaded with the WdogLoad register value and it starts to decrement according to behaviour detailed in *Section 16.4*. When the counter reaches zero an interrupt is generated. Clearing this bit, the counter and the interrupt are disabled.

16.5.6 WdogIntClr register

A write of any value to the WO WdogIntClr (interrupt clear) register clears the watchdog module interrupt. Then the counter is re-loaded with the value in the WdogLoad register and another count down sequence starts.

16.5.7 WdogRIS register

The WdogRIS (raw interrupt status) is a RO register indicates the raw interrupt status from the counter (before masking by WdogControl register). The WdogRIS bit assignments are given in *Table 250*.

Table 250. WdogRIS register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read: undefined.
[00]	WDOGRIS	1'h0	If set, it indicates that an interrupt has been raised by the Watchdog counter reaching zero.

16.5.8 WdogMIS register

The WdogMIS (masked interrupt status) is a RO register indicates the masked interrupt status from the counter (after masking by the Wdogcontrol register). The WdogMIS bit assignments are given in *Table 251*.

Table 251. WdogMIS register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read: undefined.
[00]	WDOGMIS	1'h0	Masked interrupt status. The value of this bit is the logical AND of the raw interrupt status (WDOGRIS bit of the WdogRIS register) with the INTEN bit of the WdogControl register.,It is the same value that is passed to the interrupt output of the Watchdog module.

16.5.9 WdogLock register

The WdogLock is a RW register allows to enable/disable write-access to all other registers. This is to prevent software from disabling the Watchdog module operation. The WdogLock bit assignments are given in *Table 252*.

Table 252. WdogLock register bit assignments

Bit	Name	Reset value	Description
[31:01]	WDOGLOCK	32'h0	Write access enable. Writing 32'h1ACCE551 to this register enables write access to all other registers. Writing any other value disables write access to all other registers. A read from this register returns the lock status rather than the actual value: 32'h00000000 = Write access to all others registers is enabled (not locked). 32'h00000001 = Write access to all others registers is disabled (locked).
[00]	Reserved	-	Read: undefined.

17 BS_General purpose timers

17.1 Overview

SPEAr300 provides three general purpose timers (GPTs) acting as APB slaves (one local timer in the CPU subsystem and two other timers in the basic subsystem).

Each GPT consists of 2 channels, each one made up of a programmable 16 bit counter and a dedicated 8 bit timer clock prescaler. The programmable 8 bit prescaler performs a clock division by 1 up to 256, and different input frequencies can be chosen through SPEAr300 configuration registers (so we can synthesize, for instance, a frequency range from 3.96 Hz to 48 MHz).

Enabling a GPT (setting the ENABLE bit in TIMER_CONTROL register, *Section 17.2.4*), its counter is firstly cleared and then it starts incrementing. When the counter reaches a pre-set compare value (in TIMER_COMPARE register, *Section 17.2.6*), two different modes of operation are available (setting the MODE bit in TIMER_CONTROL register, *Section 17.2.4*):

- Auto-reload mode, an interrupt source is activated, the counter is automatically cleared and then it restarts incrementing.
- Single-shot mode, an interrupt source is activated, the counter is stopped and the GPT is disabled.
- Capture Mode, which is used for measurement of input timing signals. when a rising transition occurs at the CPTRx(x=1,2), the actual counter value is stored into the rising edge capture register and a dedicated interrupt source is activated. In the same way, when a falling edge transition occurs at the CPTRx (=1,2) actual counter value is stored into the falling edge capture register and an other interrupt source is also activated. The processor can read the value stored in the two capture registers and compute the duration of the rising to falling edge (or vice versa) time interval.

17.2 Programming model

17.2.1 External pin connection

Table 253. External pin connection

Subsystem	Basic address	Signals	Ball	Usage	Note
CPU	0xF000_0000	-	-	-	Not available

Table 253. External pin connection (continued)

Subsystem	Basic address	Signals	Ball	Usage	Note
Basic	0xFC80_0000	TMR_CLK1	E9	Output signal which toggles TIMER generates an interrupt. (OUTPUT generated for both TIMER/CAPTURE MODE)	See PL_GPIO Sharing Schemes to verify the availability
		TMR_CLK2	A10		
		TMR_CPTR1	C10	These Input pins are used to receive the signals for which the measurement of timing is done. (Used only in CAPTURE MODE)	
		TMR_CPTR2	B11		
	0xFCB0_0000	TMR_CLK3	B10	Output signal which toggles TIMER generates an interrupt. (OUTPUT generated for both TIMER/CAPTURE MODE)	
		TMR_CLK4	A11		
		TMR_CPTR3	C11	These Input pins are used to receive the signals for which the measurement of timing is done. (Used only in CAPTURE MODE)	
		TMR_CPTR4	A12		

Figure 28. GPT block diagram

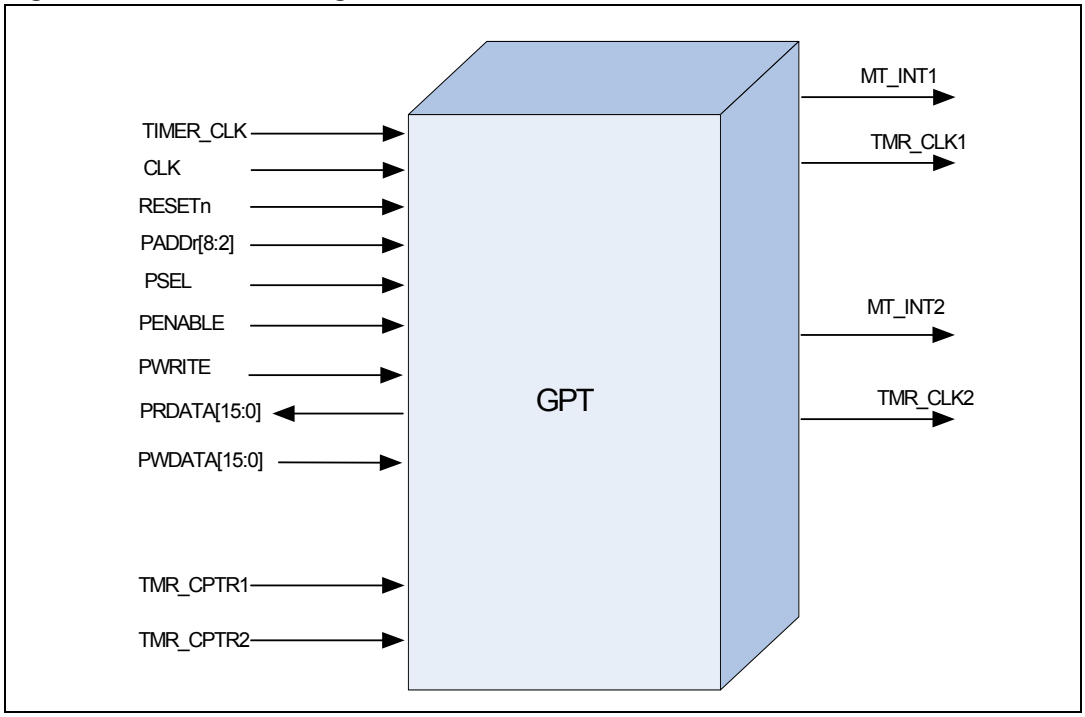


Table 254. GPT interface signal description

Pin Name	Type	Source/Destination	Description
CLK	In	Clock root	APB system (bus) clock. This clock times all the bus transfers. Synchronous logic inside the GPT id rising edge clock triggered.
RESETn	In	Reset block	Synchronous reset of the GPT, all internal registers are cleared when this input is driven low.
PADDR[8:2]	In	APB Bridge	Standard APB address bus.
PSEL	In	APB Bridge	Standard APB psel signal.
PENABLE	In	APB Bridge	Standard APB penable signal.
PWRITE	In	APB Bridge	Standard APB write signal.
PRDATA [15:0]	Out	APB Bridge	Standard APB prdata signal.
PWDATA[15:0]	In	APB Bridge	Standard APB pwdata signal.
TIMER_CLK	In	Clock root	Timer clock (Refer to Bit 11 & 12 in *Table 218: SSPIMSC register bit assignments*).
TMR_CPAT1	In	External Pin	Asynchronous signal provided for the measurement of timing signals in Timer1.
TMR_CAPT2	In	External Pin	Asynchronous signal provided for the measurement of timing signals in Timer2.
MT_INT1	Out	Interrupt Controller	Active low interrupt to Interrupt Control block from Timer1.
TMR_CLK	Out	Interrupt Controller	Clock which toggles each time the MT_INT1 goes active.
MT_INT2	Out	Interrupt Controller	Active low Interrupt to Interrupt Control block from Timer2.
TMR_CLK	Out	Interrupt Controller	Clock which toggles each time the MT_INT2 goes active.

17.2.2 Register map

Programming a set of 16 bit wide registers can configure each GPT. The registers of the three GPTs are mapped in memory by couples, namely:

- The local timer in the CPU subsystems, which can be accessed at the base address 0xF000_0000.
- The two timers in the basic subsystem, which can be accessed at the base addresses 0xFC80_0000 and 0xFCB0_0000.

The registers are same for both the couples of GPTs and are listed in *Table 255*. *Section 17.2.3* describes the registers of a generic GPT.

Table 255. Couple of GPTs registers summary

Name	Offset	Type	Reset Value	Description
TIMER_CONTROL1	0x0080	RW	16'h0000	Control register of 1st timer in the couple (GPT0 or GPT2).
TIMER_STATUS_INT_ACK1	0x0084	RW	16'h0000	Status register of 1st timer.
TIMER_COMPARE1	0x0088	RW	16'hFFFF	Compare register of 1st timer.
TIMER_COUNT1	0x008C	RO	16'h0000	Count register of 1st timer.
TIMER_REDG_CAPT1	0x0090	RO	16'h0000	Rising edge capture register of 1st timer.
TIMER_FEDG_CAPT1	0x0094	RO	16'h0000	Falling edge capture register of 1st timer.
TIMER_CONTROL2	0x0100	RW	16'h0000	Control register of 2nd timer in the couple (GPT1 or GPT3).
TIMER_STATUS_INT_ACK2	0x0104	RW	16'h0000	Status register of 2nd timer.
TIMER_COMPARE2	0x0108	RW	16'hFFFF	Compare register of 2nd timer.
TIMER_COUNT2	0x010C	RO	16'h0000	Count register of 2nd timer.
TIMER_REDG_CAPT2	0x0110	RO	16'h0000	Rising edge capture register of 2nd timer.
TIMER_FEDG_CAPT2	0x0114	RO	16'h0000	Falling edge capture register of 2nd timer.

17.2.3 Register description

17.2.4 Timer_control register

The Timer_Control register bit assignments are given in *Table 256*.

Table 256. Timer_Control register bit assignments

Bit	Name	Reset value	Description
[15:11]	Reserved	-	Read: undefined. Write: should be zero.
[10]	REDGE_INT	1'h0	If set, it enables interruption on a rising edge capture.
[09]	FEDGE_INT	1'h0	If set, it enables interruption on a falling edge capture.
[08]	MATCH_INT	1'h0	If set, it enables interruption when comparator matches.
[07:06]	CAPTURE	2'h0	Capture mode. This 2 bit field indicates the mode of capture, according to encoding. 2'b00 = No capture. 2'b01 = Capture in rising edge. 2'b10 = Capture in falling edge. 2'b11 = Capture in bit edges.

Table 256. Timer_Control register bit assignments (continued)

Bit	Name	Reset value	Description
[05]	ENABLE	1'h0	Timer enable. Setting this bit, the GPT is enabled. Once enabled, an initialization phase is performed before starting to count, and capture registers (TIMER_REDG_CAPT and TIMER_FEDG_CAPT) and counter register (TIMER_COUNT) are cleared. Clearing this bit, the GPT is disabled, and capture as well as counter registers are frozen. After reset the GPT is disabled and all interrupt sources are masked.
[04]	MODE	1'h0	Operation mode. This bit allows to select the operation mode of the GPT, according to encoding. 1'b0 = Auto-reload mode. 1'b1 =Single-shot.
[03:00]	PRESCALER	4'h0	Prescaler configuration. This 4 bit field controls the prescaler configuration, according to encoding.

Table 257. PRESCALER configuration

Value	Division Scale	Frequency [MHz]	Resolution [ns]	Max time [ms]
4'b0000	1	66.5	15.04	0.971
4'b0001	2	33.25	30.08	1.971
4'b0010	4	16.625	60.15	3.942
4'b0011	8	8.313	120.30	7.884
4'b0100	16	4.156	240.20	15.768
4'b0101	32	2.078	481.20	31.538
4'b0110	64	1.039	962.41	63.071
4'b0111	128	0.520	1924.81	126.143
4'b1000	256	0.260	3849.62	252.285
4'b1001 to 4'b1111	Not allowed	Not allowed	Not allowed	Not allowed

Note: This table just illustrates the use of prescaler and its different value on a particular Input Frequency. The table in this shows the value with 66.5 MHz as Timer clock.Timer Max period = (65536 xTimer Resolution).

17.2.5 TIMER_STATUS_INT_ACK register

The TIMER_STATUS_INT_ACK (Status and Interrupt Acknowledge Timer) is a RW register which indicates the raw interrupt sources status, prior to any masking. The TIMER_STATUS_INT_ACK bit assignments are given in *Table 258*.

Table 258. TIMER_STATUS_INT_ACK register bit assignments

Bit	Name	Reset value	Description
[15:03]	Reserved	13'h0	Read undefined. Write: should be zero.
[02]	REDGE	1'h0	Rising edge capture. Reading this bit as 1'b1, it means that a rising edge has been detected on the capture input and an interrupt is raised. Writing 1'b1, the interrupt source is cleared, whereas there is no effect when writing 1'b0.
[01]	FEDGE	1'h0	Falling edge capture. Reading this bit as 1'b1, it means that a falling edge has been detected on the capture input and an interrupt is raised. Writing 1'b1, the interrupt source is cleared, whereas there is no effect when writing 1'b0.
[00]	MATCH	1'h0	Match status. Reading this bit as 1'b1, it means that a match has occurred in the compare unit and an interrupt is raised. Writing 1'b1, the interrupt source is cleared, whereas there is no effect when writing 1'b0.

Note: Independently by the timer activity, pending interruptions remain active until they have been acknowledged (writing a 1'b1 in the relevant bit) and they are not automatically deactivated when the timer is disabled or enabled. It is therefore strongly recommended to acknowledge all active interrupt sources before enabling a timer.

17.2.6 TIMER_COMPARE register

The TIMER_COMPARE is a RW register allows the software to program the timer period. The TIMER_COMPARE bit assignments are given in *Table 259*.

Table 259. TIMER_COMPARE register bit assignments

Bit	Name	Reset value	Description
[15:00]	COMPARE_VALUE	16'hFFFF	Compare value.

The COMPARE_VALUE is expressed as an integer number of clock periods (where the input clock of the timer is the output of the prescaler) ranging from the 16'h0001 minimum value to the 16'hFFFF maximum value (default, to be intended as free-running timer in auto-reload mode). When the counter reaches the COMPARE_VALUE, the GPT behaves depending on the operation mode (auto-reload or single-shot).

Note: 1 In auto-reload mode, when the counter reaches the COMPARE_VALUE, it is cleared and restarts:

TIMER_PERIOD = (COMPARE_VALUE - 1) x COUNTER_PERIOD + 2 TIMER_CLOCK periods.

2 COUNTER_PERIOD is the period of the timer's input clock (i.e. the prescaler's output).

17.2.7 TIMER_COUNT register

The TIMER_COUNT is a RO register indicates the current counter value. The TIMER_COUNT bit assignments are given in *Table 260*.

Table 260. TIMER_COUNT register bit assignments

Bit	Name	Reset value	Description
[15:00]	CONT_VALUE	16'h0000	Current counter value.

17.2.8 TIMER_REDG_CAPT register

The TIMER_REDG_CAPT (timer rising edge capture) is a RO register which is used to store the current value of the timer counter when a rising edge occurs. When a capture has occurred, the REDGE bit is set in the TIMER_STATUS_INT_ACK register (*Section 17.2.5*) and the corresponding interrupt, if enabled (REDGE_INT bit set to 1'b1 in TIMER_CONTROL register, *Section 17.2.4*), is raised.

The TIMER_REDG_CAPT bit assignments are given in *Table 261*.

Table 261. TIMER_REDG_CAPT register bit assignments

Bit	Name	Reset value	Description
[15:00]	COUNT_VALUE_REDGE	16'h0000	Current value of timer when a rising edge occurs

Note: In the interrupt service routine, the capture register must be read before the next capture event occurs: if not the current capture value will be overwritten by the next one.

17.2.9 TIMER_FEDG_CAPT register

The TIMER_FEDG_CAPT (timer falling edge capture) is a RO register which is used to store the current value of the counter when a falling edge occurs. When a capture has occurred, the FEDGE bit is set in the TIMER_STATUS_INT_ACK register (*Section 17.2.5*) and the corresponding interrupt, if enabled (FEDGE_INT bit set to 1'b1 in TIMER_CONTROL register, *Section 17.2.4*), is raised. The TIMER_FEDG_CAPT bit assignments are given in *Table 262*.

Table 262. TIMER_FEDG_CAPT register bit assignments

Bit	Name	Reset value	Description
[15:00]	CONT_VALUE_FEDGE	16'h0000	Current value of timer when a falling edge occurs

Note: In the interrupt service routine, the capture register must be read before the next capture event occurs: if not the current capture value will be overwritten by the next one.

18 BS_General purpose input/output (GPIO)

18.1 Overview

Within its Basic Subsystem, SPEAr300 provides a **General Purpose Input/Output (GPIO)** providing 6 programmable inputs or outputs. Each input/output can be controlled through an APB interface.

Main features of the GPIO are:

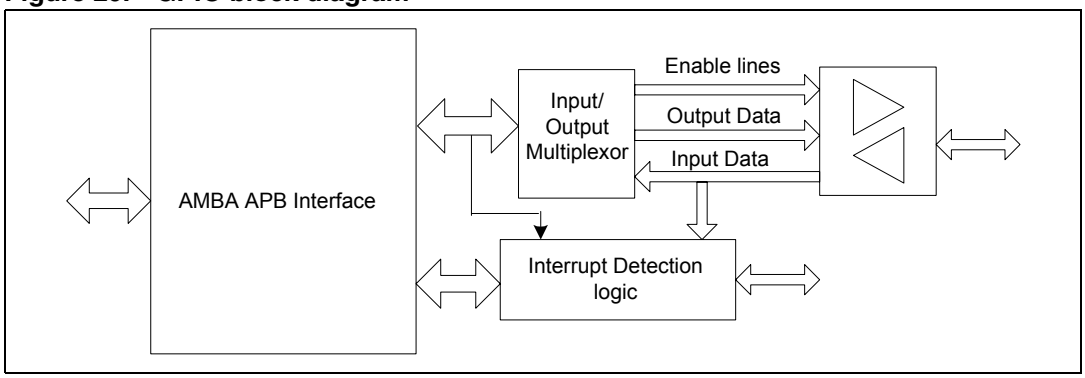
- Six individually programmable input/output pins (default to input at reset)
- 2x GPIO dedicated for SPI chip select
- An APB slave acting as control interface
- Programmable interrupt generation capability on any number of pins.
- Bit masking in both read and write operation through address lines.

18.2 Functional description

18.2.1 Block diagram

Figure 29 shows the block diagram of GPIO.

Figure 29. GPIO block diagram



18.2.2 Signal interfaces

The GPIO directly interfaces with the signals summarized in *Table 263: GPIO signal interface*. A functional diagram of these signal interfaces is given in *Figure 30*.

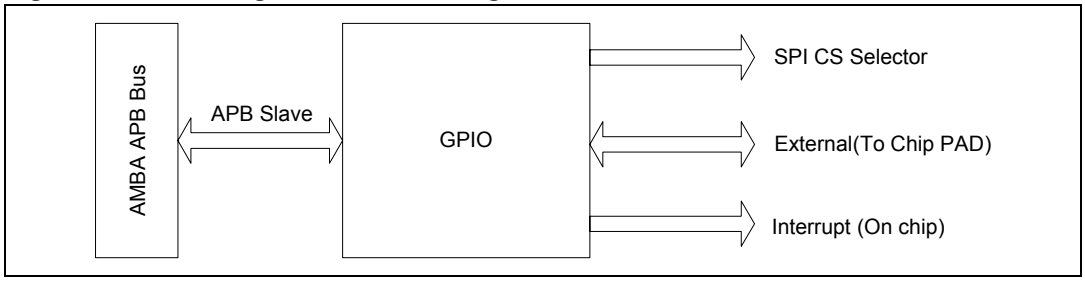
Table 263. GPIO signal interface

Group	Signal name	Direction	Size (bit)	Description
External (to chip pads)	nGPEN	Output	6	Output pad enable signal (active low).
	GPOUT	Output	6	Output pad data signal driver.
	GPIN	Input	6	Input data from chip pad.

Table 263. GPIO signal interface (continued)

Group	Signal name	Direction	Size (bit)	Description
Interrupt (on-chip)	GPIOMIS	Output	6	Masked interrupt signals, to interrupt controller.
	GPIOINTR	Output	1	Combined OR version of **GPIOMIS**, to interrupt controller.
APB Slave	-	Input/Output	-	See AMBA specification.

Figure 30. GPIO signal interfaces diagram



18.3 Main functions description

18.3.1 APB slave interface

The *APB slave interface* block allows to connect the GPIO to the AMBA APB bus.

In particular, the APB Slave Interface properly decodes read and write command on APB bus providing access to GPIO internal registers (data, data direction, mode control and interrupt). Moreover, these registers are memory-mapped in the APB Slave Interface.

18.3.2 Interrupt detection logic

The *interrupt detection* logic is the GPIO block which allow to generate mask-programmable interrupts based on either the signal level or the transitional value (edge) of any of GPIO lines.

As depicted in *Figure 29*, this block interfaces with both the interrupt control registers (*Table 266*) hosted by the APB slave and the input signals from pad **(GPIN[5:0]).** Depending on registers configuration (see *Section 18.4.2: Control interrupt generation* for details) and actual input signals features, a GPIO interrupt signal **(GPIOINTR)** is generated by the Interrupt Detection Logic block as the OR combination of all the GPIO masked interrupt lines.

The resulting combined **GPIOINTR** signal can be then used to indicate to an external interrupt controller that an interrupt occurred in one or more of the GPIO lines. Additional output signals **(GPIOMIS[5:0])** are also generated by the interrupt detection logic block, reflecting the status of each single masked interrupt lines. Provisional of individual outputs as well as combined interrupt output allows to use either a global interrupt service routine (trapping the **GPIOINTR** signal) or modular device drivers (looking at **GPIOMIS[5:0])** to handle GPIO interrupts.

18.3.3 Mode control

Each GPIO line can be controlled through APB interface.

The data direction is controlled by the data direction register (GPIODIR, *Section 18.5.3*). Data writing and reading are performed through APB interface, according to operation detailed in *Section 18.4*.

18.4 How to

18.4.1 Read from and write to input/output lines

So that independent software drivers can set their GPIO bits without affecting any other pins in a single write operation, the APB address bus **(PADDR)** is used as a mask on read/write operations.

The GPIO data register (GPIODATA) effectively covers 64 locations in the address space, that is the same register appears at 64 different locations (with offset ranging from 0x00 to 0xFC with respect to base address). To access these locations, the 6 bit subset of the APB address bus is used, according to the following rules:

- During a **write operation** to GPIODATA register: a data bit of the GPIODATA register is altered only if the associated address bit in **PADDR[9:2]** is set, otherwise it is left unchanged.
- During a **read operation** from GPIODATA register: a data bit of the GPIODATA register is read only if the associated address bit in **PADDR[9:2]** is set, otherwise a zero is returned regardless of its state.

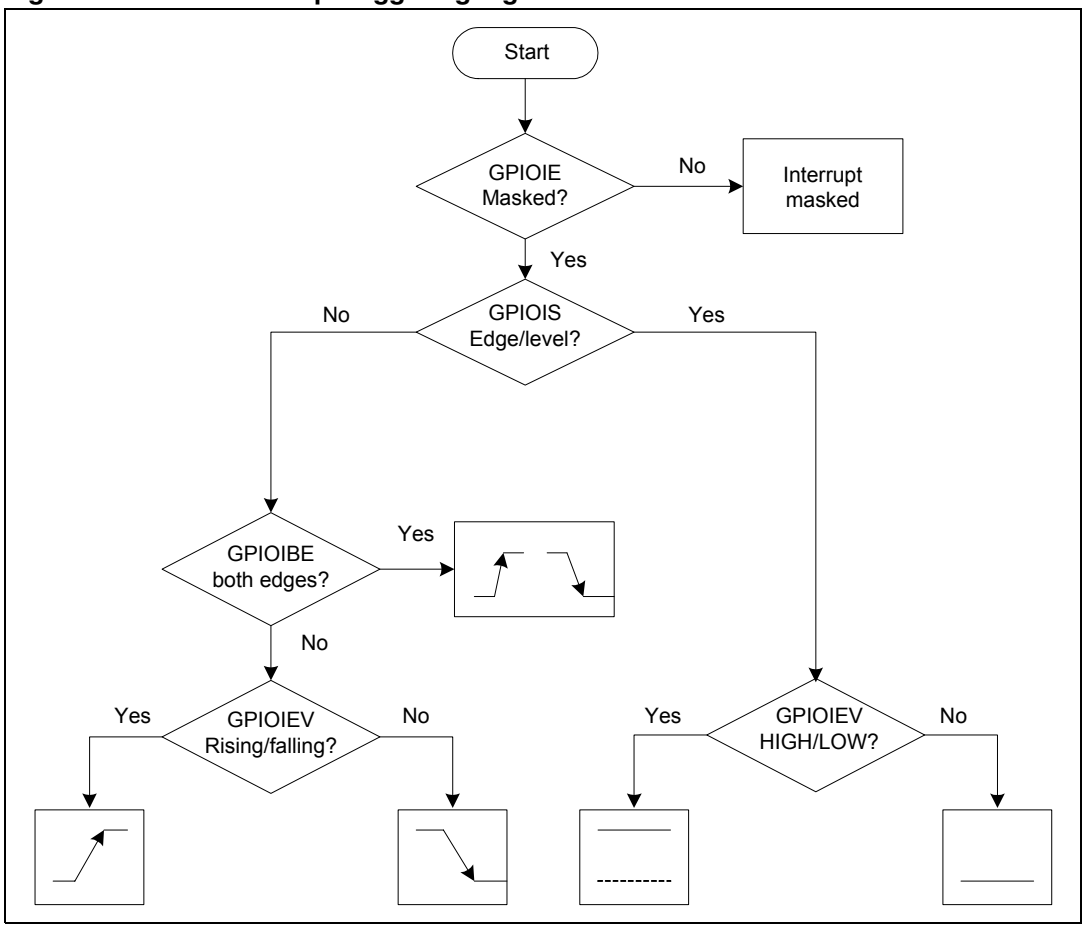
18.4.2 Control interrupt generation

The GPIO interrupt generation capability is fully controlled by a set of seven registers located in the APB slave interface.

These registers allows to select, for each single pin, the interrupt source (the edge or the level of signal on that pin), the event (rising/falling edge or high/low signal level) which triggers the interrupt and any interrupt masking.

Figure 31 shows how the three main interrupt control registers (namely GPIOIS, GPIOIBE and GPIOIEV) should be set to select an interrupt source event for a single pin. Please refer to *Section 18.5.5*. and following for detailed description of these registers.

Figure 31. GPIO interrupt triggering logic



Note: 1 For level detection case, it is assumed that an external source holds the level constant for the interrupt to be recognized by the processor.

2 Interrupt control registers must be programmed when corresponding interrupts are not enabled, in order to avoid spurious interrupts to be generated.

18.5 Programming model

18.5.1 Register map

The GPIO can be fully configured by programming its 6 bit wide registers which can be accessed through the APB slave interface at the base address 0xFC98_0000.

GPIO registers can be logically arranged:

- **Data direction register** (listed in *Table 264*), for pins configuration as input or output
- **Data register** (listed in *Table 265*), used to read value on those GPIO lines configured as inputs, or to write a value on those GPIO lines configured as outputs.

Note: The same data register appears at 64 locations in Memory Map (with offset ranging from 0x00 to 0xFC), allowing to use the address bus [9:2] as an additional bit masking feature.

- **Interrupt Control Registers (**listed in *Table 266*), for interrupt generation configuration.
- **Identification Registers** (listed in *Table 267.*), containing peripheral & BIOS information

Table 264. GPIO data direction register

Name	Offset	Type	Width(bit)	Reset Value	Description
GPIODIR	0x400	RW	6	6'h0	Data Direction

Table 265. GPIO data register

Name	Offset	Type	Width(bit)	Reset Value	Description
GPIODATA	0x000(1)	RW	8	8'h0	Data

1. For the first data register, but up to 0xFC for the 64th (See Note above).

Table 266. GPIO interrupt control registers summary

Name	Offset	Type	Width(bit)	Reset Value	Description
GPIOIS	0x404	RW	6	6'h0	Interrupt Sense.
GPIOIBE	0x408	RW	6	6'h0	Interrupt Both Edges.
GPIOIEV	0x40C	RW	6	6'h0	Interrupt Event.
GPIOIE	0x410	RW	6	6'h0	Interrupt Mask.
GPIORIS	0x414	RO	6	6'h0	Raw/Interrupt Status
GPIOMIS	0x418	RO	6	6'h0	Masked Interrupt Status
GPIOIC	0x41C	WO	6	6'h0	Interrupt Clear.

Table 267. GPIO identification registers summary

Name	Offset	Type	Width (bit)	Resetvalue	Description
GPIOPeriphID0	0xFE0	RO	8	8'h61	Peripheral identification register (bits 7:0).
GPIOPeriphID1	0xFE4	RO	8	8'h10	Peripheral identification register (bits 15:8).
GPIOPeriphID2	0xFE8	RO	8	8'h04	Peripheral identification register (bits 23:16).
GPIOPeriphID3	0xFEC	RO	8	8'h00	Peripheral identification register (bits 31:24).
GPIOPCellID0	0xFF0	RO	8	8'h0D	ID Register bits (7:0)
GPIOCellID1	0xFF4	RO	8	8'hF0	ID Register bits (15:8)
GPIOCellID2	0xFF8	RO	8	8'h05	ID Register bits (23:16)
GPIOCellID3	0xFFC	RO	8	8'hB1	ID Register bits (31:24)

18.5.2 Register description

18.5.3 GPIODIR register

The GPIODIR is the data direction RW register which allows to configure each pin as either an input or an output. The GPIODIR bit assignments are given in *Table 268*.

Table 268. GPIODIR register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIODIR	6'h0	Each bit is associated to a pin. If a bit is set, the relevant pin is configured to be an output. Clearing a bit configures the relevant pin to be input (default).

Note: GPIO 6 & GPIO 7 are dedicated for SPI chipselect & can be configured in O/P mode only.

18.5.4 GPIODATA register

The GPIODATA is the data RW register which allows to read from and write to GPIO pins configured as input or output, respectively, when GPIO is in software mode. The GPIODATA bit assignments are given in *Table 269*.

In software mode, the GPIODATA content is transferred to the pins which have been configured as output through the GPIODIR register.

Table 269. GPIODATA register bit assignments

Bit	Name	Reset value	Description
[07:00]	GPIODATA	8'h0	Input/output data.

18.5.5 GPIOIS register

The GPIOIS (Interrupt Sense) is a RW register which allows configuring each pin to detect either a level or an edge for interrupt triggering. The GPIOIS bit assignments are given in *Table 270*.

Table 270. GPIOIS register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIOIS	6'h0	Each bit is associated to a pin. If a bit is set, level on the relevant pin is detected. Clearing a bit, edge on the relevant pin is detected (default).

18.5.6 GPIOIBE register

The GPIOIBE (Interrupt Both Edges) is a RW register which allows to configure each pin to detect both rising and falling edges for interrupt triggering, in case edge detection for that pin is enabled (clearing relevant bit in GPIOIS register). The GPIOIBE bit assignments are given in *Table 271*.

Table 271. GPIOIBE register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIOIBE	6'h0	Each bit is associated to a pin. If a bit is set, both edges on the relevant pin trigger an interrupt, regardless of GPIOIEV setting (*Section 18.5.7*). Clearing a bit, interrupt generation event is controlled by the GPIOIEV register (default). Single edge is determined by the corresponding bit in that register.

18.5.7 GPIOIEV register

The GPIOIEV (Interrupt Event) is a RW register which allows to select for each pin the interrupt triggering event (rising/falling edge, high/low level), depending on GPIOIS register setting (*Section 18.5.5*). The GPIOIEV bit assignments are given in *Table 272*.

Table 272. GPIOIEV register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIOIEV	6'h0	Each bit is associated to a pin. If a bit is set, rising edge or high level on the relevant pin triggers the interrupt. Clearing a bit, falling edge or low level on that pin triggers the interrupt (default).

18.5.8 GPIOIE register

The GPIOIE (interrupt mask) is a RW register which allows to enable/disable interrupt triggering for each pin. The GPIOIE bit assignments are given in *Table 273*.

Table 273. GPIOIE register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIOIE	6'h0	Each bit is associated to a pin. If a bit is set, the relevant pin is allowed to trigger their interrupts (pin not masked). Clearing a bit, the relevant pin is masked and interrupt triggering is disabled for that pin (default).

18.5.9 GPIORIS register

The GPIORIS (Raw Interrupt Status) is a RO register which reflects the raw status (prior to masking through GPIOIE register) of interrupts trigger conditions on each pin. The GPIORIS bit assignments are given in *Table 274*.

Table 274. GPIORIS register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIORIS	6'h0	Each bit is associated to a pin. If a bit is set, it indicates that all requirements for interrupt triggering have been met on the relevant pin. If a bit is cleared, it means that requirements have not been met on the relevant pin and an interrupt has not been initiated (default).

18.5.10 GPIOMIS register

The GPIOMIS (Masked Interrupt Status) is a RO register which reflects the status of interrupts trigger conditions on each pin after masking (through GPIOIE register). The GPIOMIS bit assignments are given in *Table 275*.

The content of this register is available externally through the **`GPIOMIS[7:0]`** signals.

Table 275. GPIOMIS register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIOMIS	6'h	Each bit is associated to a pin. If a bit is set, it indicates that the relevant pin is triggering an interrupt. If a bit is cleared, it means that on that pin either no interrupt has been generated or the interrupt is masked by GPIOIE (default).

18.5.11 GPIOIC register

The GPIOIC (Interrupt Clear) is a WO register which allows to clear the interrupt edge detection. The GPIOIC bit assignments are given in *Table 276*.

Table 276. GPIOIC register bit assignments

Bit	Name	Reset value	Description
[05:00]	GPIOIC	6'h0	Each bit is associated to a pin. Setting a bit, the corresponding interrupt request is cleared. Clearing a bit has no effect (default).

19 BS_DMA controller

19.1 Overview

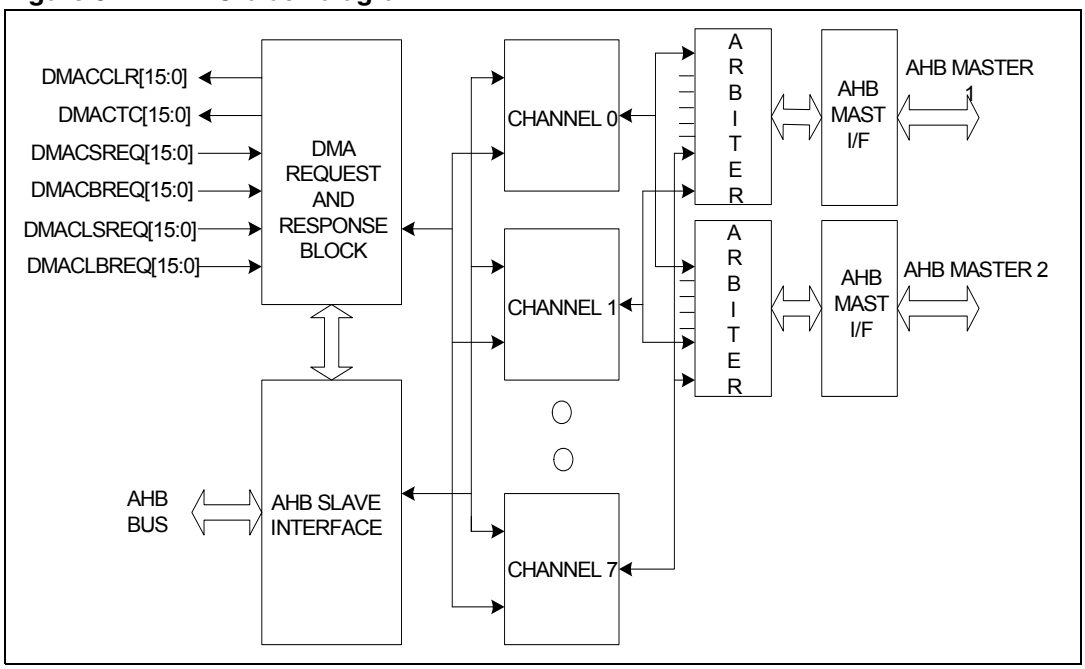
Within its basic subsystem, SPEAr300 provides an DMA controller (DMAC) able to service up to 8 independent DMA channels for sequential data transfers between single source and destination (i.e., memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral).

Main features of the DMAC are:

- Each DMA channel can support a unidirectional transfer, with internal 16-words FIFO per channel.
- 16 peripheral DMA request lines, where each peripheral connected to the DMAC can assert either a single DMA request or a Burst DMA request (with programmable size to increase data transfer effectiveness).
- Hardware priority (0 the highest to 7 the lowest) for each DMA channel to manage requests from more than 1 channel at the same time.
- Scatter or gather DMA support through the use of linked lists.
- An AHB slave acting as programming interface to access to DMA control registers:
- Two AHB masters for data transfer following a DMA request.
- 32 bit AHB master bus width, supporting 8, 16, and 32 bit wide transactions.
- Support both big-endian and little-endian (Little endian default on DMAC reset).
- Separate and combined DMA error and DMA count interrupt requests, with three interrupt request signals (DMACINTTC, DMACINTERR and DMACINTR).
- Interrupt masking and raw interrupt status (prior to masking).

19.2 Block diagram

Figure 32. DMAC block diagram



19.3 Signal interfaces

The DMAC directly interfaces with the signals summarized in *Table 277*. A functional diagram of these signal interfaces is given in *Figure 33*.

Figure 33. DMAC signal interface diagram

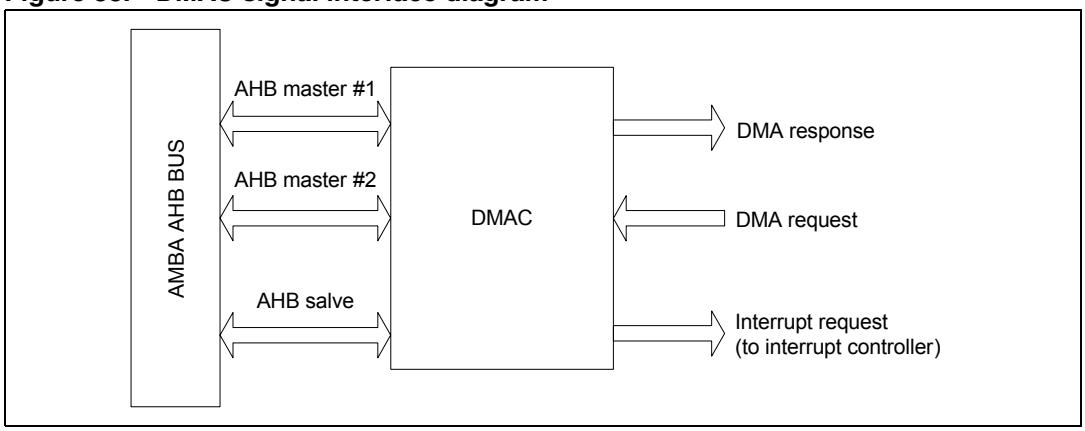


Table 277. DMAC signal interface

Group	Signal name	Direction	Size (bit)	Description
DMA request	DMACBREQ	Input	16	DMA burst transfer request.
	DMACLBREQ	Input	16	DMA last burst transfer request.
	DMACSREQ	Input	16	DMA single transfer request.
	DMACLSREQ	Input	16	DMA last single transfer request
DMA response	DMACCLR	Output	16	DMA request clear.
	DMACTC	Output	16	DMA terminal count (transaction complete).
Interrupt request	DMACINTERR	Output	1	DMA error interrupt request.
	DMACINTTC	Output	1	DMA terminal count interrupt request.
	DMACINTR	Output	1	DMA interrupt request. This signal combines the `DMACINTERR` and `DMACINTTC` requests.
AHB Master #1	-	Input/Output	-	See AMBA specification.
AHB Master #2	-	Input/Output	-	See AMBA specification.
AHB Slave	-	Input/Output	-	See AMBA specification.

19.4 Main functions description

19.4.1 AHB slave interface

The *AHB slave interface* block allows to connect the DMAC to the AMBA AHB bus.

In particular, the AHB slave interface properly decodes read and write command on AHB bus providing access to DMAC memory-mapped registers for configuration purposes.

It is worth noticing that the AHB slave and the two AHB masters use the same clock, HCLK, that is they are all synchronous.

19.4.2 AHB master interfaces

The DMAC contains two full independent AHB masters for data transfer. This feature allows, for example, the DMAC to transfer data directly from the memory connected to AHB port #1 to any AHB peripheral connected to AHB port #2. Besides, it enables transactions between the DMAC and any APB peripheral to occur independently of transactions on AHB bus 1.

Each AHB master is capable of dealing with all types of AHB transactions, including:

- Split, retry and error responses from AHB slaves. If a peripheral performs a split or retry, the DMAC stalls and waits until the transaction can complete.
- Locked transfers for source and destination of each stream.
- Setting of protection bits for transfers on each stream.

The two AHB masters are connected to buses of the same width (the default is a 32 bit bus). However, source and destination transfers can be with different widths, and can be the same

width or narrower than the physical bus width. In this case, the DMAC packs or unpacks data as appropriate.

Note: The DMAC uses `HSIZE1` or `HSIZE2` to indicate the width of a transfer, and if this fails to match the width expected by the peripheral, then the peripheral can assert an error on `HRESP1` or `HRESP2`, respectively.

19.4.3 DMA interface

The *DMA interface* provides the set of signals (listed in *Table 277*) to be used by a generic peripheral. Over this interface the connected peripheral is allowed to request a data transfer (through DMA request signals), and DMA is able to reply to peripheral both acknowledging the request and stating whether the data transfer has been completed (through DMA response signals).

As stated in *Table 277*, each DMA request/response signal is 16 bit wide, allowing then DMAC connectivity with up to 16 peripherals.

Note: Some peripherals do not use all the signals provided by the DMA interface. In this case, response signals that are not required can be left unconnected, and request signals that are not required can be tied to low.

Because of DMA interface, the DMAC enables four different data transfer types:

- Memory-to-memory
- Memory-to-peripheral
- Peripheral-to-memory
- Peripheral-to-peripheral,

where each transfer can have either the peripheral or the DMAC as the flow controller, resulting then in eight different scenarios (see FlowCntrl field in *Table 300: DMAC Configuration register bit assignments*).

19.5 Scatter/gather

As mentioned before, the DMAC provides for scatter/gather DMA through the use of a series of linked lists, allowing then source and destination of any DMA transfer to occupy non-contiguous areas in memory.

Each item of a linked list, referred to as Linked List Item (LLI), controls the transfer of one block of data (the packet) over a DMA channel, and then optionally loads another LLI to continue the DMA operation (in case of more than a packet transfer), or stops the DMA stream.

An LLI consists of four words:

- the source address of the data to be transferred over a DMA channel,
- the destination address of the data to be transferred over a DMA channel,
- the pointer to next LLI (set to 0 in case current LLI is the last in its linked list),
- a control word containing information about the corresponding DMA channel.

The first LLI of each linked list is programmed into the DMAC using the DMA channel registers, namely DMACCnSrcAddr, DMACC*n*DestAddr, DMACC*n*LLI and DMACC*n*Control. Then, these registers are updated as soon as a complete packet has been transferred over the DMA channel by following the linked list.

Note: The DMAC configuration is not part of the LLI description, but allows to configure the relevant DMA channel.

19.5.1 How to program the DMAC for scatter/gather DMA

1. Write to memory all the LLIs for the complete DMA transfer (source address, destination address, pointer to next LLI and control word for each LLI).
2. Choose a free DMA channel with the required priority (0 the highest to 7 the lowest).
3. Write the first LLI, previously written to memory (step 1), to the relevant DMA channel in the DMAC, setting the corresponding registers (DMACC*n*SrcAddr, DMACCnDestAddr, DMACC*n*LLI and DMACC*n*Control).
4. Write the DMA channel configuration information to the DMAC Configuration and set its channel enable bit (E, bit [0]).
5. An interrupt can be generated at the end of each LLI setting the terminal count bit (I, bit [31]) in the DMACC*n*Control register (*Section 19.7.20*). Then, the interrupt request must be serviced and the relevant bit of the IntTCClear field in the DMACIntTCClear register (*Section 19.7.5*) must be set to clear the interrupt.

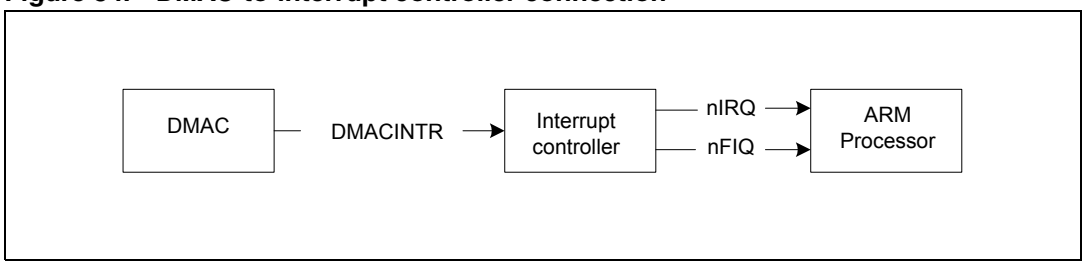
19.6 Interrupt requests

The DMAC allows to generate an interrupt to the ARM processor:

- In case of a DMA error (assertion of an error response on the AHB during data transfer),
- At the end of DMA transfer (terminal count reached 0).

The corresponding interrupt request signals are listed in *Table 277*. The combined DMACINTR signal (generated as an OR function of the individual request signals, DMACINTERR and DMACINTTC) can be useful in low performance system with a few interrupt controller request inputs. As depicted in *Figure 34*, in this case only the DMACINTR request signal coming from DMAC is directly connected to the interrupt controller, but both the DMACIntErrStatus (*Section 19.7.6*) and the DMACIntTCStatus (*Section 19.7.4*) registers, in conjunction with the DMACIntStatus (*Section 19.7.3*) register, must be read to find the actual source of the interrupt.

Figure 34. DMAC-to-interrupt controller connection



In any case the error and terminal count interrupts can be masked at the DMAC-level by programming the relevant bits (IE and ITC, respectively) on the relevant DMACC*n*Configuration (*Section 19.7.15*) channel register. In order to get interrupt status prior to masking, the DMACRawIntErrStatus (*Section 19.7.9*) and the DMACRawIntTCStatus (*Section 19.7.8*) registers are provided by the DMAC.

19.6.1 How to operate single combined DMACINTR interrupt request signal

1. Wait until the combined interrupt request from the DMAC (DMACINTR) goes active.
2. Read the interrupt controller status register and determine whether the source of the request was the DMAC.
3. Read the DMACIntStatus register (*Section 19.7.4*) to determine the DMA channel that generated the interrupt. If more than one request is active (that is, more than one bit are set in the IntStatus field), it is recommended to check the highest priority channels first (0, 1, 2 and so on).
4. Read the DMACIntTCStatus register (*Section 19.7.4*) to determine whether the interrupt was generated because of the end of the transfer or because an error occurred. If the bit corresponding to the DMA channel (from step #3) in the field IntTCStatus is set, the data transfer has been completed? step #6.
5. Read the DMACIntErrStatus register (*Section 19.7.7*) to determine whether the interrupt was generated because of the end of the transfer or because an error occurred. If the bit corresponding to the DMA channel (from step #3) in the field IntErrorStatus is set, an error occurred › step #6
6. Set the relevant bit in the DMACIntTCClear register (*Section 19.7.5*) or in the DMACIntErrClr register (*Section 19.7.7*), respectively, to clear the interrupt request.

19.7 Programming model

19.7.1 Register map

The DMAC can be fully configured by programming its 32 bit wide registers which can be accessed through the AHB slave interface at the base address 0xFC40_0000.

DMAC registers can be logically arranged in four main groups:

- Global registers, listed in *Table 278*, for DMAC-level configuration,
- Channel registers for programming a single DMA channel. Each DMA channel is associated to these five registers, listed in *Table 279* where n ranges from 0 to 7 being 8 the number of DMA channels supported by the DMAC,
- Peripheral identification registers, listed in *Table 280*,
- Cell identification registers, listed in *Table 281*.

Table 278. DMAC global registers summary

Name	Offset	Type	Reset Value	Description
DMACIntStatus	0x000	RO	32'h0	Interrupt status.
DMACIntTCStatus	0x004	RO	32'h0	Interrupt terminal count status.
DMACIntTCClear	0x008	WO	32'h0	Interrupt terminal count clear.
DMACIntErrorStatus	0x00C	RO	32'h0	Interrupt error status.
DMACIntErrClr	0x010	WO	32'h0	Interrupt error clear.
DMACRawIntTCStatus	0x014	RO	32'h0	Raw interrupt terminal count status.
DMACRawIntErrorStatus	0x018	RO	32'h0	Raw interrupt error status.

Table 278. DMAC global registers summary (continued)

Name	Offset	Type	Reset Value	Description
DMACEnbldChns	0x01C	RO	32'h0	Enabled channel.
DMACSoftBReq	0x020	RW	32'h0	Software burst request.
DMACSoftSReq	0x024	RW	32'h0	Software single request.
DMACSoftLBReq	0x028	RW	32'h0	Software last burst request.
DMACSoftLSReq	0x02C	RW	32'h0	Software last single request.
DMACConfiguration	0x030	RW	32'h0	DMAC configuration
DMACSync	0x034	RW	32'h0	Synchronization.

Table 279. DMAC channel registers summary

Name	Offset	Type	Reset Value	Description
DMACC*n*SrcAddr	0x100 + (*n* · 0x020)	RW	32'h0	Channel source address.
DMACC*n*DestAddr	0x104 + (*n* · 0x020)	RW	32'h0	Channel destination address.
DMACC*n*LLI	0x108 + (*n* · 0x020)	RW	32'h0	Channel linked list item.
DMACC*n*Control	0x10C + (*n* · 0x020)	RW	32'h0	Channel control.
DMACC*n*Configuration	0x110 + (*n* · 0x020)	RW	32'h0	Channel configuration.

Table 280. DMAC peripheral registers summary

Name	Offset	Type	Description
DMACPeriphID0	0xFE0	RO	See *Section 19.7.22*.
DMACPeriphID1	0xFE4	RO	See *Section 19.7.22*.
DMACPeriphID2	0xFE8	RO	See *Section 19.7.22*.
DMACPeriphID3	0xFEC	RO	See *Section 19.7.22*.

Table 281. DMAC cell identification registers summary

Name	Offset	Type	Description
DMACPCellID0	0xFF0	RO	See *Section 19.7.23*.
DMACPCellID1	0xFF4	RO	See *Section 19.7.23*.
DMACPCellID2	0xFF8	RO	See *Section 19.7.23*.
DMACPCellID3	0xFFC	RO	See *Section 19.7.23*.

19.7.2 Register description

19.7.3 DMACIntStatus register

The DMACIntStatus (interrupt status) is a RO register which shows the status of the interrupts after masking. The DMACIntStatus bit assignments are given in *Table 282*.

Table 282. DMACIntStatus register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined.
[07:00]	IntStatus	8'h00	Status of DMA interrupts after masking. Each bit is associated to a DMA channel. If a bit is set, it means that an interrupt request is active for the relevant DMA channel.

19.7.4 DMACIntTCStatus register

The DMACIntTCStatus (interrupt terminal count status) is a RO register which shows the status of the terminal count after masking. The DMACIntTCStatus bit assignments are given in *Table 283*.

Note: This register must be used in conjunction with the DMACIntStatus register if the combined interrupt request, DMACINTR, is used. If the DMACINTTC interrupt request is used, reading this register only is enough to determine source of the interrupt request.

Table 283. DMACIntTCStatus register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined.
[07:00]	IntTCStatus	8'h00	Interrupt terminal count request status. Each bit is associated to a DMA channel. If a bit is set, it means that an interrupt terminal count request is active for the relevant DMA channel.

19.7.5 DMACIntTCClear register

The DMACIntTCClear (interrupt terminal count clear) is a WO register which allow to clear a terminal count interrupt request. The DMACIntTCClear bit assignments are given in *Table 284*.

Table 284. DMACIntTCClear register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Write as zero.
[07:00]	IntTCClear	8'h00	Terminal count request clear. Each bit is associated to a DMA channel. When writing to this register, each bit that is set causes the corresponding bit in the DMACIntTCStatus register to be cleared. In contrast, bits that are not set have no effect on the corresponding bit in the DMACIntTCStatus register.

19.7.6 DMACIntErrorStatus register

The DMACIntErrorStatus (interrupt error status) is a RO register which shows the status of the error request after masking. The DMACIntErrorStatus bit assignments are given in *Table 285*.

Note: This register must be used in conjunction with the DMACIntStatus register if the combined interrupt request, DMACINTR, is used. If the DMACINTERR interrupt request is used, reading this register only is enough to determine source of the interrupt request.

Table 285. DMA CIntErrorStatus register bit assignments

Bit	Name	Reset Value	Description
[31:08]	Reserved	-	Read: undefined.
[07:00]	IntErrorStatus	8'h00	Interrupt error status. Each bit is associated to a DMA channel. If a bit is set, it means that an interrupt error request is active for the relevant DMA channel.

19.7.7 DMACIntErrClr register

The DMACIntErrClr (interrupt error clear) is a WO register which allow to clear an error interrupt request. The DMACIntErrClr bit assignments are given in *Table 286*.

Table 286. DMACIntErrClr register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Write as zero.
[07:00]	IntErrClr	8'h00	Interrupt error request clear. Each bit is associated to a DMA channel. When writing to this register, each bit that is set causes the corresponding bit in the DMACIntErrorStatus register to be cleared. In contrast, bits that are not set have no effect on the corresponding bit in the DMACIntErrorStatus register.

19.7.8 DMACRawIntTCStatus register

The DMACRawIntTCStatus (raw interrupt terminal count status) is a RO register which indicates the DMA channels that are requesting a transfer complete, terminal count interrupt, prior to masking. The DMACRawIntTCStatus bit assignments are given in *Table 287*.

Table 287. DMACRawIntTCStatus register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined.
[07:00]	RawIntTCStatus	8'h00	Status of the terminal count interrupt prior to masking. Each bit is associated to a DMA channel. If a bit is set, it means that a terminal count interrupt request is active prior to masking for the relevant DMA channel.

19.7.9 DMACRawIntErrorStatus register

The DMACRawIntErrorStatus (raw interrupt error status) is a RO register which indicates the DMA channels that are requesting an errror interrupt prior to masking. The DMACRawIntErrorStatus bit assignments are given in *Table 288*.

Table 288. DMACRawIntErrorStatus register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined.
[07:00]	RawIntErrorStatus	8'h00	Status of the error interrupt prior to masking. Each bit is associated to a DMA channel. If a bit is set, it means that an error interrupt request is active prior to masking for the relevant DMA channel.

19.7.10 DMACEnbldChns register

The DMACEnbldChns (enabled channel) is a RO register which indicates the DMA channels that are enabled, as indicated by the Enable bit (E) in the DMACCnConfiguration register (*Section 19.7.15*). The DMACEnbldChns bit assignments are given in *Table 289*.

Table 289. DMACEnbldChns register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined.
[07:00]	EnabledChannels	8'h00	Channel enable status. Each bit is associated to a DMA channel. If a bit is set, it means that corresponding DMA channel is enabled. A bit is cleared on completion of the DMA transfer.

19.7.11 DMACSoftBReq register

The DMACSoftBReq (software burst request) is a RW register which enables DMA burst requests to be generated by software. The DMACSoftBReq bit assignments are given in *Table 290*.

Table 290. DMACSoftBReq register bit assignments

Bit	Name	Reset value	Description
[31:16]	Reserved	-	Read: undefined. Write as zero.
[15:00]	SoftBReq	16'h0000	Software last burst request. Each bit is associated to one out of 16 peripheral DMA request lines. Setting a bit, a DMA last burst request for the corresponding peripheral is generated, and the bit is cleared when the transaction has completed. Reading this field of the register indicates the sources that are requesting DMA last burst transfers.

Note: A DMA burst request can be generated form either a peripheral or the software request register. However, it is recommended not to use software and hardware peripheral requests at the same time.

19.7.12 DMACSoftSReq register

The DMACSoftSReq (software single request) is a RW register which enables DMA single requests to be generated by software. The DMACSoftSReq bit assignments are given in *Table 291*.

Table 291. DMACSoftSReq register bit assignments

Bit	Name	Reset value	Description
[31:16]	Reserved	-	Read: undefined. Write as zero.
[15:00]	SoftSReq	16'h0000	Software single request. Each bit is associated to one out of 16 peripheral DMA request lines. Setting a bit, a DMA single request for the corresponding peripheral is generated, and the bit is cleared when the transaction has completed. Reading this field of the register indicates the sources that are requesting DMA single transfers.

Note: A DMA single request can be generated form either a peripheral or the software request register. However, it is recommended not to use software and hardware peripheral requests at the same time.

19.7.13 DMACSoftLBReq register

The DMACSoftLBReq (software last burst request) is a RW register which enables DMA last burst requests to be generated by software. The DMACSoftLBReq bit assignments are given in *Table 292*.

Table 292. DMACSoftLBReq register bit assignments

Bit	Name	Reset value	Description
[31:16]	Reserved	-	Read: undefined. Write as zero.
[15:00]	SoftLBReq	16'h0000	Software last burst request. Each bit is associated to one out of 16 peripheral DMA request lines. Setting a bit, a DMA last burst request for the corresponding peripheral is generated, and the bit is cleared when the transaction has completed. Reading this field of the register indicates the sources that are requesting DMA last burst transfers.

Note: A DMA last burst request can be generated form either a peripheral or the software request register.

19.7.14 DMACSoftLSReq register

The DMACSoftLSReq (software last single request) is a RW register which enables DMA last single requests to be generated by software. The DMACSoftLSReq bit assignments are given in *Table 293*.

Table 293. DMACSoftLSReq register bit assignments

Bit	Name	Reset value	Description
[31:16]	Reserved	-	Read: undefined. Write as zero.
[15:00]	SoftLSReq	16'h0000	Software last single request. Each bit is associated to one out of 16 peripheral DMA request lines. Setting a bit, a DMA last single request for the corresponding peripheral is generated, and the bit is cleared when the transaction has completed. Reading this field of the register indicates the sources that are requesting DMA last single transfers.

Note: A DMA last single request can be generated form either a peripheral or the software request register.

19.7.15 DMAC configuration register

The DMACConfiguration is a RW register which allows to configure the operation of the DMAC. The DMACConfiguration bit assignments are given in *Table 294*.

Table 294. DMACConfiguration register bit assignments

Bit	Name	Reset value	Description
[31:03]	Reserved	-	Read: undefined. Write as zero.
[02]	M2	1'h0	AHB master 2 endianness configuration. This bit enables to alter the endianness of the AHB master interface 2, according to encoding: 1'b0 = Little-endian mode. 1'b1 = Big-endian mode.

Table 294. DMACConfiguration register bit assignments (continued)

Bit	Name	Reset value	Description
[01]	M1	1'h0	AHB master 1 endianness configuration. This bit enables to alter the endianness of the AHB master interface 1, according to the same encoding as M2 (see above).
[00]	E	1'h0	DMAC enable. Setting this bit, the DMAC is enabled. Clearing this bit, the DMAC is disabled reducing power consumption.

19.7.16 DMACSync register

The DMACSync (synchronization) is a RW register which allows to enable/disable synchronization logic for the DMA request signals, namely DMACBREQ[15:0], DMACSREQ[15:0], DMACLBREQ[15:0] and DMACLSREQ[15:0]. The DMACSync bit assignments are given in *Table 295*.

Note: Synchronization logic must be used when the peripheral generating the DMA request runs on a different clock to the DMAC. For peripherals running on the same clock as DMA, disabling the synchronization logic improves the DMA request response time.

Table 295. DMACSync register bit assignments

Bit	Name	Reset value	Description
[31:16]	Reserved	-	Read: undefined. Write as zero.
[15:00]	DMACSync	16'h0000	DMA synchronization logic enable. Each bit is associated to one out of 16 peripheral DMA request lines. A cleared bit (as for default) indicates that the synchronization logic for the request signals is enabled. In contrast, setting the bit the synchronization logic is disabled.

19.7.17 DMACCnSrcAddr register

The DMACCnSrcAddr (channel n source address) is a RW register which contains the current source address (byte-aligned) of the data to be transferred over the n-th DMA channel. The DMACCnSrcAddr bit assignments are given in *Table 296*.

Note: Source and destination addresses must be aligned to the source and destination widths.

Software programs the DMACCnSrcAddr register directly before the appropriate DMA channel is enabled. Once the corresponding DMA channel is enabled, this register is updated:

- As the source address is incremented,
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the DMA channel is active does not provide useful information. This is because by the time the software has processed the value read, the channel might have progressed. It is intended to be read-only when the channel has stopped, and in such case, it shows the source address of the last item read.

Table 296. DMACCnSrcAddr register bit assignments

Bit	Name	Reset value	Description
[31:00]	SrcAddr	32'h0	DMA source address.

19.7.18 DMACCnDestAddr register

The DMACCnDestAddr (channel n destination address) is a RW register which contains the current destination address (byte-aligned) of the data to be transferred over the n-th DMA channel. The DMACCnDestAddr bit assignments are given in *Table 297*.

Note: Source and destination addresses must be aligned to the source and destination widths.

Software programs the DMACCnDestAddr register directly before the appropriate DMA channel is enabled. Once the corresponding DMA channel is enabled, this register is updated:

- as the destination address is incremented,
- by following the linked list when a complete packet of data has been transferred.

Reading the register when the DMA channel is active does not provide useful information. This is because by the time the software has processed the value read, the channel might have progressed. It is intended to be read-only when the channel has stopped, and in such case, it shows the source address of the last item read.

Table 297. DMACCnDestAddr register bit assignments

Bit	Name	Reset value	Description
[31:00]	DestAddr	32'h0	DMA destination address.

19.7.19 DMACCnLLI register

The DMACCnLLI (channel n linked list item) is a RW register which contains the address (word-aligned) of the next Linked List Item (LLI). If next LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled after all DMA transfers associated with it are completed. The DMACCnLLI bit assignments are given in *Table 298*.

Note: Programming this register when the corresponding DMA channel is enabled has unpredictable results.

Table 298. DMACCnLLI register bit assignments

Bit	Name	Reset value	Description
[31:02]	LLI	30'h0	Next LLI address. This field contains the bits [31:2] of the address for the next LLI. Address LSB bits [1:0] are 1'b0 both.
[01]	Reserved	-	Read: undefined. Write as zero.
[00]	LM	1'h0	AHB master select. This bit allows to select the AHB master for loading the next LLI, according to encoding: 1'b0 = AHB master 1. 1'b1 = AHB master 2.

19.7.20 DMACCn control register

The DMACCnControl is a RW register which contains control information about the DMA channel n, such as transfer size, burst size and transfer width. The DMACCnControl bit assignments are given in *Table 299*.

Software programs the DMACCnControl register directly before the appropriate DMA channel is enabled. Once the corresponding DMA channel is enabled, this register is updated by following the linked list when a complete packet of data has been transferred.

Reading the register when the DMA channel is active does not provide useful information. This is because by the time the software has processed the value read, the channel might have progressed. It is intended to be read-only when the channel has stopped.

Table 299. DMACCnControl register bit assignments

Bit	Name	Reset value	Description
[31]	I	1'h0	Terminal count interrupt enable. This bit controls whether the current LLI is expected to trigger the terminal count interrupt.
[30:28]	Port	3'h0	Protection. This 3 bits field reports AHB access information which are primarily intended to be used by source and destination peripherals for implementing some level of protection. This field directly controls the AHB `HPROT[3:1]` signals, and bit assignment is given: [28], HPROT[1] = 1'b0 user mode [28], HPROT[1] = 1'b1 privileged mode [29], HPROT[2] = 1'b0 non-bufferable [29], HPROT[2] = 1'b1 bufferable [30], HPROT[3] = 1'b0 non-cacheable [30], HPROT[3] = 1'b1 cacheable
[27]	DI	1'h0	Destination increment. If the bit is set, the destination (resp. source) address is incremented after each transfer.
[26]	SI	1'h0	Source increment. If the bit is set, the destination (resp. source) address is incremented after each transfer.
[25]	D	1'h0	Destination AHB master select. This bit allows to select the AHB master for the destination (resp. source) transfer, according to encoding: 1'b0 = AHB master 1. 1'b1 = AHB master 2.
[24]	S	1'h0	Source AHB master select. This bit allows to select the AHB master for the destination (resp. source) transfer, according to encoding: 1'b0 = AHB master 1. 1'b1 = AHB master 2.

Table 299. DMACCnControl register bit assignments (continued)

Bit	Name	Reset value	Description
[23:21]	Dwidth	3'h0	Destination transfer width. This 3 bits field states the width of destination (resp. source) transfer, according to encoding: 3'b000 = Byte (8 bit) 3'b001 = Halfword (16 bit) 3'b010 = Word (32 bit) 3'b011 to 3'b111 = Reserved The hardware automatically packs and unpacks the data when required. Note: Transfers wider than the AHB master bus width are illegal. Besides, the source and the destinations widths can be different from each other.
[20:18]	Swidth	3'h0	Source transfer width. This 3 bits field states the width of destination (resp. source) transfer, according to encoding: 3'b000 = Byte (8 bit) 3'b001 = Halfword (16 bit) 3'b010 = Word (32 bit) 3'b011 to 3'b111 = Reserved The hardware automatically packs and unpacks the data when required. Note: Transfers wider than the AHB master bus width are illegal. Besides, the source and the destinations widths can be different from each other.
[17:15]	DBSize	3'h0	Destination burst size. This 3 bits field indicates the number of transfers that make up a destination (resp. source) burst transfer request, according to the encoding: 3'b000 = 1 3'b001 = 4 3'b010 = 8 3'b011 = 16 3'b100 = 32 3'b101 = 64 3'b110 = 128 3'b111 = 256 This value must be set to the burst size of the destination (resp. source) peripheral, being the burst size the amount of data that is transferred when the n-th DMACBREQ signal goes active in the destination (resp. source) peripheral. In case destination (resp. source) is the memory, this value must be set to the memory boundary size. Note: Burst equal or greater than 32 are available only using data-width 32. The data-width 8 and 16 support only bursts of 1,4,8 & 16.

Table 299. DMACCnControl register bit assignments (continued)

Bit	Name	Reset value	Description
[14:12]	SBSize	3'h0	Source burst size. This 3 bits field indicates the number of transfers that make up a destination (resp. source) burst transfer request, according to the encoding: 3'b000 = 1 3'b001 = 4 3'b010 = 8 3'b011 = 16 3'b100 = 32 3'b101 = 64 3'b110 = 128 3'b111 = 256 This value must be set to the burst size of the destination (resp. source) peripheral, being the burst size the amount of data that is transferred when the n-th DMACBREQ signal goes active in the destination (resp. source) peripheral. In case destination (resp. source) is the memory, this value must be set to the memory boundary size.
[11:00]	Transfer Size	12'h0	Transfer size. A write to this field sets the size of the transfer in case the DMAC is the flow controller. This value counts down from the original value to zero, and a read from this field provides then the number of transfers still to be completed on the destination bus. Note: This field should be set to zero if the DMAC is not the flow controller, avoiding then the DMAC might attempt to use a non-zero value instead of ignoring the field.

19.7.21 DMAC Configuration register

The DMAC Configuration is a RW register which allow to configure the relevant DMA channel. The DMAC Configuration bit assignments are given in *Table 300*.

Table 300. DMAC Configuration register bit assignments

Bit	Name	Reset value	Description
[31:19]	Reserved	-	Read: undefined. Write as zero.
[18]	H	1'h0	Halt. Setting this bit, extra source DMA requests are ignored (otherwise enabled), and the content of channel FIFO is drained. This bit can be jointly used with the active bit (A field in this register) and the channel enable bit (E field in this register) to cleanly disable a DMA channel.
[17]	A	1'h0	Active (read-only). If this read-only field is set, it means that there is still data in the channel FIFO. This bit can be jointly used with the halt bit (H field in this register) and the channel enable bit (E field in this register) to cleanly disable a DMA channel.

Table 300. DMAC Configuration register bit assignments (continued)

Bit	Name	Reset value	Description
[16]	L	1'h0	Lock. Setting this bit, locked transfers are enabled: when a burst occurs, the HLOCK signal is asserted by the DMAC, so that the AHB arbiter doesn't degrant the DMAC during the burst until the lock is deasserted, even if another master with greater priority requests the bus.
[15]	ITC	1'h0	Terminal count interrupt mask. Clearing this bit, it masks out the terminal count interrupt for this DMA channel.
[14]	IE	1'h0	Error interrupt mask. Clearing this bit, it masks out the error interrupt for this DMA channel.
[13:11]	FlowCntrl	3'h0	Flow control and transfer type. This 3 bits field indicates both the flow controller (DMAC, destination peripheral or source peripheral) and the transfer type (memory-to-memory, memory-to-peripheral, …), according to encoding: 3'b000 = Memory-to-memory, DMAC 3'b001 = Memory-to-peripheral, DMAC 3'b010 = Peripheral-to-memory, DMAC 3'b011 = Source periph.-to-destination periph., DMAC 3'b100 = Source periph.-to-destination periph., Destination peripheral 3'b101 = Memory-to-peripheral, Peripheral 3'b110 = Peripheral-to-memory, Peripheral 3'b111 = Source periph.-to-destination periph. Source peripheral, DestPeripheral,
[10]	Reserved	-	Read: undefined. Write as zero.
[09:06]	DestPeripheral	4'h0	Destination peripheral. This 4 bits field allows to select the DMA destination (resp. source) request peripheral. The value is ignored in case the destination (resp. source) of the transfer is the memory. Note: The DestPeripheral and SrcPeripheral fields are the binary value of the request line (4'h0 to 4'hF, that is 0 to 15) and not a mask value.
[05]	Reserved	-	Read: undefined. Write as zero.

Table 300. DMAC Configuration register bit assignments (continued)

Bit	Name	Reset value	Description
[04:01]	SrcPeripheral	4'h0	Source peripheral. This 4 bits field allows to select the DMA destination (resp. source) request peripheral. The value is ignored in case the destination (resp. source) of the transfer is the memory. Note: The DestPeripheral and SrcPeripheral fields are the binary value of the request line (4'h0 to 4'hF, that is 0 to 15) and not a mask value.
[00]	E	1'h0	Channel enable. Setting this bit, the relevant DMA channel is enabled. When this bit is cleared, the current AHB transfer – if any – is firstly completed (losing any data in the channel FIFO), then the channel is disabled. Note: Restarting the DMA channel by setting back the E bit results in unpredictable effects and the channel must be fully re-initialized. If a DMA channel has to be disabled without losing data in its channel's FIFO, at first the Halt bit must be set, so that subsequent DMA requests are ignored. Then, the Active bit must be polled until it reaches 1'b0, indicating that there is no data left in the channel's FIFO. Finally, the Channel Enable bit can be cleared. The DMA channel is also disabled (and the E bit cleared) when either the last LLI is reached or if a channel error is encountered. Reading this bit indicates whether the DMA channel is enabled or disabled.

19.7.22 DMACPeriphID register

The DMACPeriphID are four 8 bit RO registers, which can be treated conceptually as a single 32 bit register. These read-only registers provide the following peripheral options:

- PartNumber[11:00] - This identifies the peripheral. The three digit product code 0x080 is used.
- Designer ID[19:12] - This is the identification of the designer. ARM Limited is 0x41 (ASCII A).
- Revision[23:20] - This is the revision number of the peripheral. The revision number starts from 0.
- Configuration[31:24] - This is the configuration option of the peripheral.

19.7.23 DMACPCellID register

The DMACPCellID are four 8 bit RO registers, which can be treated conceptually as a single 32 bit register. The register is a standard cross-peripheral identification system. The DMACPCellID register is set to 0xB105_F00D.

20 BS_Real time clock

20.1 Overview

Within its basic subsystem, SPEAr300 provides a real time clock (RTC) acting as an APB slave.

Main features of the RTC block are:

- Provides time-of-day clock in 24 hours mode.
- Provides calendar.
- Makes available alarm capability.
- Provides two general purpose registers.
- Supports a self-isolation mode, which allows RTC to work even if power is not supplied to the rest of the device.

20.2 Programming model

20.2.1 Register map

The RTC can be fully configured by programming its 32 bit wide registers (listed in *Table 301*) which can be accessed at the base address 0xFC90_0000.

Table 301. RTC functional registers summary

Name	Offset	Type	Reset value	Description
TIME	0x000	RW	Undefined	Time register
DATE	0x004	RW	Undefined	Date register
ALARM TIME	0x008	RW	Undefined	Alarm time register
ALARM DATE	0x00C	RW	Undefined	Alarm date register
CONTROL	0x010	RW	Undefined	Control register
STATUS	0x014	RW	Undefined	Status register
REG1MC	0x018	RW	Undefined	General purpose register
REG2MC	0x01C	RW	Undefined	General purpose register

20.2.2 Register description

20.2.3 CONTROL register

The CONTROL is a RW register which allows the software to control the RTC. The CONTROL register bit assignments are given *Table 302*.

Table 302. CONTROL register bit assignments

Bit	Name	Reset value	Description
[31]	IE		Interrupt event enable. Setting this bit, interrupt event is enabled, and interrupts generated by alarm logic are sent out (see ALARM TIME and ALARM DATE registers).
[30:10]	Reserved	-	Read: undefined. Write: should be zero.
[09]	TB		Time bypass (for testing purpose only).
[08]	PB		Prescaler bypass (for testing purpose only).
[07:06]	Reserved	-	Read: undefined. Write: should be zero.
[05:00]	MASK		Force time-calendar comparisons. Each bit of this 6 bit field allows to mask one time-calendar element (seconds, minutes, hours, days, months, years), according to encoding. The aim is to generate an interrupt for any masked element, apart from actual matching of programmed alarms. [00] = Seconds. [01] = Minutes. [02] = Hours. [03] = Days. [04] = Months. [05] = Years.

20.2.4 STATUS register

The STATUS is a RW register (with some RO field) which indicates the status of the RTC and allows to clear any pending interrupt. The STATUS register bit assignments are given in *Table 303*.

Table 303. STATUS register bit assignments

Bit	Name	Reset value	Type	Description
[31]	I		RW	Interrupt status. Reading from this 1 bit field, the interrupt status returns. Writing 1‘b1 to this bit clears any pending interrupts, whereas there is no effect writing 1‘b0.
[30:06]	Reserved	-	-	Read: undefined. Write: should be zero.
[05]	LD		RO	Write to DATE register lost. If a second write to DATE register is requested before the first is completed, this second request is aborted and the LD bit is set. This bit is cleared when a write to DATE register is performed successfully.

Table 303. STATUS register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[04]	LT		RO	Write to TIME register lost. If a second write to TIME register is requested before the first is completed, this second request is aborted and the LT bit is set. This bit is cleared when a write to TIME register is performed successfully.
[03]	PD		RO	Pending write to DATE register. If set, this bit indicates that a write to DATE register request is asserted from 48 MHz part to 32 kHz part. It is independent from PT. A new write can be successfully requested only when this bit is cleared.
[02]	PT		RO	Pending write to TIME register. If set, this bit indicates that a write to TIME register request is asserted from 48 MHz part to 32 kHz part. A new write can be successfully requested only when this bit is cleared.
[01]	Reserved	-	-	Read: undefined. Write: should be zero.
[00]	RC		RO	Isolation of timer If cleared (1'b0), the RTC is self-isolated from the rest of the chip. Reading and writing to TIME and DATE registers can be safely done when this bit is set only.

20.2.5 TIME register

The TIME is a RW register which defines the time (hour, minutes, seconds) when the RTC can start to count the time. The TIME register bit assignments are given in *Table 304*.

Note: All values in this TIME register are in binary-coded decimal (BCD) format.

Table 304. TIME register bit assignments

Bit	Name	Reset value	Description
[31:22]	Reserved	-	Read: undefined. Write: should be zero.
[21:20]	HT		Current hours tens.
[19:16]	HU		Current hours units.
[15]	Reserved	-	Read: undefined. Write: should be zero.
[14:12]	MT		Current minutes tens.
[11:08]	MU		Current minutes units.
[07]	Reserved	-	Read: undefined. Write: should be zero.
[06:04]	ST		Current seconds tens.
[03:00]	SU		Current seconds units.

20.2.6 DATE register

The DATE is a RW register which defines the date (year, month, day) when the RTC can start to count the time. The DATE register bit assignments are given in *Table 305*.

Note: All values in this DATE register are in binary-coded decimal (BCD) format.

Table 305. DATE register bit assignments

Bit	Name	Reset value	Description
[31:28]	YM		Current year millenniums.
[27:24]	YH		Current year hundreds.
[23:20]	YT		Current year tens.
[19:16]	YU		Current year units.
[15]	Reserved	-	Read: undefined. Write: should be zero.
[14:12]	MT		Current month tens.
[11:08]	MU		Current month units.
[07:06]	Reserved	-	Read: undefined. Write: should be zero.
[05:04]	DT		Current day tens.
[03:00]	DU		Current day units.

20.2.7 ALARM TIME registers

The ALARM TIME is a RW register which defines a successive time, so that when the value of TIME register is equal to the value set in this ALARM TIME register, an interrupt is generated (if enabled, that is if IE bit in CONTROL register is set). The ALARM TIME register bit assignments are given in *Table 306*.

Note: All values in this ALARM TIME register are in binary-coded decimal (BCD) format.

Table 306. ALARM TIME register bit assignments

Bit	Name	Reset value	Description
[31:22]	Reserved	-	Read: undefined. Write: should be zero.
[21:20]	HT		Target hour tens.
[19:16]	HU		Target hour units.
[15]	Reserved	-	Read: undefined. Write: should be zero.
[14:12]	MT		Target minute tens.
[11:08]	MU		Target minute units.
[07]	Reserved	-	Read: undefined. Write: should be zero.
[06:04]	ST		Target second tens.
[03:00]	SU		Target second units.

20.2.8 ALARM DATE registers

The ALARM DATE is a RW register which defines a successive date, so that when the value of DATE register is equal to the value set in this ALARM DATE register, an interrupt is generated (if enabled, that is if IE bit in CONTROL register is set). The ALARM DATE register bit assignments are given in *Table 307*.

Note: All values in this ALARM DATE register are in binary-coded decimal (BCD) format.

Table 307. ALARM DATE register bit assignments

Bit	Name	Reset value	Description
[31:28]	YM		Target year millenniums.
[27:24]	YH		Target year hundreds.
[23:20]	YT		Target year tens.
[19:16]	YU		Target year units.
[15]	Reserved	-	Read: undefined. Write: should be zero.
[14:12]	MT		Target month tens.
[11:08]	MU		Target month units.
[07:06]	Reserved	-	Read: undefined. Write: should be zero.
[05:04]	DT		Target day tens.
[03:00]	DU		Target day units.

20.2.9 REGxMC register

These general purpose registers, battery powered, can be used to store information when the system goes in a deep power saving state like suspend to ram. During this state only the DDR memory is powered and all the other parts of the system (SOC included) are completely off. The REGxMC registers bit assignments are given in *Table 308* and *Table 309*.

Table 308. REG1MC registers bit assignments

Bit	Name	Reset value	Description
[31:00]	REG1MC		General purpose bits

Table 309. REG2MC register bit assignments

Bit	Name	Reset value	Description
[31:00]	REG2MC		General purpose bits

21 AS_Cryptographic co-processor (C3)

21.1 Overview

Within its Application Connection Subsystem, SPEAr300 provides one Channel Control Co-processor (C3). C3 is a high-performance instruction driven DMA based co-processor. It executes instruction flows generated by the host processor. After it has been set-up by the host it runs in a completely autonomous way (DMA data in, data processing, DMA data out), until the completion of all the requested operations.

C3 has been used to accelerate the processing of cryptographic, security and network security applications. It can be used for other types of data intensive applications as well.

Main features provided by the C3 are listed below:

- C3 Hardware ID base address 0xFFFF_1000.
- High performance DMA based co-processor enabling the acceleration of data-driven computationally expensive functions, such as: Cryptography, Pattern matching, Signal Processing, etc.
- Highly programmable (instruction driven) controller
- AMBA AHB 2.0 Master and Slave Interfaces
- Scatter and Gather DMA engine
- C3 has 8 slots for including channels (hardware accelerators). The configuration is as follows:

Channel 0 - Empty

Channel 1 - Data Encryption Standard (DES and TripleDES)

- DES (56 bit keys, no parity)
- DES ECB encryption + decryption
- DES CBC encryption + decryption
- TripleDES (168 bit keys EncDecEnc)
- 3DES ECB encryption + decryption
- 3DES CBC encryption + decryption
- FIFO Size
- Input FIFO: 16x32 bits
- Output FIFO: 16x32 bits

Channel 2 - Advanced Encryption Standard (AES)

Channel ID: 0x0000_3000

Supported Algorithms

- AES (128, 192, 256 bit keys)
- AES ECB encryption + decryption
- AES CBS encryption + decryption“AES Counter Mode encryption + decryption

FIFO Size

- Input FIFO: 16x32 bits
- Output FIFO: 16x32 bits

Channel 3 - Unified Hash with HMAC

Channel ID: 0x0000_4002

Supported Algorithms

- MD5
- Hash with 128 bit digest
- HMAC
- SHA1
- Hash with 160 bit digest
- HMAC
- FIFO Size
- Input FIFO: 16x32 bits
- Output FIFO: 8x32 bits

Channel 4 - Empty

Channel 5 - Empty

Channel 6 - Empty

Channel 7 - Empty

- Number of Instruction Dispatchers is 4. The configuration is as follows:-
  - ID0 - Available
  - ID1 - Empty
  - ID2 - Empty
  - ID3 - Empty
- Number Coupling / Chaining Module (internal cross-bar) for inter channel direct high-speed communications is 1.

21.2 Functional description

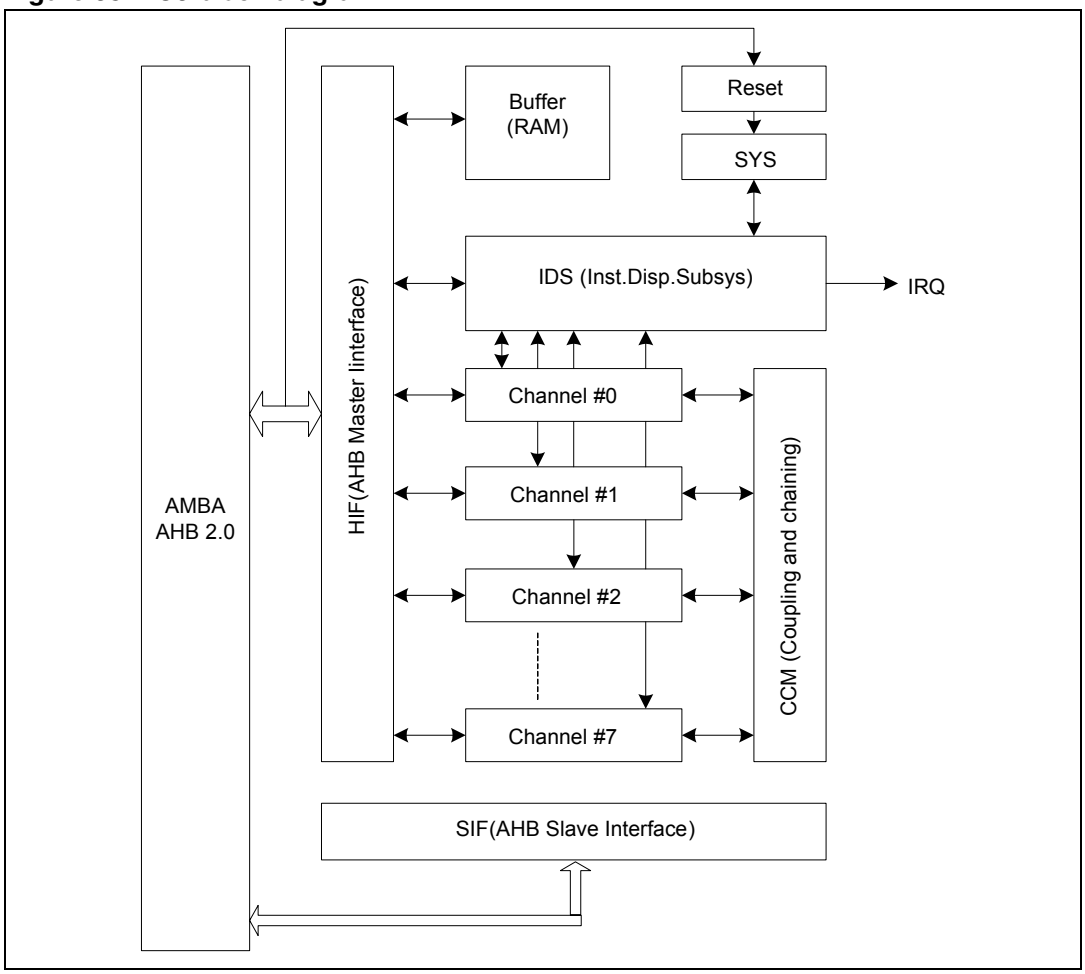
21.2.1 Device summary

Table 310. C3 device summary

Features	C3 version 3
Interfaces	AMBA AHB 2.0 Master Interface AMBA AHB 2.0 Slave Interface
Instruction Dispatchers (IDS) Available	1
Channels Available	3
Coupling/Chaining Paths Available	1

21.3 Block diagram

Figure 35. C3 block diagram



21.4 Main functions description

C3 is a highly programmable DMA based hardware co-processor that executes some instructions flows (programs) written in memory by the host processor. These programs specify which operations must be performed and where to locate data buffers (input, output, parameters) in memory.

After being set-up C3 is completely autonomous and can perform an unlimited number of operations, until it hits an end of program instruction in which case it can signal the end of processing by the means of an interrupt request (if programmed to do so).

C3 has two interfaces:

- **AHB Master Interface**: it is used to fetch instruction flows, to access input data, parameters and to store output data to system memory.
- **AHB Slave Interface**: it is used to set-up the device and to access all the registers.

C3 has been designed to perform acceleration of data-intensive applications where computationally expensive algorithms must operate on medium to large memory buffers.

Such applications can be found in the fields of security (data encryption, integrity check, etc.) and networking.

There are many other fields of application for C3, such as signal processing, image processing and in general applications that require complex mathematical computations

21.4.1 HIF (High speed bus interface)

This block implements an AMBA AHB 2.0 compliant Master interface. It receives internal requests from all the C3 internal initiator blocks and it translates them into AHB bus requests. The HIF is a master only and will not allow access to the C3 internal.

21.4.2 SIF (Slave bus interface)

This block implements an AMBA AHB 2.0 compliant Slave interface. This interface is used to set-up the device and access all the memory mapped internal registers. Each channel is allocated a 1K byte address space and the major blocks of the C3 are also allocated space in the map.

21.4.3 IDS (Instruction dispatchers sub-system)

This block contains the instruction dispatchers (up to 4) that are in charge of fetching the programs from memory and dispatch the instructions to the requested channels. Since the instruction dispatchers operate in parallel up to 4 instruction flows can be executed concurrently.

If programmed to do so the instruction dispatcher signal the end of processing to the host processor by raising an interrupt signal.

Each Instruction Dispatchers (ID) can handle two types of instructions:

- **Flow Type Instructions**: these are C3 control instructions and are executed by the ID itself
- **Application Specific Instructions**: these are processing instructions that are directly dispatched by the ID to the right channel that will perform decoding and execution.

21.4.4 Channel

A Channel is a specific function or set of similar functions and a wrapper to provide data flow management, instruction handling and interface to the various parts of C3. Channels implement the data processing. Total number of channels available in this version of C3 is 3.

Channels operate in parallel and contend the access to the system bus through the HIF arbitration mechanism. Channels are designed to efficiently support the data-flow model of computation.

The channel main features are the following:

- Decoding of the instructions received from the dispatcher sub-system.
- DMA engine with support of scatter and gather operations.
- Internal input / output data-flows buffering by the means of FIFO in order to accommodate for the system bus latency.
- I/O multiplexing, so that data can be directly received from/sent to other channels without going through memory.
- Support for multiple algorithms in the Core block. The Core block implementation is application specific.

21.4.5 CCM (Coupling/Chaining module)

This block implements a cross-bar allowing direct connection of channels Input/Outputs between them. Typical use cases of the CCM are for:

- Sending the output data generated by a channel to the input of another channel, also known as “chaining”.
- Sending the input data received by a channel from the system bus to the input of another channel, also known as “coupling”.

21.5 Processing overview

This section outlines the main steps involved in setting up C3 for processing.

- The host processor creates a program and stores it in memory.
- The program contains C3 instructions and their arguments (usually pointers to data buffers in system memory).
- The host processor writes the base address of the program in the Instruction Pointer Register of one of the Instruction Dispatcher (ID).
- Instruction Dispatchers (IDs) work independently from each other and each ID can handle a C3 instruction flow.
- From the system/software standpoint, each logical device (associated to an Instruction Dispatcher) can be managed by an independent software task, which is to say that a C3 with 4 dispatchers is equivalent to 4 logical co-processors.
- The selected Instruction Dispatcher starts fetching the program from system memory and fills its instruction queue. It then process the instruction flow sequentially, one instruction at a time.
- The ID either executes the instruction itself (in case of Flow Type instruction) or it dispatches the instruction to the specified channel (in case of application specific instruction).
- The channel executes the instruction
- The channel generally performs DMA access request for reading input data and parameters from system memory, but it may be set-up to receive data from another channel.
- The channel Core block (CB) performs the actual data processing, which is specific to the implemented algorithm.
- The channel generally then performs DMA access request for writing the output data to system memory, but it may be set-up to send data to another channel too.
- When the ID hits the end of program it signals completion by rising an interrupt
- Interrupt generation may be disabled, in which case polling by the host processor of the C3 status register can be used to determine the end of processing.

21.6 Programming model

21.6.1 Register map

Most components of C3 have registers mapped in AHB address space starting at the base address 0xD900_0000. Registers are accessed using the C3 AHB Slave Interface (SIF). An address space of 1 KB is allocated for each of these components. The total AHB address window of C3 is 32 KB permitting the mapping of up to 32 components. All registers are 32 bit wide and access to them must be done using aligned 32 bit words read and writes. The current mapping is listed in the *Table 311*

Table 311. C3 components system register summary

Symbol	Name	Offset	Reset Value
C3_SYS	System Registers	0x0000	32'h400
C3_HIF	Master Interfaced Registers	0x0400	32'h400

Table 311. C3 components system register summary (continued)

Symbol	Name	Offset	Reset Value
	unused	0x0800	32'h400
	unused	0x0C00	32'h400
C3_ID0	Instruction Dispatcher #0 Registers	0x1000	32'h400
C3_ID1	unused	0x1400	32'h400
C3_ID2	unused	0x1800	32'h400
C3_ID3	unused	0x1C00	32'h400
C3_CH0	unused	0x2000	32'h400
C3_CH1	Channel #1 Registers	0x2400	32'h400
C3_CH2	Channel #2 Registers	0x2800	32'h400
C3_CH3	Channel #3 Registers	0x2C00	32'h400
	unused	0x3000	32'h400
	unused	0x3400	32'h400
	unused	0x3800	32'h400
	unused	0x3C00	32'h400
	unused	0x4000	32'h400
	unused	0x4400	32'h400
	unused	0x4800	32'h400
	unused	0x4C00	32'h400
	unused	0x5000	32'h400
	unused	0x5400	32'h400
	unused	0x5800	32'h400
	unused	0x5C00	32'h400
	unused	0x6000	32'h400
	unused	0x6400	32'h400
	unused	0x6800	32'h400
	unused	0x6C00	32'h400
	unused	0x7000	32'h400
	unused	0x7400	32'h400
	unused	0x7800	32'h400
	unused	0x7C00	32'h400

21.6.2 System registers (C3_SYS)

System registers are registers whose scope is the whole C3.

21.6.3 Register configuration

Table 312 summarizes AHB mapped registers for the system (SYS).

Table 312. C3 components system registers map

Symbol	Name	Type	Reset value	Offset
SYS_SCR	Status and control register	RW	-	0x000
SYS_STR	Channel status register	RO	-	0x040
SYS_VER	Hardware version and revision	RO	VER	0x3F0
SYS_HWID	Hardware ID	RO	HWID	0x3FC

Zero is read from undefined locations, writing has no effect.

21.6.4 Register description

Bit	31	30	29	28	27	26	25	24
Symbol	ID3SH	IDS3L	IDS2H	IDS2L	IDS1H	IDS1L	IDS0H	IDS0L
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	23	22	21	20	19	18	17	16
Symbol	ISD	ISD2	ISD1	ISD0	ISA	CISR	BEND	ARST
Initial Value	0	0	0	0	0	0	0	0
Type	R(W)	R(W)	R(W)	R(W)	R(W)	R(W)	R(W)	R(W)

Bit	15	14	13	12	11	10	9	8
Symbol	C7SH	C7SL	C6SH	C6SL	C5SH	C5SL	C4SH	C4SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	7	6	5	4	3	2	1	0
Symbol	C3SH	C3SL	C2SH	C2SL	C1SH	C1SL	C0SH	C0SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit 31 to 24 - Instruction dispatcher n status (IDnS)

The status of each Instruction Dispatcher is mirrored in these bits. Bits 31-30 are the state of ID3, bits 29-28 of ID2, bits 27-26 of ID1 and bits 25-24 of ID0. These bits are the same as

the ones in the Instruction Dispatcher Status and Control Register (ID_SCR) of each ID. These bits allow knowing the status of all Instruction Dispatcher with a single AHB slave read. See the Instruction Dispatcher document section for more details

Bit 23 to 20 - Instruction dispatcher n interrupt status (ISDn)

Interrupt States (IS) of every Instruction Dispatcher are made available in these bits. These bits are the same than the ones in the Instruction Dispatcher Status and Control Register (ID_SCR) of each ID. Interrupts can be acknowledged using the Status and Control Register of the Instruction Dispatcher (ID_SCR) or using these bits. See the Instruction Dispatcher document section for more details.

Bit 23-20 ISDn	Description
1'b1	The Instruction Dispatcher n is requesting an Interrupt.
1'b0	(Clearing conditions) The Interrupt Status (IS) of Instruction Dispatcher n can be cleared writing one to this flag. Writing zero has no effect.

Bit 19 - Interrupt status of all instruction dispatchers (ISA)

The Interrupt Status of All Instruction Dispatchers (ISA) is the logical OR of bits ISD3-ISDO. This bit represents the state of the Interrupt pin of the C3 document. Writing one to this flag has the same effect as writing one in all ISD3-ISDO.

Bit 19 ISA	Description
1'b1	At least one Instruction Dispatcher is requesting an Interrupt.
1'b0	(Clearing conditions) The Interrupt Status (IS) of Instruction Dispatcher can be cleared writing one to this flag. Writing zero has no effect.

Bit 18 - Clear interrupt status on read (CISR)

If the Clear Interrupt Status on Read bit (CISR) is set, clearing of Interrupt States is performed by reading the Status and Control Register of the System (SYS_SCR). The Status and Control Register of Instruction Dispatchers (ID_SCR) is not affected by this bit.

Bit 18 CISR	Description
1'b1	Reading SYS_SCR dears Interrupt States of all Instruction Dispatchers.
1'b0	Do not clear Interrupt States on SYS_SCR read.

Bit 17 - Big endian (BEND)

Not implemented. This bit should be set to zero.

Bit 16 - Asynchronous master reset (ARST)

The whole C3 can be reset using this bit. The reset is done asynchronously in Hardware thus guaranteeing a well known state after its execution. A special Hardware block takes care of correct timings for the reset sequence. It takes about 6 clock cycles for the Hardware reset. The Internal Memory may not be cleared.

Bit 16 ARST	Description
1'b1	Reset the whole C3.
1'b0	(Clearing conditions) This bit is cleared as a consequence of the reset, so it is always read zero. Writing zero has no effect.

Bit 15 to 0 - Channel n status (CnS)

The status of each Channel is mirrored in these bits. These bits are the same ones as found in the Instruction Dispatcher Status and Control Register (ID_SCR). See the Instruction Dispatcher document section for more details. To know the status of the other 8 Channels (Channels 8 to 15) you must use the Channel Status Register (SYS_STR).

Status and control register (SYS_SCR)

Bit	31	30	29	28	27	26	25	24
Symbol	C15SH	C15SL	C14SH	C14SL	C13SH	C13SL	C12SH	C12SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	23	22	21	20	19	18	17	16
Symbol	C11SH	C11SL	C10SH	C10SL	C9SH	C9SL	C8SH	C8SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	15	14	13	12	11	10	9	8
Symbol	C7SH	C7SL	C6SH	C6SL	C5SH	C5SL	C4SH	C4SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	7	6	5	4	3	2	1	0
Symbol	C3SH	C3SL	C2SH	C2SL	C1SH	C1SL	C0SH	C0SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit 31 to 0 - Channel n status (CnS)

The status of each Channel is mirrored in these bits. The lower 16 bits (bits 15 to 0) are the same ones as found in the Instruction Dispatcher Status and Control Register (ID_SCR) and in the System Status and Control Register (SYS_SCR). See the Instruction Dispatcher registers description (section 3) for more details. The upper 16 bits (bits 31 to 16) represents the status of Channels 8 to 15. Using this register is the only way to know the status of Channels 8 to 15.

Hi Bit CnSH	LoBit CnSL	Description
0	0	Not Present: This Channel does not exist in Hardware.
1	0	Idle: The Channel is idle and instructions can be dispatched to it.
1	1	Busy: The Channel is executing instructions dispatched by an Instruction Dispatcher.
0	1	Error: The Channel is in error state, use Channel registers to know the cause.

Hardware Version and Revision Registers (SYS_VER)

The Hardware Version and Revision Register (SYS_VER) contain the RTL source version from which the Hardware was generated.

Bit	31	30	29	28	27	26	25	24
Symbol	V7	V6	V5	V4	V3	V2	V1	V0
Initial Value	0	0	0	0	0	0	1	1
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	23	22	21	20	19	18	17	16
Symbol	R7	R6	R5	R4	R3	R2	R1	R0
Initial Value	R7	R7	R7	R7	R7	R7	R7	R7
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	15	14	13	12	11	10	9	8
Symbol	S15	S14	S13	S12	S11	S10	S9	S8
Initial Value	S15	S14	S13	S12	S11	S10	S9	S8
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	7	6	5	4	3	2	1	0
Symbol	S7	S6	S5	S4	S3	S2	S1	S0

Bit	7	6	5	4	3	2	1	0
Initial Value	S7	S6	S5	S4	S3	S2	S1	S0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit 31 to 24 - Hardware Version

Bits V7-V0 represents the Version. This is always 3 (the v3 part in C3v3).

Bit 23 to 16 - Hardware Revision

Bits R7-R0 represents the RTL Revision.

Bit 15 to 0 - Hardware Sub-revision

Bits S15-S0 represents the RTL Sub-revision. For example the version of a C3v3 RTL source tree 3.1.5 is identified by Vn set to 3, Rn set to 1 and Sn set to 5.

Hardware ID Register (SYS_HWID)

The Hardware ID register contains the Identifier of the Hardware. The Hardware ID has no bit-field structure: the value is a mere index in a database table. There is currently no maintained Hardware IDs Table. There are however a bunch of reserved Hardware IDs:

HWID	Usage
32'h0000_0000	Illegal Value
32'h1234_5678	Endianess Test
32'hFFFF_xxxx	Prototype on Programamble Logic

21.6.5 Master interface register (C3_HIF)

The Master Interface (HIF) interfaces Channels and Instruction Dispatchers (ID) to the Initiator Bus and to an Internal Memory (IM). The purpose of the HIF is to allow read and write accesses generated by Channels and Instruction Dispatchers to be transferred to an Initiator Bus or to the Internal Memory. An arbiter in the HIF prevents data access collisions from occurring. ID0 has the highest priority to perform accesses on this block followed in order by ID1 to ID3 and Channels #0 to #15 (lowest priority). Read Transfers have higher priority than Write Transfers.

The HIF is able to route requests to an internal Memory instead of the Bus if this Memory is enabled (using a configuration bit in HIF_MCR). The maximum size of the Internal Memory is 64 KB and is always 32 bit wide.

HIF is also able to route requests to a Byte Bucket if this is enabled (using a configuration bit in HIF_NCR).

Transactions can simultaneously occur on the Bus, on the Internal Memory and on the Byte Bucket. A Base Address for transactions that must target the Internal Memory or the Byte Bucket instead of the Bus must be programmed in the HIF prior to utilizing the Internal Memory.

Write transaction requests coming from IDs or Channels that are within an address window of 64 KB starting from the programmed Byte Bucket Base Address (HIF_NBAR) will be routed to the Byte Bucket. That is, every thing written to this address window is thrown away.

Read transactions from this address window are not affected by the Byte Bucket: they are normally routed either to the Internal Memory or to the Bus.

Transaction requests coming from IDs or Channels that are within an address window of 64 KB starting from the programmed Memory Base Address (HIF_MBAR) will be routed to the Internal Memory. Higher addresses of the internal Memory window are aliased if the Internal Memory is smaller than 64 KB.

The Byte Bucket has priority if both the Byte Bucket Base Address and the Memory Base Address are programmed with the same addresses.

A burst transaction always completes on the initial target even if addresses span two different targets.

The Move Channel (move_cnl) can be used to transfer data to/from the Internal Memory from/to the Bus and vice versa. Internal Memory content is undefined at startup or after an asynchronous master reset.

The other way to access the internal Memory contents is making transfers to the C3 AHB Slave Interface. There are two different methods to achieve this: mapping a 512 Bytes page of the Internal Memory into AHB address space (HIF_MP) and/or using a pair of Address and Data Registers (HIF_MAAR and HIF_MADR) to access single locations.

The internal Memory can be accessed by an ID or Channel and simultaneously from the AHB Slave Interface.

21.6.6 Register configuration

Table 313. contains the AHB mapped registers for the Master Interface (HIF).

Table 313. AHB mapped registers for Master Interface (HIF)

Symbol	Name	Type	Initial Value	Address
HIF_MP	Memory Page	RW	-	0x000 to 0x1FF
HIF_MSIZE	Memory Size in Bytes	RO	MSIZE	0x300
HIF_MBAR	Memory Base Address Register	R/W	32’h0	0x304
HIF_MCR	Memory Control Register	R/W	32’h0	0x308
HIF_MPBAR	Memory Page Base Address Register	R/W	32’h0	0x30C
HIF_MAAR	Memory Access Address Register	R/W	32’h0	0x310
HIF_MADR	Memory Access Data Register	R/W	-	0x314
HIF_NBAR	Byte Bucket Base Address Register	R/W	32’h0	0x344
HIF_NCR	Byte Bucket Control Register	R/W	32’h0	0x348

Zero is read from undefined locations, writing has no effect.

21.6.7 Register Description

21.6.8 Memory page (HIF_MP)

A 512 Bytes page of the Internal Memory is mapped in the Memory Page address range (HIF_MP, 0x000 to 0x1FF). The page number to be mapped is programmed using the Memory Page Base Address Register (HIF_MPBAR). AHB Reads and Writes to the C3 Slave Interface in this address space leads to Internal Memory access.

21.6.9 Memory size register (HIF_MSIZE)

The content of this register represents the size of the internal Memory in Bytes. If an internal Memory does not exist in Hardware, this register will be zero. This is a way for the Software to know if an internal Memory is there and what its size is. The maximum memory size is 64KB so the maximum value of MSIZE is 0x10000. The lower 2 bits of MSIZE are always zero (only 32 bit wide memories are supported).

Bit	31	30	29	28	27	26	25	24
Symbol	-	-	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	23	22	21	20	19	18	17	16
Symbol	-	-	-	-	-	-	-	S16
Initial Value	0	0	0	0	0	0	0	S16
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	15	14	13	12	11	10	9	8
Symbol	S15	S14	S13	S12	S11	S10	S9	S8
Initial Value	S15	S14	S13	S12	S11	S10	S9	S8
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	7	6	5	4	3	2	1	0
Symbol	S7	S6	S5	S4	S3	S2	-	-
Initial Value	S7	S6	S5	S4	S3	S2	-	-
Type	RO	RO	RO	RO	RO	RO	RO	RO

21.6.10 Memory base address register (HIF_MBAR)

The Base Address of the Internal Memory can be programmed to any multiple of 64 KB. Bits 31-16 of MBAR are used for this. Channel and Instruction Dispatcher transactions that fall within a window of 64 KB starting from MBAR are then routed to the Internal Memory (if enabled). The Internal Memory Base Address can be changed at any time but behavior of the active transactions done in this range is undefined. The Byte Bucket has priority if its Base Address (NBAR) is programmed with the same value as MBAR.

Bit	31	30	29	28	27	26	25	24
Symbol	B31	B30	B29	B28	B27	B26	B25	B24
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Symbol	B23	B22	B21	B20	B19	B18	B17	B16
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Symbol	-	-	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	7	6	5	4	3	2	1	0
Symbol	-	-	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

21.6.11 Memory control register (HIF_MCAR)

The Internal Memory must be enabled to allow Channels and Instruction Dispatchers to access it. This is done using the Enable Memory Mapping bit (EMM). The correct procedure for the Software to enable the Internal Memory is to first program its base address using HIF_MBAR and then enable it by setting the EMM bit of HIF_MCR. The Internal Memory can be enabled or disabled at any time but the behaviour of the active transactions done in this range is undefined.

Normally, when using the Address and Data registers pair (HIF_MAAR and HIF_MDAR) to access Internal Memory locations from AHB, the Address register is auto incremented. To disable this feature Disable Auto Increment on Read and Disable Auto Increment Write bits (DAIR and DAIW) are offered.

Bit	31	30	29	28	27	26	25	24
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	23	22	21	20	19	18	17	16
Symbol	res	res	res	res	res	res	DAIR	DAIW
Initial Value	-	-	-	-	-	-	0	0
Type	-	-	-	-	-	-	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Symbol	res	res	res	res	res	res	res	BMM
Initial Value	-	-	-	-	-	-	-	0
Type	-	-	-	-	-	-	-	R/W

- Bit 31 to 18, 15 to 1 - Reserved

These bits are reserved and should be set to zero.

Bit 17 - Disable Auto Increment on Read (DAIR)

Bit 17 DAIR	Description
1’b0	Memory Access Address Register (HIF_MAAR) is auto incremented when an Internal Memory location is read from AHB using the Memory Access Data Register (HIF_MADR).
1’b1	Memory Access Address Register (HIF_MAAR) auto increment is disabled on HIF_MADR reads.

Bit 16 - Disable Auto Increment on Write (DAIW)

Bit 16 DAIR	Description
1'b0	Memory Access Address Register (HIF_MAAR) is auto incremented when an Internal Memory location is read from AHB using the Memory Access Data Register (HIF_MADR).
1'b1	Memory Access Address Register (HIF_MAAR) auto increment is disabled on HIF_MADR writes.

Bit 0 - Enable Memory Mapping (EMM)

Bit 0 EMM	Description
1'b0	Disable the Internal Memory. Transactions from Channels and Instruction Dispatchers go either to the Bus or the Byte Bucket (if enabled). AHB slave accesses to the Internal Memory are not affected by this bit they are always enabled.
1'b1	Enable the Internal Memory.

21.6.12 Memory page base address register (HIF_MPBAR)

Bit	31	30	29	28	27	26	25	24
Symbol	B31	B30	B29	B28	B27	B26	B25	B24
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	23	22	21	20	19	18	17	16
Symbol	B23	B22	B21	B20	B19	B18	B17	B16
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	15	14	13	12	11	10	9	8
Symbol	P15	P14	P13	P12	P11	P10	P9	-
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Symbol	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

A 512 Bytes page of the Internal Memory is mapped on the AHB address space HIF_MP (0x000 to 0x01FF). The page is selected using bits P15-P9 of this registers. Bits B31-B16 (read-only) are those programmed in the Internal Memory Base Address Register (HIF_MBAR).

21.6.13 Memory access address register (HIF_MAAR)

Bit	31	30	29	28	27	26	25	24
Symbol	B31	B30	B29	B28	B27	B26	B25	B24
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	23	22	21	20	19	18	17	16
Symbol	B23	B22	B21	B20	B19	B18	B17	B16
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	15	14	13	12	11	10	9	8
Symbol	A15	A14	A13	A12	A11	A10	A9	A8
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Symbol	A7	A6	A5	A4	A3	A2	-	-
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO

AHB slave accesses to the Memory Access Data Register (HIF_MADR) targets the Internal Memory location programmed in the Memory Access Address Register (HIF_MAAR). Bits A15-A2 are used for this. Bits B31-B16 (read only) are those programmed in the Internal Memory Base Address Register (HIF_MBAR). Bits 1-0 are always zero since only aligned 32 bit transactions are supported.

21.6.14 Memory access data register (HIF_MADR)

The Internal Memory location which address is programmed in the Memory Access Address Register (HIF_MAAR) can be accessed reading and writing the Memory Access Data Register (HIF_MADR). By default, when reading or writing the Memory Access Data Register, the Memory Access Address Register is auto incremented. This feature can be disabled setting bits Disable Auto Increment on Read (DAIR) and/or Disable Auto Increment on Write (DAIW) in the Memory Control Register (HIF_MCR).

21.6.15 Byte bucket base address register (HIF_NBAR)

The Base Address of the Byte Bucket can be programmed to any multiple of 64 KB. Bits 31-16 of NBAR are used for this. Channel and Instruction Dispatcher write transactions that fall within a window of 64 KB starting from NBAR are then discarded by the Byte Bucket (if enabled). The Byte Bucket Base Address can be changed at any time but the behaviour of the active transactions done in this range is undefined. The Byte Bucket has priority if its Base Address (NBAR) is programmed with the same value as the Memory Base Address (MBAR). Read transactions are ignored by the Byte Bucket and are always routed either to the Bus or the Memory.

Bit	31	30	29	28	27	26	25	24
Symbol	B31	B30	B29	B28	B27	B26	B25	B24
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Symbol	B23	B22	B21	B20	B19	B18	B17	B16
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Symbol	-	-	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	7	6	5	4	3	2	1	0
Symbol	-	-	-	-	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

21.6.16 Byte bucket control register (HIF_NCR)

The Byte Bucket must be enabled to allow Channels and Instruction Dispatchers to discard data using it. This is done using the Enable Byte Bucket Mapping bit (ENM). The correct procedure for the Software to enable the Byte Bucket is to first program its base address using HIF_NBAR and then enable it by setting the ENM bit of HIF_NCR. The Byte Bucket can be enabled or disabled at any time but the behaviour of the active transactions done in this range is undefined.

Bit	31	30	29	28	27	26	25	24
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	23	22	21	20	19	18	17	16
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	15	14	13	12	11	10	9	8
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Symbol	res	res	res	res	res	res	res	ENM
Initial Value	-	-	-	-	-	-	-	0
Type	-	-	-	-	-	-	-	R/W

- Bit 31 to 1 - Reserved

These bits are reserved and should be set to zero.

- Bit 0 - Enable Byte Bucket Mapping (ENM)

Bit 17 DAIR	Description
1’b0	Disable the Byte Bucket. Transactions from Channels and Instruction Dispatchers go either to the Bus or the Memory (if enabled).
1’b1	Enable the Byte Bucket.

21.6.17 Instruction dispatcher registers (C3_IDn)

Up to four Instruction Dispatchers can exist in Hardware. Each Instruction Dispatcher has its own set of registers.

Register Configuration

Table 314. summarizes AHB mapped registers for an Instruction Dispatcher (ID).

Table 314. AHB mapped registers for an Instruction Dispatcher (ID)

Symbol	Name	Type	Initial Value	Address
ID_SCR	Status and Control Register	R/W	-	0x000
ID_IP	Instruction Pointer	R/W	32'h0	0x010
ID_IR0	Instruction Word 0 Register	RO	32'h0	0x020
ID_IR1	Instruction Word 1 Register	RO	32'h0	0x024
ID_IR2	Instruction Word 2 Register	RO	32'h0	0x028
ID_IR3	Instruction Word 3 Register	RO	32'h0	0x02C

Status and Control Register (ID_SCR)

Bit	31	30	29	28	27	26	25	24
Symbol	IDSH	IDSL	BERR	res	res	CERR	CBSY	CDNX
Initial Value	0 or 1	0	0	-	-	0	0	0
Type	RO	RO	RO	-	-	RO	RO	RO

Bit	23	22	21	20	19	18	17	16
Symbol	IS	IES	IER	SSC	SSE	res	IGR	RST
Initial Value	0	0	0	0	0	-	0	0
Type	R/(W)	R/W	R/W	R/W	R/W	-	R/W	R/(W)

Bit	15	14	13	12	11	10	9	8
Symbol	C7SH	C7SL	C6SH	C6SL	C5SH	C5SL	C4SH	C4SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit	7	6	5	4	3	2	1	0
Symbol	C3SH	C3SL	C2SH	C2SL	C1SH	C1SL	C0SH	C0SL
Initial Value	0 or 1	0	0 or 1	0	0 or 1	0	0 or 1	0
Type	RO	RO	RO	RO	RO	RO	RO	RO

Bit 31 to 30 - Instruction Dispatcher Status (IDS)

The Instruction Dispatcher Status bits (IDSn) indicates the state in which the addressed Instruction Dispatcher (ID) is. The Software can use these bits at system startup to know if an ID is present. In normal operation mode the Software uses these bits to know the reason of an Interrupt or to know when a program execution has finished if in polling mode.

Bit 31 IDSH	Bit 31 IDSH	Description
0	0	Not Present: This Instruction Dispatcher does not exist in Hardware.
1	0	Idle: The Instruction Dispatcher has successfully terminated the execution of a program and is ready to accept a new Instruction Pointer.
1	1	Run: The Instruction Dispatcher is executing program.
0	1	Error: The Instruction Dispatcher has stopped the execution of a program because of an error. Error cause can be analyzed using bits 29-24 of ID_SCR.

The Instruction Dispatcher exits the Error state in three ways: resetting the Instruction Dispatcher (the ID goes to Idle state), launching a new program (the ID goes to Run state) or requesting an asynchronous master reset.

Bit 29 - Bus Error (BERR)

Every module attached to the HIF receives its own Bus error signal. This signal is set by the HIF if a bus error condition is detected for a Bus transaction initiated by the corresponding module.

Bit 29 DERR	Description
1'b0	The HIF reported a bus error condition for a transaction initiated by this Instruction Dispatcher.
1'b1	(Cleaning Conditions) This flag is cleared in three ways: resetting the Instruction Dispatcher, launching a new program or requesting an asynchronous master reset.

Bit 28 to 27 - Reserved

These bits are reserved and should be set to zero.

Bit 26 - Channel Error (CERR)

Channels report their states to each Instruction Dispatcher. When the ID dispatches an instruction to a Channel that is in error state or if the Channel goes to error state when it executes the received instruction, the Instruction Dispatcher goes in turn in error state and this bit is set.

Bit 26 DERR	Description
1’b1	The Channel to which the current instruction was addressed is in error state or went to error state executing the instruction.
1’b0	(Cleaning Conditions) This flag is cleared in three ways: resetting the Instruction Dispatcher, launching a new program or requesting an asynchronous master reset.

Bit 25 - Channel Busy (CBSY)

Bit 25 CBSY	Description
1’b1	The Channel to which the current instruction was addressed is busy. It is already running under control of another Instruction Dispatcher.
1’b0	(Cleaning Conditions) This flag is cleared in three ways: resetting the Instruction Dispatcher, launching a new program or requesting an asynchronous master reset.

Bit 24 - Channel Does Not Exist (CDNX)

Bit 24 CDNX	Description
1’b1	The Channel to which the current instruction was addressed does not exist in Hardware.
1’b0	(Cleaning Conditions) This flag is cleared in three ways: resetting the Instruction Dispatcher, launching a new program or requesting an asynchronous master reset.

Bit 23 - Interrupt Status (IS)

The Interrupt Status (IS) bit reflects the status of the Interrupt port of the Instruction Dispatcher. Interrupt ports of every Instruction Dispatcher are “ORed” together to generate the final Interrupt signal which drives the Interrupt Pin of the C3 component.

Bit 23 IS	Description
1’b1	The Instruction Dispatcher is requesting an Interrupt because IES and / or IER are set and one of their corresponding event occured.
1’b0	(Cleaning Conditions) This flag is cleared writing one to it, resetting the Instruction Dispatcher or requesting an asynchronous master reset. Launching a new program will not clear this flag. Writing zero has no effect.

Bit 22 - Interrupt Enable on Stop (IES)

Bit 22 IES	Description
1'b1	The Instruction Dispatcher generates an Interrupt on normal termination of a program execution (when the stop instruction executes).
1'b0	Do not generate Interrupt. Cleaning this bit does not clear pending interrupts.

Bit 21 - Interrupt Enable on Error (IER)

Bit 21 IER	Description
1'b1	The Instruction Dispatcher generates an Interrupt when a program encounters an error. The error cause can be analyzed through bits 29-24 of ID_SCR.
1'b0	Do not generate Interrupt. Cleaning this bit does not clear pending Interrupts.

Bit 20 - Single Step Command (SSC)

If the Instruction Dispatcher is put in Single Step Mode using bit SSE of ID_SCR it will await for SSC to be set before executing/dispatching the next instruction. In this context, a Single Step is defined as the execution/dispatching of the instruction and its argument. The first instruction is not executed/dispatched launching a new program when SSE is set.

Bit 20 SSC	Description
1'b1	Writing one to this flag in Single Step Mode (SSE is set) executes / dispatches the next instruction.
1'b0	(Cleaning Conditions) This bit is cleared when the execution of the current single instruction terminates. Writing zero has no effect.

Bit 19 - Single Step Enable (SSE)

Bit 19 SSE	Description
1'b1	Enable Single Step Mode. This bit can be changed anytime.
1'b0	Disable Single Step Mode. Exiting Single Step Mode clears also SSC.

Bit 18 - Reserved

This bit is reserved and should be set to zero.

Bit 17 - Ignore Errors (IGR)

Not implemented. This bit should be set to zero.

Bit 16 - Reset Command (RST)

Each Instruction Dispatcher can be reset independently from each other using this bit. In Hardware the reset is done synchronously and not all registers are affected by it. The following are the effects of a synchronous reset:

- bits 29-16 of SCR are all cleared,
- the Instruction Dispatcher goes in Idle state eventually aborting program execution and bits 31-30 (IDS) of SCR are set to Idle.

Bit 16 RST	Description
1'b1	Reset this Instruction Dispatcher.
1'b0	(Cleaning Conditions) This bit is cleared as a consequence of the reset,. so it is always read zero. Writing zero has no effect.

Bit 15 to 0 - Channel n Status (CnS)

All sixteen Channels report their state to each Instruction Dispatcher. The status of the first eight Channels (Channel 0 to 7) is mirrored into these bits. A Channel status is encoded into two bits of the SCR: status for Channel #0 is in bits 1-0, for Channel #1 in bits 3-2 and so on till Channel #16 in bits 31-30. You must use the System Channel Status Register (SYS_STR) to know the status of all 16 Channels.

Hi Bit CnSH	Lo Bit CnSL	Description
0	0	Not Present: This Channel does not exist in Hardware.
1	0	Idle: The Channel is Idle and instructions can be dispatched to it.
1	1	Busy: The Channel is executing instructions dispatched by an Instruction Dispatcher.
0	1	Error: The Channel is in error state, use Channel registers to know the cause.

Instruction Pointer Register (ID_IP)

The Instruction Pointer Register is used to store the pointer of the first instruction to be fetched and to launch program execution. It can be read back at any time (particularly in Single Step Mode) to know the address of the next instruction that will be executed. Effects of changing Instruction Pointer while a program is running are unspecified. The Instruction Pointer must be 32 bit aligned (the lower two bits are ignored and are always read zero). When an Instruction Pointer is written the Instruction Dispatcher goes in run state, begins filling its Instruction Queue and as soon as the first instruction is available it executes/dispatches it.

Instruction Word 0-3 Register (ID_IRn)

Instruction Word Registers are used to read back the OP Code of the current executing instruction. The instruction can be 1 to 4 words long. IR1-3 contents are undefined for 1 word instructions, IR2-3 contents are undefined for 2 word instructions and, similarly, IR3 is undefined for 3 word instructions. These register are used mainly in Single Step Mode to read back the last executed instruction.

21.6.18 Channel registers (C3_CHn)

Each Channel has its own specific set of registers. See the Channel User Manual to know more about them. There is however one register that is mandatory to each Channel: the Channel Identity Register (CH_ID). This read-only register is mapped in a fixed location and it is typically used by the SW (at system startup) to know which Channels are available in the C3.

Register Configuration

Table 315. summarizes AHB mapped registers for a Channel (CH).

Table 315. AHB mapped registers for Channel (CH)

Symbol	Name	Type	Initial Value	Address
	Channel Specific Registers	-	-	0x000-0x3FB
CH_ID	Channel ID	RO	CH_ID	0x3FC

21.6.19 Channel ID register (CH_ID)

The Channel ID register contains the Identifier of the Channel. The Software knows that a Channel is not present reading zero from this register (or using the ID_SCR or the SYS_SCR). The Channel ID has no bit-field structure: the value is a mere index in a database table.

The database containing all the assigned IDs is provided in a separate document [CH_ID_TABLE].

There is a unique 32 bit channel ID associated to a channel/version pair. In order to avoid using an already allocated channel ID new channel developers should contact the C3 project team to obtain unique numbers for their channels. Such centralized allocation enable maintaining interoperability between all channel libraries and the baseline C3 platform.

Before designing a new Channel please consider looking at this table to see if a Channel that performs a similar function does not already exist.

21.7 Channel ID

Each Channel/Version is assigned a unique 32 bit Identifier that can be read in the Channel ID register of each channel.

The upper 8 bit of the Channel ID are reserved for specific organizations. In order to manage the situation in which different versions of the same channel are developed by different company organizations the organization specific bit indicates which organization is

in charge of maintaining the corresponding version of the channel. In the device, the bit mask to be used for retrieving this ID is 0x8000_0000.

Table 316. Channel ID Table

Channel No.	Channel Name	Function	Type	Channel ID
0	EMPTY_CNL	No channel	-	-
1	DES_CNL	DES/3DES algorithm (ECB, CBC modes)	Cryptography	0x00002000
2	AES_CNL	AES algorithm (ECB, CBC, CTR modes)	Cryptography	0x00003000
3	UH_CNL	SHA-1/MD5 + HMAC algorithms	Cryptography	0x00004002
4	EMPTY_CNL	No Channel	-	-
5	EMPTY_CNL	No Channel	-	-
6	EMPTY_CNL	No Channel	-	-
7	EMPTY_CNL	No Channel	-	-

21.8 DES channel

21.8.1 Overview

This Channel can compute DES and 3DES encryption and decryption in ECB and CBC mode; executing C3 Flow type DES [START/APPEND] instruction.

21.8.2 Instruction set

The DES Channel executes DES [START/APPEND] ENCRYPT and DES [START/APPEND] DECRYPT instructions. Instructions that do not conform to the following bit encodings are unknown to the DES Channel that will go in error state.

21.8.3 DES instructions

There are 2 different DES instructions:

- DES START
- DES APPEND

The first instruction is used for setting the operation parameters, such as the key and the initialization vector. The second one is used for passing the data to encrypt or decrypt.

21.8.4 DES START instruction

The DES START instruction can be applied with 2 different modes of operation:

- ECB
- CBC

21.8.5 ECB

The DES START ECB instruction is 2 words long. This instruction is used to set the key for the following operations. The length of the key is encoded in the first instruction word, while the second word represents the Source Address for the key.

Table 317. DES ECB start instruction bit encoding

W#	Bit Encoding
1	xxxx01ab 000x xxxx cccc cccc cccc cccc
2	(32 bit Source Address for the key)

Bit 'a' in the above table is used to set the algorithm to use:

Table 318. DES ECB bit 'a' encoding

Bit 25	
a	Operation
0	DES
1	3DES

Bit 'b' in the above table is used to set the operation to perform:

Table 319. DES ECB bit 'b' encoding

Bit 24	
b	Operation
0	Encryption
1	Decryption

Bits 15 to 0 in the first instruction word (cccc in the above table) represent the length in Bytes of the key.

21.8.6 CBC

The DES START CBC instruction is 3 words long. This instruction is used to set the key and the initialization vector for the following operations. The length of the key is encoded in the first instruction word, the second word represents the Source Address for the key and the third word represents the Source Address for the Initialization Vector.

Table 320. DES CBC START Instruction Bit Encoding

W#	Bit Encoding
1	xxxx 10ab 001x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the key
3	32 bit Source Address for the IV

Bits ‘a’ and ‘b’ in the above table are used to set the algorithm and the operation to perform and have the same encoding as in the ECB instruction. Bits 15 to 0 in the first instruction word (cccc in the above table) represent the length in Bytes of the key.

21.8.7 DES APPEND instruction

The DES APPEND instruction can be applied with 3 different modes of operation:

- ECB
- CBC

21.8.8 ECB

The DES APPEND ECB instruction is 3 words long. This instruction is used for passing the data to process (encrypt or decrypt). The length of the data to process is encoded in the first instruction word, the second word represents the Source Address and the third word represents the Destination Address.

Table 321. DES ECB APPEND Instruction bit encoding

W#	Bit Encoding
1	xxxx 10ab 100x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the data
3	32 bit Destination Address for the data

Bit a in the above table is used to set the algorithm to use:

Bit 25	
a	Operation
0	DES
1	3DES

Bit b in the above table is used to set the operation to perform:

Bit 25	
b	Operation
0	Encryption
1	Decryption

Bits 15 to 0 in the first instruction word (cccc in the above table) represent the length in Bytes of the data to be processed.

21.8.9 CBC

The DES APPEND CBC instruction is 3 words long. This instruction is used for passing the data to process (encrypt or decrypt). The length of the data to process is encoded in the first

instruction word, the second word represents the Source Address and the third word represents the Destination Address.

Table 322. DES CBC Append Instruction Bit Encoding

W#	Bit Encoding
1	xxxx 10ab 101x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the data
3	32 bit Destination Address for the data

Bits ‘a’ and ‘b’ in the above table are used to set the algorithm and the operation to perform and have the same encoding as in the ECB instruction (*Table 318.* and *Table 319.*). Bits 15 to 0 in the first instruction word (cccc in *Table 322.*) represent the length in Bytes of the key.

21.8.10 Register set

Table 323. DES registers map

Symbol	Name	Type	Initial value	Address
DES_DATA_INOUT_HI	Data input/output register #0	R/W	32’h0	0x000
DES_DATA_INOUT_LO	Data input/output register #1	R/W	32’h0	0x004
DES_FEEDBACK_HI	Feedback register #0	R/W	32’h0	0x008
DES_FEEDBACK_LO	Feedback register #1	R/W	32’h0	0x00C
DES_CONTROL_STATUS	Control and status register	R/(W)	32’h0	0x010
DES_KEY1_HI	Key register #0	R/W	32’h0	0x020
DES_KEY1_LO	Key register #1	R/W	32’h0	0x024
DES_KEY2_HI	Key register #2	R/W	32’h0	0x028
DES_KEY2_LO	Key register #3	R/W	32’h0	0x02C
DES_KEY3_HI	Key register #4	R/W	32’h0	0x030
DES_KEY3_LO	Key register #5	R/W	32’h0	0x034
DES_IR	Channel ID register	RO	32’h0	0x3FC

21.8.11 DES register description

Note: Changing the register values while the DES Channel is executing an instruction may produce wrong results and unexpected behaviors.

21.8.12 Data input/output registers (DES_DATAINOUT)

The same address refers to 2 different blocks of registers, depending on the operation (read or write). The Data Input Registers contain the current data input to the DES Channel (accessed using the write operation). The Data Output Registers contain the current data output of the DES Channel (accessed using the read operation).

Note: A read operation on these registers just after a write operation will not return the same value previously written.

21.8.13 Feedback registers (DES_FEEDBACK)

The Feedback Registers contain the value that is added to the DES input for implementing the selected mode of operation (it depends on the selected mode).

21.8.14 Control and status register (DES_CONTROL_STATUS)

Bit	31	30	29	28	27	26	25	24
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	23	22	21	20	19	18	17	16
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	15	14	13	12	11	10	9	8
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Symbol	res	res	res	res	res	ED	MODE	ALGO
Initial Value	-	-	-	-	-	0	0	0
Type	-	-	-	-	-	R/W	R/W	R/W

- Bits 31 to 3 - Reserved

These bits are reserved and should be written zero.

- Bit 2 - Encryption/Decryption (ED)

This bit indicates the operation to perform (Encryption or Decryption).

Bit 2	Description
1’b0	Encryption
1’b1	Decryption

- Bit 1 - Mode of operation (MODE)

This bit indicates the mode of operation (ECB or CBC).

Bit 1	Description
1’b0	ECB
1’b1	CBC

Bit 0 - Algorithm (ALGO)

This bit indicates the algorithm to use (DES or 3DES).

Bit 1	Description
1’b0	DES
1’b1	3DES

21.8.15 TKey registers (DES_KEY)

The Key Registers contain the key.

21.8.16 Channel ID (DES_ID)

The Channel ID register contains the Identifier of this version of the DES channel. The Channel ID for this version of the DES channel is 0x0000_2000.

21.9 AES channel

21.9.1 Overview

This channel computes AES encryption and decryption in ECB, CTR and CBC mode; executing C3 Flow type instruction set.

21.9.2 Instruction set

The AES Channel executes AES [START/APPEND] ENCRYPT and AES [START/APPEND] DECRYPT instructions specified in the C3v3 flow type instruction set v2.1 document [ISv2.1]. Instructions that do not conform to the following bit encodings or to [ISv2.1] are unknown to the AES Channel that will go in error state.

21.9.3 AES instructions

There are 2 different AES instructions:

- AES START
- AES APPEND

The first instruction is used for setting the operation parameters, such as the key and the initialization vector. The second one is used for passing the data to encrypt or decrypt.

21.9.4 AES START instruction

The AES START instruction can be applied with 3 different modes of operation:

- ECB
- CBC
- CTR

21.9.5 ECB

The AES START ECB instruction is 2 words long. This instruction is used to set the key for the following operations. The length of the key is encoded in the first instruction word, while the second word represents the Source Address for the key.

Table 324. AES ECB START instruction bit encoding

W#	Bit Encoding
1	xxxx 01xa 000x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the key

Bit ‘a’ in the above table is used to set the operation to perform:

Table 325. AES ECB Bit ‘a’ encoding

Bit #24 - a	Operation
0	DES
1	3DES

Bits 15 to 0 in the first instruction word (cccc in *Table 322.*) represent the length in Bytes of the key.

21.9.6 CBC

The AES START CBC instruction is 3 words long. This instruction is used to set the key and the initialization vector for the following operations. The length of the key is encoded in the first instruction word, the second word represents the Source Address for the key and the third word represents the Source Address for the Initialization Vector.

Table 326. AES CBC START instruction bit encoding

W#	Bit Encoding
1	xxxx 10xa 001x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the key
3	32 bit Source Address for the IV

Bit ‘a’ in the above table is used to set the operation to perform and has the same encoding as in the ECB instruction (*Table 323.*). Bits 15 to 0 in the first instruction word (cccc in *Table 326*) represent the length in Bytes of the key.

21.9.7 CTR

The AES START CTR instruction is 3 words long. This instruction is used to set the key and the initialization vector for the following operations. The length of the key is encoded in the first instruction word, while the second word represents the Source Address for the key.

Table 327. AES CTR START instruction bit encoding

W#	Bit Encoding
1	xxxx 10xa 010x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the key
3	32 bit Source Address for the IV

Bit ‘a’ in the above table is used to set the operation to perform and has the same encoding as in the ECB instruction (*Table 323.* Bits 15 to 0 in the first instruction word (cccc in *Table 327*) represent the length in Bytes of the key.

21.9.8 AES APPEND instruction

The AES APPEND instruction can be applied with 3 different modes of operation:

- ECB
- CBC
- CTR

21.9.9 ECB

The AES APPEND ECB instruction is 3 words long. This instruction is used for passing the data to process (encrypt or decrypt). The length of the data to process is encoded in the first instruction word, the second word represents the Source Address and the third word represents the Destination Address.

Table 328. AES ECB APPEND instruction bit encoding

W#	Bit Encoding
1	xxxx 10xa 100x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the data
3	32 bit Destination Address for the data

Bit ‘a’ in the above table is used to set the operation to perform.

Bit 24	Description
a	Operation
0	Encryption
1	Decryption

Bits 15 to 0 in the first instruction word (cccc in *Table 328*) represent the length in Bytes of the data to be processed.

21.9.10 CBC

The AES APPEND CBC instruction is 3 words long. This instruction is used for passing the data to process (encrypt or decrypt). The length of the data to process is encoded in the first instruction word, the second word represents the Source Address and the third word represents the Destination Address.

Table 329. AES CBC APPEND instruction bit encoding

W#	Bit Encoding
1	xxxx 10xa 101x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the data
3	32 bit Destination Address for the data

Bit ‘a’ in the above table is used to set the operation to perform and has the same encoding as in the ECB instruction *Table 323.*. Bits 15 to 0 in the first instruction word (cccc in *Table 329*) represent the length in Bytes of the key.

21.9.11 CTR

The AES APPEND CTR instruction is 3 words long. This instruction is used for passing the data to process (encrypt or decrypt). The length of the data to process is encoded in the first instruction word, the second word represents the Source Address and the third word represents the Destination Address.Address.

Table 330. AES CTR APPEND instruction bit encoding

W#	Bit Encoding
1	xxxx 10xa 110x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the data
3	32 bit Destination Address for the data

Bit ‘a’ in the above table is used to set the operation to perform and has the same encoding as in the ECB instruction (*Table 323.*). Bits 15 to 0 in the first instruction word (cccc in *Table 330.*) represent the length in Bytes of the key.

21.10 Register set

21.10.1 Register configuration

The following table summarizes AHB mapped registers of the AES Channel connected to Channel2 of C3.

Table 331. AES registers map

Symbol	Name	Type	Initial value	Address
AES_DATA_INOUT0	Data Input/output register #0	R/W	32’h0	0x000
AES_DATA_INOUT1	Data Input/output register #1	R/W	32’h0	0x004

Table 331. AES registers map (continued)

Symbol	Name	Type	Initial value	Address
AES_DATA_INOUT2	Data Input/output register #2	R/W	32'h0	0x008
AES_DATA_INOUT3	Data Input/output register #3	R/W	32'h0	0x00C
AES_FEEDBACK0	Feedback register #0	R/W	32'h0	0x010
AES_FEEDBACK1	Feedback register #1	R/W	32'h0	0x014
AES_FEEDBACK2	Feedback register #2	R/W	32'h0	0x018
AES_FEEDBACK3	Feedback register #3	R/W	32'h0	0x01C
AES_COUNTER0	Counter register #0	R/W	32'h0	0x020
AES_COUNTER1	Counter register #1	R/W	32'h0	0x024
AES_COUNTER2	Counter register #2	R/W	32'h0	0x028
AES_COUNTER3	Counter register #3	R/W	32'h0	0x02C
AES_CONTROL_STATUS	Control and status register	R/(W)	32'h0	0x040
AES_KEY0	Key register #0	R/W	32'h0	0x050
AES_KEY1	Key register #1	R/W	32'h0	0x054
AES_KEY2	Key register #2	R/W	32'h0	0x058
AES_KEY3	Key register #3	R/W	32'h0	0x05C
AES_KEY4	Key register #4	R/W	32'h0	0x060
AES_KEY5	Key register #5	R/W	32'h0	0x064
AES_KEY6	Key register #6	R/W	32'h0	0x068
AES_KEY7	Key register #7	R/W	32'h0	0x06C
AES_IR	Channel ID register	RO	32'h0	0x3FC

Note: Changing the register values while the AES Channel is executing an instruction may produce wrong results and unexpected behaviors.

21.10.2 Data input/output registers (AES_DATAIN_OUT)

The same address refers to 2 different blocks of registers, depending on the operation (read or write).

The Data Input Registers contain the current data input to the AES Channel (accessed using the write operation).

The Data Output Registers contain the current data output of the AES Channel (accessed using the read operation).

Note: A read operation on these registers just after a write operation will not return the same value previously written.

21.10.3 Feedback registers (AES_FEEDBACK)

The Feedback Registers contain the value that is added to the AES input for implementing the selected mode of operation (it depends on the selected mode).

21.10.4 Counter registers (AES_COUNTER)

The Counter Registers contain the counter used in CTR mode (that will be automatically incremented).

21.10.5 Control and status register (AES_CONTROL_STATUS)

Bit	31	30	29	28	27	26	25	24
Symbol	ED	KEYSZ1	KEYSZ0	MODE2	MODE1	MODE0	KEYRDY	CTXSR1
Initial Value	0	0	0	0	0	0	0	0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Symbol	CTXSR0	res	res	res	res	res	res	res
Initial Value	0	-	-	-	-	-	-	-
Type	R/W	-	-	-	-	-	-	-

Bit	15	14	13	12	11	10	9	8
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit 31 - Encryption/Decryption (ED)

This bit indicates the operation to perform (Encryption or Decryption). For writing this field the bit #4 of the input word has to be set to 1.

Bit 31	Description
1’b1	Encryption
1’b0	Decryption

Bits 30 to 29 - Key Size (KEYSZ)

These 2 bits represent the key length, as in the following internal representation. For writing this field the bit #3 of the input word has to be set to 1.

Bit 30 to 29	Description
2’b00	128 bits
2’b01	192 bits
2’b10	256 bits
2’b11	Not Used

Bits 28 to 26 - Mode of operation (MODE)

These 3 bits represent the mode of operation, as in the following internal representation. For writing this field the bit #2 of the input word has to be set to 1.

Bit 28 to 26	Description
3’b000	ECB
3’b001	CBR
3’b010	CTR
3’b011	Not Used
3’b100	Not Used
3’b101	Not Used
3’b110	Not Used
3’b111	Not Used

Bit 25 - Key ready (KEYRDY)

This bit indicates if the key value is valid or not. For writing this field the bit #1 of the input word has to be set to 1.

Bit 25	Description
1’b1	The Key value is NOT ready
1’b0	The Key value is ready

Bits 24 to 23 - Context Save/Restore (CTX_SR)

These 2 bits represent the operation to do with the context, as in the following internal representation. For writing this field the bit #0 of the input word has to be set to 1.

Bit 30 to 29	Description
2’b00	None
2’b01	Context restore
2’b10	Context Save
2’b11	Not Used

Bits 22 to 0 - Reserved

These bits are reserved and should be written zero.

21.10.6 Key registers (AES_KEY)

The Key Registers contain the key.

21.10.7 Channel ID (AES_ID)

The Channel ID register contains the Identifier of this version of the AES channel. The Channel ID for this version of the AES channel is 0x0000_3000.

21.11 Unified hash with HMAC channel

21.11.1 Overview

Unified HASH channel can compute MD5 and SHA-1 digests of a message, executing C3 Flow type instruction set's HASH [MD5/SHA1] instruction.

It can compute the HMAC of a message with MD5 and SHA-1, executing C3 Flow type instruction set's HMAC [MD5/SHA1] instruction.

It can save and restore the internal context in order to allow the stop and the resume of the computation, executing C3 Flow type instruction set's HASH [CONTEXT] and HMAC [CONTEXT] instructions.

21.11.2 Instruction set

The UHH Channel executes HASH [MD5/SHA1/CONTEXT] and HMAC [MD5/SHA1/CONTEXT] instructions specified in the C3v3 flow type instruction set. Instructions that do not conform to the following bit encodings are unknown to the UHH Channel that will go in error state.

21.11.3 HASH instruction

There are 4 different HASH instructions:

- HASH MD5
- HASH SHA1
- HASH CONTEXT

The first 3 instructions are used for computing the digest of a message and work in the same way. The last one is used for saving and restoring the context.

21.11.4 HASH [MD5/SHA1] instructions

Each HASH [MD5/SHA1] instruction is composed by 3 sub-instructions:

- INIT
- APPEND
- END

21.11.5 INIT

The HASH [MD5/SHA1/SHA2] INIT instruction is 1 word long. This instruction is used to set the Function.

Table 332. HASH INIT Instruction bit encoding

W#	Bit Encoding
1	xxxx 000a a00x xxxx cccc cccc cccc cccc

Bits ‘aa’ in the above table are used to set the algorithm to use:

Table 333. HASH INIT Bit ‘aa’ encoding

Bit 24 to 23	
aa	Algorithm
2’b00	MD5
2’b01	SHA-1
2’b10	Not Used
2’b11	Context (See CONTEXT instruction)

21.11.6 APPEND

The HASH [MD5/SHA1] APPEND instruction is 2 words long. This instruction is used to set the Source Address Register for the message and to start the computation of the digest.

The length of the message is encoded in the first instruction word, while the second word represents the Source Address for the message.

Table 334. HASH APPEND Instruction bit encoding

W#	Bit Encoding
1	xxxx 010a a01x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the message

Bits aa in the above table are used to set the algorithm to use and have the same encoding as in the INIT instruction (*Table 333.*). Bits 15 to 0 in the first instruction word (cccc in *Table 334.*) represent the Count in Bytes of the input message.

21.11.7 END

The HASH [MD5/SHA1] END instruction is 2 words long. This instruction is used to set the Destination Address Register for the message and to end the computation of the digest.

The second word represents the Destination Address for the digest.

Table 335. HASH END Instruction bit encoding

W#	Bit Encoding
1	xxxx 010a a01t xxxx cccc cccc cccc cccc
2	32 bit Destination Address for the message

Bits ‘aa’ in the above table are used to set the algorithm to use and have the same encoding as in the INIT instruction(*Table 333.*).

Bit ‘t’ in the above table is used to truncate the result to 96 bits:

Bit 20	
t	Truncate
1’b0	Full digest
1’b1	Truncated 96 bit digest

21.11.8 HASH CONTEXT instruction

The HASH CONTEXT instruction is composed by 2 sub-instructions:

- SAVE
- RESTORE

21.11.9 SAVE

The HASH CONTEXT SAVE instruction is 2 words long. This instruction is used to set the Destination address Register for the context and to save the full context.

The second word represents the Destination Address for the context.

Table 336. HASH CONTEXT SAVE Instruction bit encoding

W#	Bit Encoding
1	xxxx 0101 10xx xxxx cccc cccc cccc cccc
2	32 bit Destination Address for the context

21.11.10 RESTORE

The HASH CONTEXT RESTORE instruction is 2 words long. This instruction is used to set the Source Address Register for the context and to restore the full context.

The second word represents the Source Address for the context.

Table 337. HASH CONTEXT RESTORE Instruction bit encoding

W#	Bit Encoding
1	xxxx 0101 11xx xxxx cccc cccc cccc cccc
2	32 bit Source Address for the context

21.11.11 HMAC instruction

There are 4 different HMAC instructions:

- HMAC MD5
- HMAC SHA1
- HMAC CONTEXT

The first 3 instructions are used for computing the HMAC of a message and work in the same way. The last one is used for saving and restoring the context.

21.11.12 HMAC [MD5/SHA1] instructions

Each HMAC [MD5/SHA1] instruction is composed by 3 sub-instructions:

- INIT
- APPEND
- END

21.11.13 INIT

The HMAC [MD5/SHA1] INIT instruction is 2 words long. This instruction is used to set the Function and the Source Address Register for the HMAC key.

Table 338. HMAC INIT Instruction bit encoding

W#	Bit Encoding
1	xxxx 011a a00x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the key

Bits ‘aa’ in the above table are used to set the algorithm to use and have the same encoding as in the HASH INIT instruction (*Table 333.*).

Bits 15 to 0 in the first instruction word (cccc in *Table 338.*) represent the length in Bytes of the key.

21.11.14 APPEND

The HMAC [MD5/SHA1] APPEND instruction is 2 words long. This instruction is used to set the Source Address Register for the message and to start the computation of the HMAC.

The length of the message is encoded in the first instruction word, while the second word represents the Source Address for the message.

Table 339. HMAC APPEND Instruction bit encoding

W#	Bit Encoding
1	xxxx 011a a01x xxxx cccc cccc cccc cccc
2	32 bit Source Address for the message

Bits ‘aa’ in the above table are used to set the algorithm to use and have the same encoding as in the INIT instruction(*Table 335.*).

Bits 15 to 0 in the first instruction word (cccc in *Table 339*) represent the Count in Bytes of the input message.

Note: The state of the HASH channel should be saved after an HMAC-APPEND instruction, if a consecutive HMAC-APPEND instruction is to be executed. The consecutive APPEND instruction should be executed after restoring the previously saved state of the channel. If this process is not followed, the HASH channel will not respond to any further instructions as it will become non-operational.

21.11.15 END

The HMAC [MD5/SHA1] END instruction is 3 words long. This instruction is used to set the Source Address Register for the HMAC key and the Destination Address Register for the HMAC and to end the computation of the HMAC.

The second word represents the Source Address for the key, while the third word represents the Destination Address for the HMAC.

Table 340. HMAC END Instruction bit encoding

W#	Bit Encoding
1	xxxx 101a a10tx xxxx cccc cccc cccc cccc
2	32 bit Source Address for the key
3	32 bit Destination Address for the message

Bits ‘aa’ in the above table are used to set the algorithm to use and have the same encoding as in the INIT instruction (*Table 335.*).

Bits 15 to 0 in the first instruction word (cccc in *Table 340.*) represent the length in Bytes of the key.

Bit ‘t’ in the above table is used to truncate the result to 96 bits.

Bit 20	
t	Truncate
1’b0	Full HMAC
1’b1	Truncated 96 bit HMAC

21.11.16 HMAC CONTEXT instruction

The HMAC CONTEXT instruction is composed by 2 sub-instructions:

- SAVE
- RESTORE

21.11.17 SAVE

The HMAC CONTEXT SAVE instruction is 2 words long. This instruction is used to set the Destination Address Register for the context and to save the full context.

The second word represents the Destination Address for the context.

Table 341. HMAC CONTEXT SAVE Instruction bit encoding

W#	Bit Encoding
1	xxxx 0111 10xx xxxx cccc cccc cccc cccc
2	32 bit Destination Address for the context

21.11.18 RESTORE

The HMAC CONTEXT RESTORE instruction is 2 words long. This instruction is used to set the Source Address Register for the context and to restore the full context.

The second word represents the Source Address for the context.

Table 342. HMAC CONTEXT RESTORE instruction bit encoding

W#	Bit Encoding
1	xxxx 0111 11xx xxxx cccc cccc cccc cccc
2	32 bit Source Address for the context

21.11.19 Register configuration

The following table summarizes AHB mapped registers of the UHH Channel connected on channel 3 of C3.

Table 343. UHH channel registers map

Symbol	Name	Type	Initial value	Address
UHH_SR(1)	Core Status Register	R/(W)	32'h0	0x020
UHH_HX0(1)	Hash Status Register #0	R/W	32'h0	0x024
UHH_HX1(1)	Hash Status Register #1	R/W	32'h0	0x028
UHH_HX2(1)	Hash Status Register #2	R/W	32'h0	0x02C
UHH_HX3(1)	Hash Status Register #3	R/W	32'h0	0x030
UHH_HX4(1)	Hash Status Register #4	R/W	32'h0	0x034
UHH_HX5(1)	Hash Status Register #5	R/W	32'h0	0x038
UHH_HX6(1)	Hash Status Register #6	R/W	32'h0	0x03C
UHH_HX7(1)	Hash Status Register #7	R/W	32'h0	0x040
UHH_X0(1)	Hash Working Register #0	R/W	32'h0	0x044
UHH_X1(1)	Hash Working Register #1	R/W	32'h0	0x048
UHH_X2(1)	Hash Working Register #2	R/W	32'h0	0x04C
UHH_X3(1)	Hash Working Register #3	R/W	32'h0	0x050
UHH_X4(1)	Hash Working Register #4	R/W	32'h0	0x054
UHH_X5(1)	Hash Working Register #5	R/W	32'h0	0x058
UHH_X6(1)	Hash Working Register #6	R/W	32'h0	0x05C
UHH_X7(1)	Hash Working Register #7	R/W	32'h0	0x060

Table 343. UHH channel registers map (continued)

Symbol	Name	Type	Initial value	Address
UHH_WX0(1)	Message Scheduler #0	R/W	32'h0	0x064
UHH_WX1(1)	Message Scheduler #1	R/W	32'h0	0x068
UHH_WX2(1)	Message Scheduler #2	R/W	32'h0	0x06C
UHH_WX3(1)	Message Scheduler #3	R/W	32'h0	0x070
UHH_WX4(1)	Message Scheduler #4	R/W	32'h0	0x074
UHH_WX5(1)	Message Scheduler #5	R/W	32'h0	0x078
UHH_WX6(1)	Message Scheduler #6	R/W	32'h0	0x07C
UHH_WX7(1)	Message Scheduler #7	R/W	32'h0	0x080
UHH_WX8(1)	Message Scheduler #8	R/W	32'h0	0x084
UHH_WX9(1)	Message Scheduler #9	R/W	32'h0	0x088
UHH_WX10(1)	Message Scheduler #10	R/W	32'h0	0x08C
UHH_WX11(1)	Message Scheduler #11	R/W	32'h0	0x090
UHH_WX12(1)	Message Scheduler #12	R/W	32'h0	0x094
UHH_WX13(1)	Message Scheduler #13	R/W	32'h0	0x098
UHH_WX14(1)	Message Scheduler #14	R/W	32'h0	0x09C
UHH_WX15(1)	Message Scheduler #15	R/W	32'h0	0x0A0
UHH_UHR(1)	Current Hash Constant	R/W	32'h0	0x0A4
UHH_BCLO(1)	Bit Count Register (LSW)	R/W	32'h0	0x0A8
UHH_BCHI(1)	Bit Count Register (MSW)	R/W	32'h0	0x0AC
UHH_RK0	Digest of the HMAC key #0	R/W	32'h0	0x0B0
UHH_RK1	Digest of the HMAC key #1	R/W	32'h0	0x0B4
UHH_RK2	Digest of the HMAC key #2	R/W	32'h0	0x0B8
UHH_RK3	Digest of the HMAC key #3	R/W	32'h0	0x0BC
UHH_RK4	Digest of the HMAC key #4	R/W	32'h0	0x0C0
UHH_RK5	Digest of the HMAC key #5	R/W	32'h0	0x0C4
UHH_RK6	Digest of the HMAC key #6	R/W	32'h0	0x0C8
UHH_RK7	Digest of the HMAC key #7	R/W	32'h0	0x0CC
UHH_RH0	HMAC working Register #0	R/W	32'h0	0x0D0
UHH_RH1	HMAC working Register #1	R/W	32'h0	0x0D4
UHH_RH2	HMAC working Register #2	R/W	32'h0	0x0D8
UHH_RH3	HMAC working Register #3	R/W	32'h0	0x0DC
UHH_RH4	HMAC working Register #4	R/W	32'h0	0x0E0
UHH_RH5	HMAC working Register #5	R/W	32'h0	0x0E4
UHH_RH6	HMAC working Register #6	R/W	32'h0	0x0E8
UHH_RH7	HMAC working Register #7	R/W	32'h0	0x0EC

Table 343. UHH channel registers map (continued)

Symbol	Name	Type	Initial value	Address
UHH_DATA_IN(1)	CB Status &Control Register	R/W	32'h0	0x0EC
UHH_CB_CONTROL_STATUS(1)	CB Status &Control Register	R/(W)	32'h0	0x200
UHH_CU_CONTROL_STATUS(1)	CU Status and Control Register	R/(W)	32'h8000_0000	0x200
CTAG_IR	Channel ID	RO	32'h0000_4001	0x3FC

1. Marked registers compose the Context (for saving and restoring), in the same order as they are listed in table. The context is composed by 38 words.

21.11.20 Register description

Note: Changing the register values while the UHH Channel is executing an instruction may produce wrong results and unexpected behaviour.

21.11.21 Control and status register (UHH_CU_CONTROL_STATUS)

Bit	31	30	29	28	27	26	25	24
Symbol	CSH	CSL	BERR	DERR	PERR	IERR	AERR	res
Initial Value	0	0	0	0	0	0	0	-
Type	RO	RO	RO	RO	RO	RO	RO	-

Bit	23	22	21	20	19	18	17	16
Symbol	res	res	res	res	res	res	res	RST
Initial Value	-	-	-	-	-	-	-	0
Type	-	-	-	-	-	-	-	R/(W)

Bit	15	14	13	12	11	10	9	8
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit 31 to 30 - Channel Status (CS)

These two bits represent the status of the Channel. The status is reported to the Instruction Dispatcher (which also duplicates this information in its bits SCR_CnS).

Bit 31 CSH	Bit 30 CSL	Description
0	0	Not Present: This Channel does not exist in Hardware.
1	0	Idle: The Channel is idle and instructions can be dispatched to it.
1	1	Busy: The Channel is executing instructions dispatched by an Instruction Dispatcher.
0	1	Error: The Channel is in error state, use Channel registers to know the cause.

When the UHH Channel goes in error state, bits 29 to 24 indicates the cause. The only way to get out from error state is to reset the channel using bit 16 (SCR_RST) or requesting an asynchronous reset of the whole C3.

Bit 29 - Bus Error (BERR)

Every module attached to the HIF receives its own Bus error signal. This signal is set by the HIF if a bus error condition is detected for a Bus transaction initiated by the corresponding module. If the UHH Channel detects a bus error condition it goes in error state and this bit is set.

Bit 29 BERR	Description
1'b1	The HIF reported a bus error condition for a transaction initiated by the UHH Channel.

Bit 28 - Dispatching Protocol Error (DERR)

If the Instruction Dispatcher goes in error state or if it is reset while it is dispatching instruction to the UHH Channel, a dispatching protocol violation could happen. If this is the case the UHH Channel goes in error state and this bit is set. Example: the ID has dispatched the first word of the Hash Append instruction. The UHH Channel is still waiting for the second word. If the ID goes now in error state (that is. because of a bus error), the UHH Channel will never receive that second word. This condition is detected and reported using this bit.

Bit 28 DERR	Description
1'b1	The UHH Channel detected a dispatching protocol violation.
1'b0	(Clearing conditions) This flag is cleared in two ways: resetting the UHH Channel or requesting an asynchronous master reset.

Bit 27 - Couple/Chaining Error (PERR)

The UHH Channel is NOT able to become a Master for Chaining operations. It is NOT able to become simultaneously a Master and a Slave for Coupling operations.

Bit 27 BERR	Description
1’b1	The Channel was requested to become a Chaining-master, or simultaneously both a Couple-Master and a Slave for cascade CCM operations.
1’b0	(Clearing conditions) This flag is cleared in two ways: resetting the UHH Channel or requesting an asynchronous master reset.

Bit 26 - Instruction Decode Error (IERR)

The UHH Channel goes in error state and this bit is set if an invalid instruction is received from the Instruction Dispatcher.

Bit 26 BERR	Description
1’b1	The UHH Channel received an invalid instruction from the Instruction Dispatcher.
1’b0	(Clearing conditions) This flag is cleared in two ways: resetting the UHH Channel or requesting an asynchronous master reset.

Bit 25 - Alignment Error (AERR)

The Source Address and the Destination Address must be 32 bit aligned. Count must be a multiple of 4 Bytes. The UHH Channel goes in error state and this bit is set if any of these condition is not respected.

Bit 25 BERR	Description
1’b1	The UHH Channel received an invalid address or count part.
1’b0	(Clearing conditions) This flag is cleared in two ways: resetting the UHH Channel or requesting an asynchronous master reset.

Bits 23 to 17 - Reserved

These bits are reserved and should be written zero.

Bit 16 - Reset Command (RST)

In Hardware the reset is done synchronously and not all registers are affected by it. The following are the effects of a synchronous reset: bits 29-24, 16, 7-0 of SCR are all cleared, FIFOs are flushed, the UHH Channel goes in Idle state eventually aborting instruction execution and bits 31-30 (CS) of UHH_CU_CONTROL_STATUS are set to Idle.

Bit 26 RST	Description
1’b1	Reset the UHH Channel
1’b0	(Clearing conditions) This bit is cleared as a consequence of the reset, so it is always read zero. Writing zero has no effect.

Bit 15 to 0 - Reserved

These bits are reserved and should be written zero.

21.11.22 Data input register (UHH_DATA_IN)

The Data Input Register contains the current data input word to the UHH Channel.

21.11.23 Control and status register (UHH_CB_CONTROL_STATUS)

The UHH_CB_CONTROL_STATUS bit assignments are given.

Bit	31	30	29	28	27	26	25	24
Symbol	res	NBLW4	NBLW3	NBLW2	NBLW1	NBLW0	STAT3	STAT2
Initial Value	-	0	0	0	0	0	0	0
Type	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Symbol	STAT1	STAT0	INVAL	SHORT	res	res	ALG1	ALG0
Initial Value	0	0	0	0	-	-	0	0
Type	R/W	R/W	R/W	R/W	-	-	R/W	R/W

Bit	15	14	13	12	11	10	9	8
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit	7	6	5	4	3	2	1	0
Symbol	res	res	res	res	res	res	res	res
Initial Value	-	-	-	-	-	-	-	-
Type	-	-	-	-	-	-	-	-

Bit 31 - Reserved

This bit is reserved and should be written zero.

Bits 30 to 26 - Number of Bits for the Last Word (N BLW)

These 5 bits represent the length in bits of the last word of the message.

Bits 25 to 22 - Cryptoblock Internal Status (STAT)

These 4 bits represent the status of the Cryptoblock, as in the following internal representation:

Bit 25 to 22	Status	Description
4’b0000	BLOCK_IDLE	Idle State
4’b0001	HASH_DO_RESET	Init for hash
4’b0010	HASH_REQUEST_DATA	Get data input for hash
4’b0011	HASH_PROCESS_DATA	Process the message
4’b0100	HMAC_DO_RESET_SHORT_KEY	Init for HMAC with short key
4’b0101	HMAC_DO_RESET_LONG_KEY	Init for HMAC with long key
4’b0110	HMAC_REQUEST_IKEY_SHORT	Get the short inner key for HMAC
4’b0111	HMAC_REQUEST_IKEY_LONG	Get the long inner key for HMAC
4’b1000	HMAC_REQUEST_DATA	Get data input for HMAC
4’b1001	HMAC_PROCESS_DATA_SHORT_KEY	Process the message (short key)
4’b1010	HMAC_PROCESS_DATA_LONG_KEY	Process the message (long key)
4’b1011	HMAC_REQUEST_OKEY_SHORT	Get the short outer key for HMAC
4’b1100	HMAC_REQUEST_OKEY_LONG	Get the long outer key for HMAC
4’b1101	CONTEXT_SAVE	Save the context
4’b1110	CONTEXT_RESTORE	Restore the context
4’b1111	Not Used	

Bit 21 - Data Input Valid (INVALID)

This bit indicates if the value in the Data Input Register is valid or not.

Bit 21 INVALID	Description
1’b1	The Data Input Register value is NOT valid
1’b0	The Data Input Register value is valid.

Bit 20 - Short Output (SHORT)

This bit indicates if the output result has to be truncated to 96 bits.

Bit 21 INVALID	Description
1’b1	The result is produced in full length.
1’b0	The result is trancated to 96 bits.

Bits 19 to 18 - Reserved

These bits are reserved and should be written zero.

Bits 17 to 16 - Current Algorithm (ALG)

These 2 bits represent the current algorithm, as in the following representation:

Bit 17 to 16	Algorithm
2'b00	MD5
2'b01	SHA-1
2'b10	Not Used
2'b11	Not Used

Bits 15 to 0 - Reserved

These bits are reserved and should be written zero.

Core Status Register (UHH_SR)

Bit	31	30	29	28	27	26	25	24
Symbol	ALG1	ALG0	res	res	Res	CPHA1	CPHA0	PST2
Initial Value	0	0	-	-	-	0	0	0
Type	R/W	R/W	-	-	-	R/W	R/W	R/W

Bit	23	22	21	20	19	18	17	16
Symbol	PST1	PST0	WCNT3	WCNT2	WCNT1	WCNT0	ST3	ST2
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	R/W	R/W	R/W	R/W	RO	RO

Bit	15	14	13	12	11	10	9	8
Symbol	ST1	ST0	LKEY	PHA1	PHA0	CST	CST	SCNT6
Initial Value	0	0	0	0	0	0	0	0
Type	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W

Bit	7	6	5	4	3	2	1	0
Symbol	SCNT5	SCNT4	SCNT3	SCNT2	SCNT1	SCNT0	LAST	res
Initial Value	0	0	0	0	0	0	0	-
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-

Bits 31 to 30 - Current Algorithm (ALG)

These 2 bits represent the current algorithm, as in the following representation:

Bit 31 to 30	Algorithm
2'b00	MD5
2'b01	SHA-1
2'b10	Not Used
2'b11	Not Used

Bits 29 to 27 - Reserved

These bits are reserved and should be written zero.

Bits 26 to 25 - Current Phase (CPHA)

These bits represent the current phase of the hash algorithm.

Bits 24 to 22 - Padder State (PST)

These bits represent the action in progress in the input padding.

Bit 24 to 22	Algorithm
3'b000	Idle state, no padding
3'b001	Insert the first 1 after the end of the message
3'b010	Insert extra zeros
3'b011	Insert the length of the message
3'b100	Insert extra key
3'b101	Pause the padding
3'b110	Not used
3'b111	Not used

Bits 21 to 18 - Number of Words (WCNT)

These 4 bits represent the number of input words already passed to the hash core.

Bits 17 to 14 - HMAC State (ST)

These bits represent the action in progress in the HMAC procedure.

Bit 17 to 14 ST	Description
4'b0000	Idle state, no work in progress
4'b0001	Get short inner key
4'b0010	Pad short inner key
4'b0011	Get message
4'b0100	Wait for the message digest
4'b0101	Get short outer key

Bit 17 to 14 ST	Description
4’b0110	Pad short outer key
4’b0111	Wait for the HMAC
4’b1000	HMAC value is ready
4’b1001	Get long inner key
4’b1010	Get long outer key
4’b1011	Wait for long inner key preparation
4’b1100	Wait for long outer key preparation
4’b1101	Not used
4’b1110	Not used
4’b1111	Not used

Bit 13 - Long Key (LKEY)

This bit indicates if the HMAC uses a short key or a long one.

Bit 13 LKEY	Description
1’b1	Short HMAC key
1’b0	Long HMAC key

Bits 12 to 11 - Phase (PHA)

These bits represent the phase of the hash algorithm for the next step.

Bits 10 to 9 - Hash Core State (CST)

These bits represent the current phase.

Bit 10 to 9 CST	Description
2’b00	Idle state, no work in progress
2’b01	Compute the digest
2’b10	Update the result
2’b11	Computation ended

Bits 8 to 2 - Step Count (SCNT)

These 7 bits represent the hash round in progress.

Bit 1 - Last Word Asserted (LAST)

This bit indicates if the whole message has been passed to the core.

Bit 0 - Reserved

This bit is reserved and should be written zero.

Hash Status Registers (UHH_HX0 - UHH_HX7)

The Hash Status Registers contain the current partial value of the digest.

Hash Working Registers (UHH_X0 - UHH_X7)

The Hash Working Registers contain a temporary value used for the computation of the digest.

Message Scheduler Registers (UHH_WX0 - UHH_WX15)

The Message Scheduler Registers contain the unrolled message.

Current Hash Constant Register (UHH_UHR)

The Current Hash Register contains the current result of the internal Hash function.

Bit Count Registers (UHH_BCLO - UHH_BCHI)

The Bit Count Registers contain the cumulated length of the processed message.

HMAC Key Digest Registers (UHH_RK0 - UHH_RK7)

The HMAC Key Digest Registers contain the computed digest of the HMAC key.

HMAC Working Registers (UHH_RH0 - UHH_RH7)

The HMAC Working Registers contain a temporary value used for the computation of the HMAC.

21.11.24 Channel ID (UHH_CH_ID)

The Channel ID register contains the Identifier of this version of the UHH Channel. The Channel ID for this version of the UHH channel is 0x0000_4002.

22 HS_USB2.0 host

22.1 Overview

Within its High-Speed (HS) Connection Subsystem, the device provides one USB 2.0 Host with 2 physical ports which are fully compliant with the Universal Serial Bus specification (version 2.0), and offering an interface to the industry-standard AHB bus.

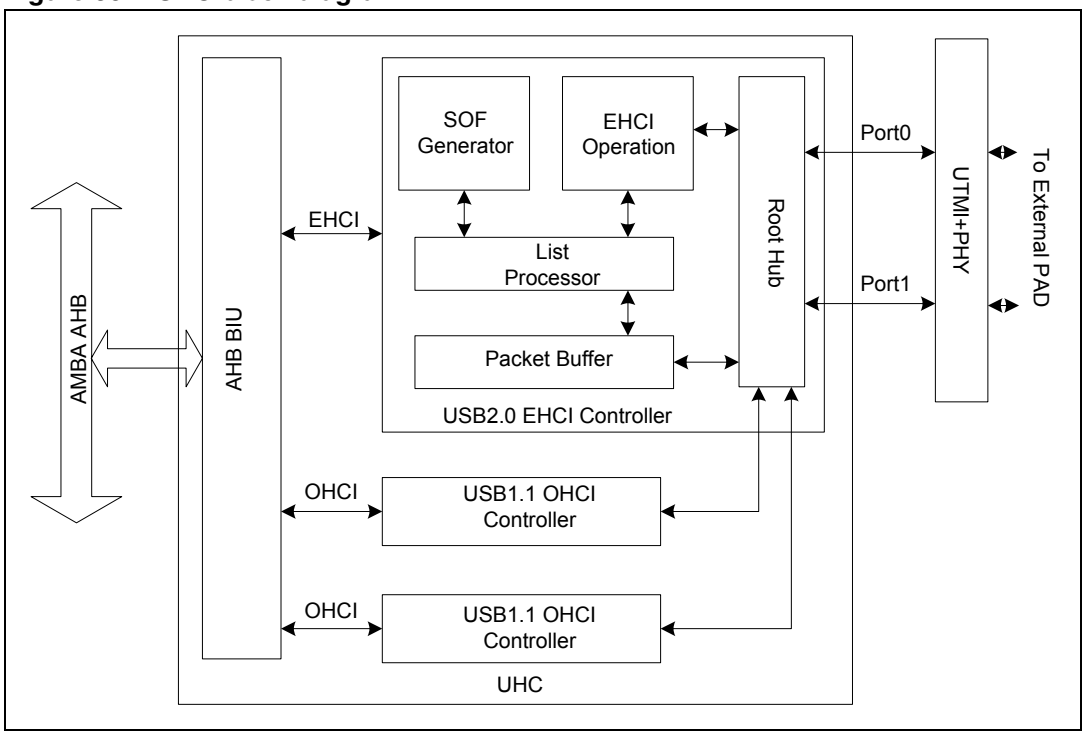
Main features provided by USB 2.0 Host are listed below:

- A PHY interface implementing a USB 2.0 Transceiver Macro-cell Interface plus (UTMI+) fully compliant with UTMI+ specification (revision 1.0), to execute serialization and de-serialization of transmissions over the USB line;
- Either 30 MHz clock for 16 bit interface or 60 MHz for 8 bit interface are supported by the UTMI+ PHY interface;
- A USB 2.0 Host Controller (UHC) which is connected to the AHB bus and generates the commands for the UTMI+ PHY;
- The UHC complies with both the Enhanced Host Controller Interface (EHCI) specification (version 1.0) and the Open Host Controller Interface (OHCI) specification (version 1.0a);
- The UHC supports the 480 Mbps high-speed (HS) for USB 2.0 through an embedded EHCI Host Controller, as well as the 12 Mbps full-speed (FS) and the low-speed (LS) for USB 1.1 through two integrated OHCI Host Controller;
- All clock synchronization is handled within the UHC;
- An AHB slave for each controller (1 EHCI and 2 OHCI), acting as programming interface to access to control and status registers;
- An AHB master for each controller (1 EHCI and 2 OHCI) for data transfer to system memory, supporting 8, 16, and 32 bit wide data transactions on the AHB bus;
- 32 bit AHB bus addressing.

Only one port can be selected to be used for 480 Mbps (HS) communication by the EHCI controller. When one port is used by the EHCI, the other can only be used for full-speed (FS) or low-speed (LS) communication (by the OHCI controller).

22.2 Block diagram

Figure 36. UHC block diagram.



22.3 Main functions description

22.3.1 AHB bus interface unit (BIU)

USB 2.0 Host access to the AHB bus is granted by the AHB Bus Interface Unit (BIU), which consists of a Master module and a Slave module.

The AHB BIU Slave module acts a slave on the AHB and responds to all EHCI/OHCI Operational registers (*Section 22.4.2: Operational registers*) accesses from an AHB master. In particular, this module allows RW access to its operational registers through the AHB bus.

Note: There is only a single AHB slave port in AHB BIU Slave module for both EHCI and OHCI host controller registers access.

The AHB BIU Master module, acting as a master on the AHB, receives requests from the List Processor block *Section 22.4.1: List processor* within the EHCI Host Controller, and transfers data with system memory through the AHB bus. The AHB BIU Master supports 8-, 16-, and 32 bit data transfers, and 32 bit address transfers.

22.3.2 EHCI host controller

An EHCI Host Controller compliant with the EHCI specification (version 1.0) is embedded within the UHC to support the 480 Mbps high-speed (HS) transaction of USB 2.0.HS device connected to one of the two downstream ports.

Major blocks of the EHCI Host Controller are described in *Section 22.4: EHCI host controller blocks*

22.3.3 OHCI host controller

Two OHCI Host Controllers compliant with the OHCI specification (version 1.0a) are also integrated in the UHC to support the 12 Mbps full-speed (FS) and the 1.5 Mbps low-speed (LS) operation of USB 1.1. FS/LS device connected to port0 is managed by OHCI0 and port1 is managed by OHCI1.

Major blocks of the OHCI Host Controller are described in *Section 22.5: OHCI host controller blocks*.

22.4 EHCI host controller blocks

22.4.1 List processor

The List Processor is the main block of the EHCI Host Controller. The List Processor is implemented with multiple state machines to perform the list service flow, which is set up by the Host Controller Driver (HCD) according to the priority set in the Operational registers *Section 22.4.2: Operational registers*

In addition, the List Processor consists of a controller that interfaces with all the other EHCI Host Controller blocks, such as the AHB BIU (Master module), the Packet Buffer, the EHCI Operational registers, the SOF Generators and the Root Hub.

22.4.2 Operational registers

This block exposes the implemented EHCI Capability and Operational registers as defined in the USB EHCI specification. In addition, certain IP-specific extended registers are also implemented, in order to configure features like Packet Buffer depth, break memory transfer, frame length.

The Operational registers block interfaces with the AHB BIU (Slave module), the List Processor and the Root Hub.

22.4.3 Start-Of-Frame (SOF) generator

The SOF Generator block implements the counter which generates the Start-Of-Frame (SOF) packets to supply micro-SOFs for each microframe. The SOF counter runs in the PHY clock domain.

Microframe duration is derived from the EHCI Frame Length Adjustment (FLADJ) register value. This ensures that the Host microframe duration and per-port microframe duration remain the same.

This block interfaces with the List Processor only.

22.4.4 Packet buffer

The Packet Buffer (PBUF) block provides storage and control for IN/OUT data transaction, with a configured size of 1024 bytes (256 x 32 = 1024 bytes).

According to its functionality, the PBUF block interface with both the List Processor and the Root Hub. Specifically, during an OUT transaction, the List Processor fetches data from the system memory and writes them in the PBUF. Besides, during an IN transaction, the data are written to PBUF by the Root Hub *Section 22.4.5: Root hub*.

The Packet Buffer size depends on the system latency and bandwidth allocated to the EHCI Host Controller. For example, in case PBUF size is programmed to 64 bytes, a 1024-bytes IN transfer would get 1024/64 = 16 data transfer on the AHB bus. If the system is not able to ensure EHCI access to AHB bus for these 16 transfers with no breaks, then a buffer overrun occurs. In this case, to avoid buffer overrun or under-run, PBUF size could be set to 1024 bytes.

22.4.5 Root hub

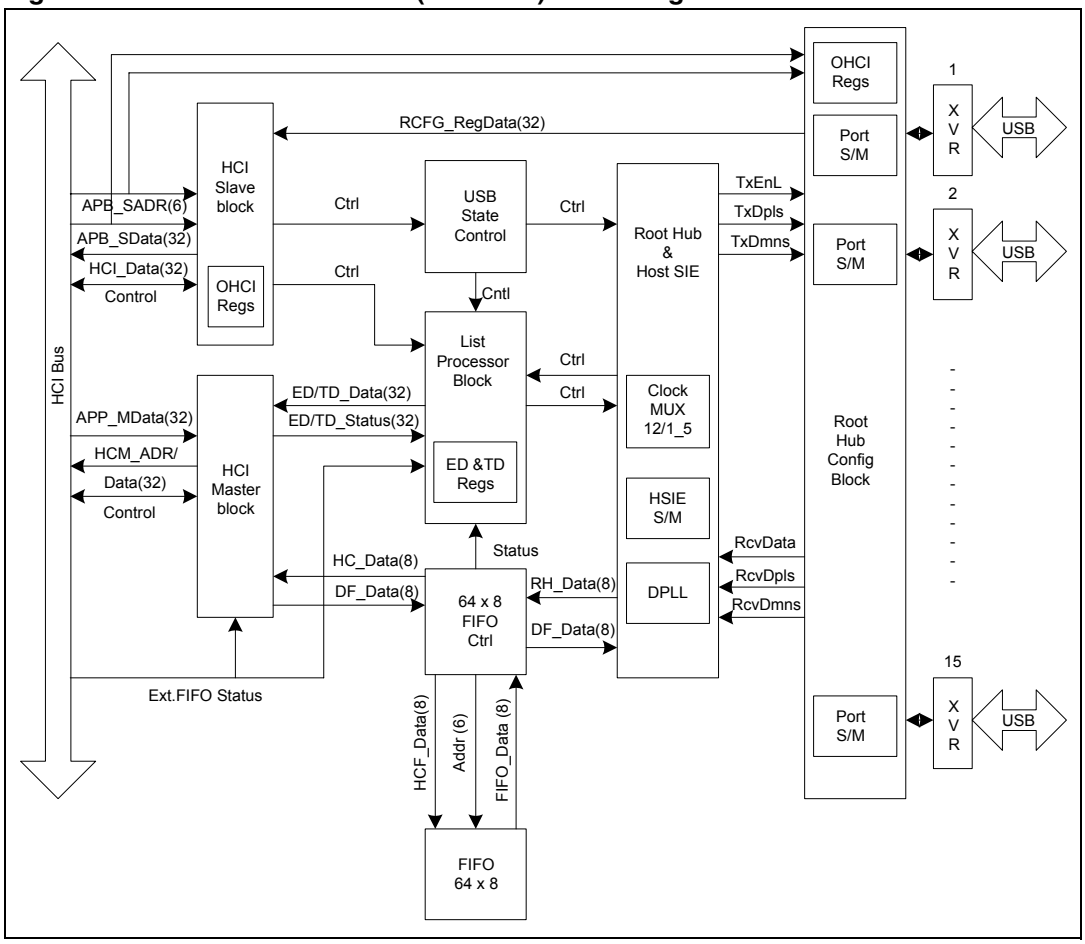
The Root Hub (RH) block interfaces between the List Processor and the USB PHY. It propagates reset and resume signals to downstream ports, and handles port connections and disconnections.

The RH operates both on the local PHY clock (a free-running 30/60 MHz clock) and on the clock source from each physical port (30 MHz with a 16 bit interface).

22.5 OHCI host controller blocks

The USB Open Host Controller is designed to be independent of the Bus Interface Unit (BIU) as in *Figure 37*. The host bus is assumed to be at least 32 bits wide with adequate performance to support the data rate of the particular implementation (100Mbit/sec or higher plus overhead for DMA structures) as well as bounded latency so that the FIFO's can have a reasonable size.

Figure 37. USB Host controller (UHOSTC) block diagram



22.5.1 HCI master block

The HCI master block is the interface between HCI master interface logic block and the HCI bus. It converts all the cycles initiated by different blocks of the list processor through HCI master inteface logic block into HCI bus cycles according to the protocol defined for HCI bus. In addition to that it implements a state machine to read/ write from/to DFIFO. When it is transferring the data returned by endpoint, it reads the data from DFIFO and merges into DWORD and then send it to the application's internal FIFO. Similarly when reading the endpoint data from the system memory, after reading every DWORD from the application's FIFO it splits the DWORD into 4 individual bytes and then sends it to the DFIFO. It also implements byte-alignment logic, that is when a write cycle is initiated by FML block at the odd boundary (not the DWORD boundary), it reads only the lower 2 bit of the address (ties them to 0), so that the application always writes at DWORD boundary, and manipulates the byte-enables accordingly.

22.5.2 HCI slave block

The HCI slave block is the slave on HCI bus. This is basically an interface between OHCI operational register internal to the Host Controller and the application. It updates the registers on writes and provides the register data on reads. All the slave accesses should be DWORD aligned. Therefore, byte enables are not used in slave accesses.

22.5.3 List processor block

The list processor block acts as a main controller of the entire controller. It has multiple state machines to implement List Service Flow, List Priority, USB-States, ED, TD Service, StatusWriteBack, TD Retirement, and so on as per the OHCI specification. Additionally, this block implements a controller which interfaces with HCI_master and hsie, helping them in the data transfer from system memory to USB and USB to system memory.

The following submodules are included:

- USB states
- List service flow
- ED-TD block
- HCI master interface logic
- Data read write logic

22.5.4 RootHub and HSIE blocks

Since implementation varies, most of the functionality of the RootHub is implemented in the Port Configuration Block. This logic is common to any user configuration. The logic in this block acts as a wrapper around HSIE and interface with Host Controller's List Processor, FIFO and OHCI registers. This block also implements the control logic to synchronize the interface between HSIE and port S/M.

This block implements the following submodules:

- Reset_Resume
- DPLL
- HSIE

22.5.5 Digital PLL block (DPLL)

The function of the DPLL Block is to extract the clock and data information from the USB Data received from the different transceiver. The Digital PLL runs on a 48 MHz user-provided clock to extract the clock information from the USB for both Full-Speed and Low-speed data. The two signals D+ and D- of the USB lines are passed through a differential receiver (external to the UHOSTC controller) and a NRZI formatted data is obtained from the output of the differential receivers. The output of the differential receiver is then used by the Digital PLL to extract clock information. The PLL Block also has a SE0 Detect Logic to detect the Single Ended Zero (SE0) in the data stream. The circuit in this module extracts clock from either high-speed data or low-speed data indicated by SIE_Switch HCLK input from SIETx State Machine.

22.5.6 HSIE functionality

The functionality of the Host Serial Interface Engine (HSIE) is to receive and transmit the USB data over D+ and D- lines in accordance with the USB protocol. During the reception of USB data, the D+ and D- signals are passed through the differential receiver (which is external to the UHOSTC controller) to get a single ended bit stream that is passed through the PLL Block to extract the clock and data information. The Clock and data are passed to the SIE Block to identify the Sync Pattern and for NRZI-NRZ conversion. This NRZ data is then passed through the Bit Stripper which strips off the excessive zeros inserted, The data stream is initially passed through the PID Decode and checker to identify different PIDs. Depending upon the type of PID, the HSIE block handles the protocol accordingly.

22.5.7 RootHub port configuration

The port configuration block implements part of the RootHub logic. This block is separated from the main RootHub block to distinguish the logic that varies with design requirements. In short, this block implements part of the OHCI registers that are specific to RootHub and a state machine for every DownStreamPort to control the port functional states.

This block has the following submodules:

- RootHub port registers
- Port S/M
- Port receive
- Port resume
- Port MUX

22.6 Programming model

22.6.1 External pin connections

Table 344. External pin connections

Signal name	Pin	Description
HOST1_DP	K1	Host port 1, positive data line
HOST1_DM	K2	Host port 1, negative data line
HOST1_VBUS	J3	Host port 1, VBUS enable line
HOST1_OVC	H4	Host port 1, overcurrent on VBUS line indicator
HOST2_DP	H1	Host port 2, positive data line
HOST2_DM	H2	Host port 2, negative data line
HOST2_VBUS	H3	Host port 2, VBUS enable line
HOST2_OVC	J4	Host port 2, overcurrent on VBUS line Indicator

22.6.2 UHC interrupts

EHCI block generates one interrupt when following conditions are occurred:

- Interrupt on Async Address
- Host System Error
- Frame List Rollover
- Port Change
- USB Error
- USB Interrupt

But this interrupt is generated only when corresponding bits are enabled in *Table 356: USBINTR register bit assignments* (*Section 22.6.4: Register descriptions of EHCI*). This interrupt is connected with IRQ26 of the CPU (*Table 23: Interrupt sources*, *Section 8.4: Interrupt connection table*).

Similarly, OHCI block generates one interrupt when any of the following conditions occurs:

- OwnershipChange
- RootHubStatusChange
- FrameNumberOverflow
- UnrecoverableError
- ResumeDetected
- StartofFrame
- WritebackDoneHead
- SchedulingOverrun

But this interrupt is generated only when corresponding bits are enabled in *Table 370: HcInterruptEnable register bit assignments*, *Section 22.6.22: Register description of OHCI*. This interrupt is connected with IRQ25 for OHCI1 and IRQ27 for OHCI2 respectively (*Table 23: Interrupt sources*, *Section 8.4: Interrupt connection table*)

22.6.3 Register map

The UHC can be fully configured by programming a set of 32 bit wide registers which can be accessed through the AHB BIU slave module at the base addresses given in *Table 345* (for controller and for the two host ports provided by the device.

Table 345. UHC registers' base address

Host controller	Host port	Base address
EHCI	1 or 2	0xE180_0000(USBBASE)
OHCI	1	0xE190_0000
OHCI	2	0xE210_0000(USBBASE)

Register map for EHCI

The EHCI controller can enable communication through one of the two ports by setting the corresponding PORTSC register in the EHCI Operation Register block.

The registers of the EHCI host controller can be grouped in four different classes:

- Read-only capability registers (listed in*Table 346*), which specify the limits, restrictions and capabilities of the EHCI host controller implementation. These values are used as parameters for the HCD.
- Read/write operational registers (listed in *Table 347*), used by system software to control and monitor the operational state of the EHCI host controller. These registers are implemented in the core power well.

Note: Each operational register is only reset (that is, initialized to its default value) in case of assertion of system hardware reset, or in response to a host controller reset (HCRESET bit set to 1'b1 in USBCMD register).

- Auxiliary power well registers (listed in *Table 348*), which are part of the operational registers but implemented in the auxiliary power well.

Note: Each auxiliary power well register is only reset (that is, initialized to its default value) by hardware in case of initial power-up of the auxiliary power well, or in response to a host controller reset (HCRESET bit set to 1'b1 in USBCMD register).

- PCR registers (listed in *Table 349*), which allow to program configurable registers, such as the packet buffer depth, break memory transfer when the threshold value is reached, the frame length, and UTMI control and status register access.

Table 346. EHCI host controller capability registers summary

Name	Offset(1)	Type	Reset value	Description
HCCAPBASE	USBBASE+ 0x00	RO	32'h01000010	Capability registers base address.
HCSPARAMS	USBBASE+ 0x04	RO	32'h00001116	Structural parameters.
HCCPARAMS	USBBASE+ 0x08	RO	32'h0000A010	Capability parameters.

1. The offset is intended to be with respect to the operational registers base address:USBBASE is fixed to the EHCI base address.

Table 347. EHCI host controller operational registers summary

Name	Offset (1)	Type	Reset value	Description
USBCMD	USBOPBASE+ 0x00	RO	32'h00080900	USB command.
USBSTS	USBOPBASE+ 0x04	RW	32'h00001000	USB status.
USBINTR	USBOPBASE+ 0x08	RW	32'h0	USB interrupt enable.
FRINDEX	USBOPBASE+ 0x0C	RW	32'h0	USB frame index.
CTRLDSSEGMENT	USBOPBASE+ 0x10	RW	32'h0	4G segment selector.
PERIODICLISTBASE	USBOPBASE+ 0x14	RW	32'h0	Periodic frame list base address.
ASYNCLISTADDR	USBOPBASE+ 0x18	RW	32'h0	Asynchronous list address.

1. Offset calculated by reading HCCAPBASE. The offset is kept with respect to the operational registers base address: USBOPBASE = USBBASE + 0x10.

Table 348. EHCI host controller auxiliary power well registers summary

Name	Offset (1)	Type	Reset value	Description
CONFIGFLAG	USBOPBASE+ 0x40	RW	32'h0	Configured flag.
PORTSC1	USBOPBASE+ 0x44	(2)	32'h00002000	Port 1 status and control.
PORTSC2	0x48		32'h00000000	Port 2 status and control.

1. The offset is intended to be with respect to the operational registers base address (USBOPBASE).
2. Depending on port power control (see PP bit description in PORTSC register).

Table 349. EHCI host controller specific registers summary

Name	Offset (1)	Size (bit)	Type	Reset value	Description
INSNREG00	USBOPBASE+0x80	14	RW	14'h0	Programmable microframe base value.
INSNREG01	USBOPBASE+ 0x84	32	RW	32'h00200020	Programmable packet buffer out/in thresholds.
INSNREG02	USBOPBASE+0x88	12	RW	12'h080	Programmable packet buffer depth.
INSNREG03	USBOPBASE+0x8C	1	RW	1'h0	Break memory transfer.
INSNREG04	USBOPBASE+ 0x90	3	RW	3'h0	For debug purposes only.
INSNREG05	USBOPBASE+0x94	32	RW	32'h00001000	UTMI control and status registers.

1. The offset is intended to be with respect to the operational registers base address (USBOPBASE).

Register Map for OHCI

Table 350. Host controller operational registers

Offset	Register name
00	HcRevision
04	HcControl
08	HcCommandStatus
0C	HcInterruptStatus
10	HcInterruptEnable
14	HcInterruptDisable
18	HcHCCA
1C	HcPeriodCurrentED
20	HcControlHeadED
24	HcControlCurrentED
28	HcBulkHeadED
2C	HcBulkCurrentED
30	HcDoneHead
34	HcFmInterval
38	HcFmRemaining
3C	HcFmNumber
40	HcPeriodicStart
44	HcLSTreshold
48	HcRhDescriptorA
4C	HcRhDescriptorB

Table 350. Host controller operational registers (continued)

Offset	Register name
50	HcRhStatus
54	HcRhPortStatus[1]
--	--
54+4*NDP	HcRhPortStatus[NDP]

22.6.4 Register descriptions of EHCI

22.6.5 HCCAPBASE register

The HCCAPBASE is a RO register which contains the base address of the DWord-aligned memory-mapped EHCI host controller capability registers. The HCCAPBASE register bit assignments are given in *Table 351*.

Table 351. HCCAPBASE register bit assignments

Bit	Name	Reset value	Description
[31:16]	HCIVERSION	16'h0100	This field contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.
[15:08]	Reserved	-	Read: undefined
[07:00]	CAPLENGTH	8'h10	This field is used as an offset to add to register base to find the beginning of the Operational Register Space.

22.6.6 HCSPARAMS register

The HCSPARAMS is a RO register stating the structural parameters of the EHCI host controller, such as the number of downstream ports, etc. The HCSPARAMS register bit assignments are given in *Table 352*.

Table 352. HCSPARAMS register bit assignments

Bit	Name	Reset value	Description
[31:24]	Reserved	-	Read: undefined.
[23:20]	DPN	4'h0	Debug port number. This field identifies which of the EHCI host controller ports is the debug port, according to encoding: 4'b0000 = No debug port. 4'b0001 = Port #1. ... =... 4'b1111 = Port #15. Note: The value in DPN field must not be greater than N_PORTS field.

Table 352. HCSPARAMS register bit assignments (continued)

Bit	Name	Reset value	Description
[19:17]	Reserved	-	Read: undefined.
[16]	P_INDICATOR	1‘h0	Port indicators. This bit indicates whether the ports support port indicator control. When this bit is set, each port status control register (PORTSC) of auxiliary power well includes a specific RW field (PIC, port indicator control) for controlling the state of the port indicator.
[15:12]	N_CC	4‘h1	Number of companion controllers. This field indicates the number of companion OHCI host controllers (USB 1.1) associated with the EHCI host controller (USB 2.0). A zero value in this field indicates that there are no companion OHCI host Controllers, whereas a non-zero value indicates that there are as many companion OHCI host controllers (default is 1).
[11:08]	N_PCC	4‘h1	Number of ports per companion controller. This field indicates the number of ports supported per each companion OHCI host controller. It is used to indicate the port routing configuration to system software. The default convention (bit PRR set to 0b0) 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, and so on. Note: The number in this field must be consistent with both N_PORTS and N_CC.
[07]	PRR	1‘h0	Port routing rules. This field indicates the port routing method which drives how all ports are mapped to companion controllers, according to encoding: 1‘b0 = The first N_PCC ports are routed to the lowest numbered function companion OHCI host controller, the next N_PCC port are routed to the next lowest function companion controller, and so on. 1‘b1 = The port routing is explicitly enumerated by the first N_PORTS elements of the HCSP-PORTROUTE array (in the capability registers). In the device, just one single port is present for each controller, so this information is actually irrelevant.
[06:05]	Reserved	-	Read: undefined.

Table 352. HCSPARAMS register bit assignments (continued)

Bit	Name	Reset value	Description
[04]	PPC	1'h1	Port power control. This field indicates whether the EHCI host controller implementation includes port power control. In particular, setting this bit a port power switch is enabled for each port, otherwise (PPC set to 1'b0) each port is hard-wired to power. Note: The value of this field affects the functionality of the port power field (PP) in each port status control registers of auxiliary power well.
[03:00]	N_PORTS	4'h2	Number of physical downstream ports. This field specifies the number of physical downstream ports implemented on this EHCI host controller. The value of this field (ranging from 4'h1 to 4'hF, that is 1 to 15) determines how many port registers are addressable in the auxiliary power well registers memory-space (ranging from offset 0x40 to 0x7C with respect to USBOPBASE address). Note: A zero-value in this field is undefined.

22.6.7 HCCPARAMS register

The HCCPARAMS is a RO register stating the capability parameters of the EHCI host controller, such as scheduling, addressing, etc. The HCCPARAMS register bit assignments are given in *Table 353*.

Table 353. HCCPARAMS register bit assignments

Bit	Name	Reset value	Description
[31:16]	Reserved	-	Read: undefined.
[15:08]	EECP	8'hA0	EHCI extended capabilities pointer. This optional field indicates the existence of a capabilities list. A zero value indicates that no extended capabilities are implemented, whereas a non-zero value indicates the offset in PCI configuration space of the first EHCI extended capability. Note: The pointer value in this field must be 8'h40 or greater in order to maintain the consistency of the PCI header defined for this class of device.

Table 353. HCCPARAMS register bit assignments (continued)

Bit	Name	Reset value	Description
[07:04]	IST	4'h1	Isochronous scheduling threshold. This field indicates, relative to the current position of the executing EHCI host controller, where software can reliably update the isochronous schedule. When bit [7] of this field is 1'b0 (default), the value of the least significant 3 bits indicates the number of micro-frames a EHCI host controller can hold a set of isochronous data structures (one as default or more) before flushing the state. When bit [7] is set to 1'b1, then host software assumes the EHCI host controller may cache an isochronous data structure for an entire frame.
[03]	Reserved	-	Read: undefined.
[02]	ASPC	1'h0	Asynchronous schedule park capability. If this bit is set, then the EHCI host controller supports the park feature for high-speed (HS) queue heads in the asynchronous schedule. The park feature can be disabled or enabled as well as set to a specific level by using the asynchronous schedule park mode enable and asynchronous schedule park mode count fields in the USBCMD register.
[01]	PFLF	1'h0	Programmable frame list flag. This bit states the frame list length, according to encoding: 1'b0 = System software must use a frame list length of 1024 elements with this EHCI host controller. In this case, the frame list size (FLS) in the USBCMD register is a read only field and it should be set to 2'b00. 1'b1 = System software can specify and use a smaller frame list, configured by the frame list size (FLS) field in the USBCMD register. The frame list must always be aligned on a 4K page boundary, in order to ensure that the frame list is always physically contiguous.
[00]	64BAC	1'h0	64 bits addressing capability. This bit documents the addressing range capability of this implementation, according to encoding: 1'b0 = Data structures using 32 bit address memory pointers. 1'b1 = Data structures using 64 bit address memory pointers.

22.6.8 USBCMD register

The USBCMD is a RW register which indicates the command to be executed by the serial bus EHCI host controller.

Note: Writing this register causes a command to be executed. The USBCMD register bit assignments are given in Table 354.

Table 354. USBCMD register bit assignments

Bit	Name	Reset value	Description
[31:24]	Reserved	-	Read: undefined. Write: should be zero.
[23:16]	ITC	8'h08	Interrupt threshold control. This field is used by system software to select the maximum rate at which the EHCI host controller will issue interrupts, according to encoding (any value other than those defined above yields undefined results): 8'h00 = Reserved. 8'h01 = 1 micro-frame. 8'h02 = 2 micro-frames. 8'h04 = 4 micro-frames. 8'h08 = 8 micro-frames (default, equal to 1 ms). 8'h10 = 16 micro-frames (2 ms). 8'h20 = 32 micro-frames (4 ms). 8'h40 = 64 micro-frames (8 ms). Note: Software modifications to this field while HH bit in USBSTS register is equal to 0 results in undefined behavior.
[15:12]	Reserved	-	Read: undefined. Write: should be zero.
[11]	ASPME	1'h1	Asynchronous schedule park mode enable. This bit is used by software to enable (bit set to 1'b1) or disable (1'b0) the Park mode. If the asynchronous park capability bit in the HCCPARAMS register (ASPC, bit [2]) is set, then this bit defaults to 1'b1 and it is RW. In contrast, this bit must be set to 1'b0 and it is RO.
[10]	Reserved	-	Read: undefined. Write: should be zero.
[09:08]	ASPMC	2'h1	Asynchronous schedule park mode count. This 2 bit field contains a count of the number of successive transactions the EHCI host controller is allowed to execute from a high-speed (HS) queue head on the asynchronous schedule before continuing the traversal of the asynchronous schedule. Valid values are 2'h1 (2'b01) to 2'h3 (2'b11) only. If the asynchronous park capability bit in the HCCPARAMS register (ASPC, bit [2]) is set, then this field defaults to 2'b11 and it is RW. In contrast, it defaults to 2'b00 and it is RO. Note: Software must not write a zero value (2'b00) to this field when park mode enable is set as it will result in undefined behavior.

Table 354. USBCMD register bit assignments (continued)

Bit	Name	Reset value	Description
[07]	LHCR	1‘h0	Light host controller reset. This bit allows the driver to reset the EHCI host controller without affecting the state of the ports or the relationship to the companion OHCI host controllers. For example, the PORSTC registers should not be reset to their default values and the CF bit (in CONFIGFLAG register setting should not go to zero (retaining port ownership relationships). If this bit is set to 1‘b0, the light host controller reset has been completed and it is safe for host software to re-initialize the EHCI host controller. Besides, if this bit is set to 1‘b1, the light host controller reset has not yet completed. Note: If light host controller reset is not implemented, reading this bit will always return a zero value (1‘b0).
[06]	IAAD	1‘h0	Interrupt on async advance doorbell. This bit is used as a doorbell by software to tell the EHCI host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1‘b1 to this bit to ring the doorbell. When the EHCI host controller has evicted all appropriate cached schedule state, it sets the interrupt on async advance status bit (IAA, bit [5]) in the USBSTS register. If the Interrupt on async advance enable bit in the USBINTR, is set, then the EHCI host controller will assert an interrupt at the next interrupt threshold. Note: The EHCI host controller clears the IAAD bit after it has set the IAA status bit in the USBSTS register. Note: In order to avoid undefined results, software should not set this bit when the asynchronous schedule is disabled.
[05]	ASE	1‘h0	Asynchronous schedule enable. This bit controls whether the EHCI host controller skips processing the asynchronous schedule, according to encoding: 1‘b0 = Don’t process the asynchronous schedule. 1‘b1 = Use the ASYNCLISTADDR register to access the asynchronous schedule.
[04]	PSE	1‘h0	Periodic schedule enable. This bit controls whether the EHCI host controller skips processing the periodic schedule, according to encoding: 1‘b0 = Don’t process the periodic schedule. 1‘b1 = Use the PERIODICLISTBASE register to access the periodic schedule.

Table 354. USBCMD register bit assignments (continued)

Bit	Name	Reset value	Description
[03:02]	FLS	2‘h0	Frame list size. This 2 bit field specifies the size of the frame list, according to encoding: 2‘b00 = 102 elements (4096 bytes). 2‘b01 = 512 elements (2048 bytes). 2‘b10 = 256 elements (1024 bytes) - for resource-constrained environments 2‘b11 = Reserved. The frame list size set by this field controls which bits in the FRINDEX register should be used for the frame list current index.
[01]	HCRESET	1‘h0	Host controller reset. This control bit is used by software to reset the EHCI host controller. When software set this bit, the EHCI host controller resets its internal pipelines, timers, counters, state machines, etc. to their initial values. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. PCI configuration registers are not affected by this reset. All operational registers, including port registers and port state machines are set to their initial values. Note: Port ownership reverts to the companion OHCI host controller(s), with the side effects. This bit is cleared by the EHCI host controller when the reset process is complete. Note: Software cannot terminate the reset process early by writing a 1‘b0 to this field. Software must reinitialize the EHCI host controller in order to return to an operational state. Note: Software setting this bit while HCHalted bit in USBSTS register is equal to 1‘b0 results in undefined behavior (because attempting to reset an actively running EHCI host controller).
[00]	RS	1‘h0	Run / stop. Setting this bit, the EHCI host controller proceeds with execution of the schedule, and it continues execution as long as RS is set. Clearing this bit, the EHCI host controller completes the current and any actively pipelined transactions on the USB and then halts. The HCHalted bit in the USBSTS register reflects this status. Note: The EHCI host controller must halt within 16 micro-frames after software clears the RS bit. Note: In order to avoid undefined results, software must not set the RS bit until the EHCI host controller is in the halted state (i.e., HCHalted in the USBSTS register is set to 1‘b1).

22.6.9 USBSTS register

The USBSTS is a RW register which indicates pending interrupts and various states of the EHCI host controller. The USBSTS register bit assignments are given in *Table 355*.

Note: 1 The status resulting from a transaction on the serial bus is not indicated in this register.

2 Software clears a bit in this register by writing a 1‘b1 to it.

Table 355. USBSTS register bit assignments

Bit	Name	Reset value	Description
[31:16]	Reserved	-	Read: undefined. Write: should be zero.
[15]	ASS	1‘h0	Asynchronous schedule status. The bit reports the current real status of the asynchronous schedule, according to encoding: 1‘b0 = Disabled. 1‘b1 = Enabled. The EHCI 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. When this bit and the asynchronous schedule enable bit are the same value, the asynchronous schedule is either enabled or disabled.
[14]	PSS	1‘h0	Periodic schedule status. The bit reports the current real status of the periodic schedule, according to encoding: 1‘b0 = Disabled. 1‘b1 = Enabled. The EHCI 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. When this bit and the periodic schedule enable bit are the same value, the periodic schedule is either enabled or disabled.
[13]	R	1‘h0	Reclamation. This is a read-only status bit, which is used to detect an empty asynchronous schedule.
[12]	HH	1‘h1	HCHalted. This bit is set by the EHCI host controller after it has stopped executing as a result of the RS bit (in USBCMD register being cleared, either by software or by the EHCI host controller hardware (e.g. internal error). Besides, this bit is set to 1‘b0 whenever the RS bit is set to 1‘b1.
[11:06]	Reserved	-	Read: undefined. Write: should be zero.

Table 355. USBSTS register bit assignments (continued)

Bit	Name	Reset value	Description
[05]	IAA	1‘h0	Interrupt on async advance. This status bit indicates the assertion of that interrupt source. System software can force the EHCI host controller to issue an interrupt the next time the EHCI host controller advances the asynchronous schedule by setting the interrupt on async advance doorbell bit (IAAD) in the USBCMD register.
[04]	HSE	1‘h0	Host system error. This bit is set by the EHCI host controller when a serious error occurs during a host system access involving the EHCI host controller module. When this error occurs, the EHCI host controller clears the RS bit in the USBCMD register to prevent further execution of the scheduled TDs(1).
[03]	FLR	1‘h0	Frame list rollover. This bit is set by the EHCI host controller when the frame list index (see FRINDEX register in rolls over from its maximum value to 0. 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, FLS, field of the USBCMD register) is 1024 (FLS is 2‘b00), the frame index register rolls over every time FRINDEX[13] toggles. Similarly, if the size is 512 (FLS is 2‘b01), the EHCI host controller sets the FLR bit every time FRINDEX[12] toggles.
[02]	PCD	1‘h0	Port change detect. This bit is set by the EHCI host controller when any port for which the port owner bit is set to 1‘b0 (bit PO in port status controls register) has a change bit transition from a 1‘b0 to a 1‘b1 or a force port resume bit transition from a ‘b0 to a ‘b1 as a result of a J-K transition detected on a suspended port. This bit will also be set as a result of the connect status change being set to 1‘b1 after system software has relinquished ownership of a connected port by writing a 1‘b1 to a port's port owner (PO) bit.

Table 355. USBSTS register bit assignments (continued)

Bit	Name	Reset value	Description
[01]	USBERRINT	1‘h0	USB error interrupt. This bit is set by the EHCI host controller when completion of a USB transaction results in an error condition (e.g., error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both this bit and USBINT bit are set.
[00]	USBINT	1‘h0	USB interrupt. This bit is set by the EHCI host controller on the completion of a USB transaction, which results in the retirement of a TD that had its IOC bit set. The EHCI host controller also sets this bit when a short packet is detected (actual number of bytes received was less than the expected number of bytes).

1. See EHCI documentation for the detailed definitions of the data structures TD, IOC, etc.

22.6.10 USBINTR register

The USBINTR is a RW register which enables to report corresponding interrupts to the software. It means that when an enabling bit of this register is set and the corresponding interrupt is active, an interrupt is generated and sent to the EHCI host controller, that issue the interrupt request (IRQ26 for EHCI1). The USBINTR register bit assignments are given in *Table 356*.

Note: Interrupt sources that are disabled in this register (enabling bit set to 1‘b0) still appear in USBSTS register allowing the software to poll for events.

Table 356. USBINTR register bit assignments

Bit	Name	Reset value	Description
[31:06]	Reserved	-	Read: undefined. Write: should be zero.
[05]	Interrupt on Async Advance Enable	1‘h0	When both this bit and the Interrupt on async advance (IAA) bit in the USBSTS register are set, the EHCI host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the IAA bit.
[04]	Host System Error Enable	1‘h0	When both this bit and the host system error (HSE) bit in the USBSTS register are set, the EHCI host controller will issue an interrupt. The interrupt is acknowledged by software clearing the HSE bit.
[03]	Frame List Rollover Enable	1‘h0	When both this bit and the frame list rollover (FLR) bit in the USBSTS register are set, the EHCI host controller will issue an interrupt. The interrupt is acknowledged by software clearing the FLR bit.

Table 356. USBINTR register bit assignments (continued)

Bit	Name	Reset value	Description
[02]	Port Change Interrupt Enable	1'h0	When both this bit and the port change detect (PGD) bit in the USBSTS register are set, the EHCI host controller will issue an interrupt. The interrupt is acknowledged by software clearing the PGD bit.
[01]	USB Error Interrupt Enable	1'h0	When both this bit and the USBERRINT bit in the USBSTS register are set, the EHCI host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit.
[00]	USB Interrupt Enable	1'h0	When both this bit and the USBINT bit in the USBSTS register are set, the EHCI host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit.

22.6.11 FRINDEX register

The FRINDEX (frame index) is a RW register used by the EHCI host controller to index into the periodic frame list. The register updates every 125 microseconds, that is each micro-frame. The FRINDEX register bit assignments are given in *Table 357*.

Note: 1 The FRINDEX register must be written as a DWord. Byte writes produce undefined results.

2 The FRINDEX register cannot be written unless the EHCI host controller is in the halted state as indicated by the HCHalted bit (in USBSTS register). A write to this register while the RS bit (in USBCMD register) is set to 0b1 produces undefined results. Writes to this register also affect the SOF value.

Table 357. FRINDEX register bit assignments

Bit	Name	Reset value	Description
[31:14]	Reserved	-	Read: undefined. Write: should be zero.
[13:00]	Frame Index	14'h0000	See *Table 358*.

The value of the frame index field increments at the end of each time frame (e.g. micro-frame). In particular, bits [N:3] of this field are used as frame list current index to select a particular entry in the periodic frame list during periodic schedule execution.

Note: This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index.

The actual number of bits (that is, N) used for the frame list current index depends on the size of the frame list as set by system software in the FLS field in the USBCMD register, according to encoding:

Table 358. USBCMD register encoding

FLS field value	Number of elements	N
2‘b00	1024	12
2‘b01	512	11
2‘b10	256	10
2‘b11	Reserved	-

The SOF frame number value for the bus SOF token is derived or alternatively managed from this register. The value of FRINDEX must be 125 µsec (1 micro-frame) ahead of the SOF token value. The SOF value may be implemented as an 11 bit shadow register. For this discussion, this shadow register is 11 bits and is named SOFV. Then, SOFV updates every 8 micro-frames (1 millisecond).

An example implementation to achieve this behavior is to increment SOFV each time the FRINDEX[2:0] increments from a 0 to a 1.

Software must use the value of FRINDEX to derive the current micro-frame number, both for high-speed isochronous scheduling purposes and to provide the get micro-frame number function required for client drivers.

Therefore, the value of FRINDEX and the value of SOFV must be kept consistent if either chip is reset or software writes to FRINDEX. Writes to FRINDEX must also write-through FRINDEX[13:3] to SOFV[10:0]. In order to keep the update as simple as possible, software should never write a FRINDEX value where the three least significant bits are 3‘b111 or 3‘b000.

22.6.12 CTRLDSSEGMENT register

The CTRLDSSEGMENT (control data structure segment) is a RW register which corresponds to the most significant address bits [63:32] for all EHCI data structures.

If the 64 bit addressing capability (64BAC) field in HCCPARAMS register is set to 1‘b0, then this register is not used. Software cannot write to it and a read from this register will return zeros.

If the 64BAC field in HCCPARAMS register is set to 1‘b1, then this register is used with the link pointers to construct 64 bit addresses to EHCI control data structures. This register is concatenated with the link pointer from either the PERIODICLISTBASE, ASYNCLISTADDR or any control data structure link field to construct a 64 bit address.

This register allows the Host software to locate all control data structures within the same 4 GByte memory segment.

22.6.13 PERIODICLISTBASE register

The PERIODICLISTBASE (periodic frame list base address) is a RW register which contains the beginning address of the periodic frame list in the system memory. If the EHCI host controller is in 64 bit mode (as indicated by a 1‘b1 in the 64BAC field in the HCCSPARAMS register, then the most significant 32 bits of every control data structure address comes from the CTRLDSSEGMENT register (see above). The PERIODICLISTBASE register bit assignments are given in *Table 359*.

The contents of this register are combined with the FRINDEX register to enable the EHCI host controller to step through the periodic frame list in sequence.

Note: 1 System software loads this register prior to starting the schedule execution by the EHCI host controller.

2 The memory structure referenced by this physical memory pointer is assumed to be 4 kbytes aligned.

Table 359. PERIODICLISTBASE register bit assignments

Bit	Name	Reset value	Description
[31:12]	Base Address	20'h0	These bits correspond to memory address signals [31:12], respectively.
[11:00]	Reserved	-	Read: undefined. Write: should be zero.

22.6.14 SYNCLISTADDR register

The ASYNCLISTADDR (current asynchronous list address) is a RW register which contains the address of the next asynchronous queue head to be executed. The ASYNCLISTADDR register bit assignments are given in *Table 360*.

Note: 1 If the Host Controller is in 64 bit mode (as indicated by a 1'b1 in the 64BAC field in the HCCSPARAMS register, then the most significant 32 bits of every control data structure address comes from the CTRLDSSEGMENT register.

2 Bits [4:0] of this register cannot be modified by system software and will always return a zero when read.

3 The memory structure referenced by this physical memory pointer is assumed to be 32 bytes (cache line) aligned.

Table 360. ASYNCLISTADDR register bit assignments

Bit	Name	Reset value	Description
[31:05]	LPL	27'h0	Link pointer low. These bits correspond to memory address signals [31:5], respectively. This field may only reference a queue head (QH).
[04:00]	Reserved	-	Read: undefined. Write: should be zero.

22.6.15 CONFIGFLAG register

The CONFIGFLAG is a RW register which is properly set by the host software as the last action in EHCI host controller initialization (after initial power-on or hardware/software reset). In particular, this register allows to control the global port routing policy of the EHCI host controller.

The CONFIGFLAG register bit assignments are given in *Table 361*.

Table 361. CONFIGFLAG register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read: undefined. Write: should be zero.
[00]	CF	1'h0	Configure flag. This bit controls the global port routing policy of the EHCI host controller, according to encoding: 1'b0 = All ports are routed to the appropriate companion OHCI host controller. 1'b1 = All ports are routed to the EHCI host controller.

22.6.16 PORTSC registers

Each EHCI host controller must implement one or more port status and control (PORTSC) registers. The actual number of PORTSC registers implemented by the EHCI host controller is reported in the N_PORTS field of the HCSPARAMS register. For SPEAr300 implementation this value is 4'h2 then two ports are available.

The bit assignments of each PORTSCi (i = 1, 2... N_PORTS) register are given in *Table 362*.

Table 362. PORTSC register bit assignments

Bit	Name	Reset value	Description
[31:23]	Reserved	-	Read: undefined. Write: should be zero.
[22]	WKOC_E	1'h0	Wake on over-current enable. Setting this bit enables the port to be sensitive to over-current conditions as wake-up events.
[21]	WKDSCNNT_E	1'h0	Wake on disconnect enable. Setting this bit enables the port to be sensitive to device disconnects as wake-up events.
[20]	WKCNNT_E	1'h0	Wake on connect enable. Setting this bit enables the port to be sensitive to device connects as wake-up events. Note: The three fields above are all zero if port power (PP bit in this register) is zero.
[19:16]	PTC	4'h0	Port test control. When this 4 bit field is zero (4'b0), the port is not operating in a test mode. In contrast, a non-zero value indicates that it is operating in test mode and the specific test mode is indicated by the specific value, according to encoding: 4'b0000 = Disabled. 4'b0001 = Test J_STATE. 4'b0010 = Test K_STATE. 4'b0011 = Test SE0_NAK. 4'b0100 = Test Packet. 4'b0101 = Test FORCE_ENABLE. 4'b0001 to 4'b1111Reserved.

Table 362. PORTSC register bit assignments (continued)

Bit	Name	Reset value	Description
[15:14]	PIC	2'h0	Port indicator control. Writing to these 2 bit field has no effect if the P_INDICATOR bit in the HCSPARAMS register is cleared. If P_INDICATOR bit is set to 1'b1, then the PIC encoding is: 2'b00 = Port indicators are off. 2'b01 = Amber. 2'b10 = Green. 2'b11 = Undefined. Note: This field is zero if port power (PP bit in this register) is zero.
[13]	PO	1'h1	Port owner. This bit unconditionally goes to 1'b0 when the CF bit in the CONFIGFLAG register makes a 1'b0 to 1'b1 transition. In contrast, this bit unconditionally goes to 1'b1 whenever the CF bit is 1'b0. System software uses this PO field to release ownership of the port to a selected host controller (in the event that the attached device is not an high-speed device). Software writes a 1'b1 to this bit when the attached device is not an HS device, meaning that a companion OHCI host controller owns and controls the relevant port.
[12]	PP	1'h0	Port Power. The function of this bit depends on the value of the port power control (PPC) field in the HCSPARAMS register, according to encoding: 1'b0 1'b1 = EHCI host controller does not have port power control switches, and each port is hard-wired to power. This field is RO. 1'b1 1'bx = EHCI host controller has port power control switches, and actual PP value represents the current setting of the switch: 1'b0 Off 1'b1 On When power is not available on a port (i.e. PP equals to 1 'b0), the port is non-functional and will not report attaches, detaches, etc. Note: When an over-current condition is detected on a powered port and PPC is set, the PP bit in each affected port may be transitioned by the EHCI host controller from 1'b1 to 1'b0 (then removing power from the port).

Table 362. PORTSC register bit assignments (continued)

Bit	Name	Reset value	Description
[11:10]	LS	2'h0	Line status. This 2 bit field reflects the current logical levels of the D+ (bit [11]) and D- (bit [10]) signal lines, according to encoding: 2'b00 SE0 = Not low-speed device, perform EHCI reset. 2'b01 J-state = Not low-speed device, perform EHCI reset. 2'b10 K-state = Low-speed device, release ownership of port. 2'b11 Undefined = Not low-speed device, perform EHCI reset. These bits are used for detection of low-speed (LS) USB devices prior to the port reset and enable sequence. Note: This field is valid only when the port enable bit is 1'b0 and the current connect status bit is set to 1'b1. Note: The value of this field is undefined if port power (PP bit in this register) is zero.
[09]	Reserved	-	Read: undefined. Write: should be zero.
[08]	PR	1'h0	Port Reset. This bit states whether the port is in reset, according to encoding: 1'b0 = Port is not in reset. 1'b1 = Port is in reset. When software writes a 1'b1 to this bit (from a 1'b0), the bus reset sequence as defined in the *Universal Serial Bus Specification Revision 2.0* is started. Software must keep this bit at a 1'b1 long enough to ensure the reset sequence completes. Note: When software writes this PR bit to a 1'b1, it must also write a 1'b0 to the port enable bit. Software writes a 1'b0 to this bit to terminate the bus reset sequence. Note: When software writes a 1'b0 to this bit there may be a delay before the bit status changes to a 1'b0. The bit status will not read as a 1'b0 until after the reset has completed. If the port is in high-speed (HS) mode after reset is complete, the EHCI host controller will automatically enable this port (e.g. set the port enable bit to a 1'b1). A EHCI host controller must terminate the reset and stabilize the state of the port within 2 milliseconds of software transitioning this bit from a1'b1 to a 1'b0. Note: The HCHalted bit in the USBSTS register should be a zero before software attempts to use the PR bit. The EHCI host controller may hold PR asserted to a one when the HCHalted bit is a one. Note: This field is zero if port power (PP bit in this register) is zero.

Table 362. PORTSC register bit assignments (continued)

Bit	Name	Reset value	Description
[07]	S	1'h0	Suspend. This bit states whether the port is in suspend, according to encoding: 1'b0 = Port is not in suspend state. 1'b1 = Port is in suspend state. This S bit together with the port enabled bit (PEN) in this register define the port states as follows: 1'b0 1'bx = Disabled. 1'b1 1'b0 = Enabled. 1'b1 1'b1 = Suspend. 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'b1. 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. A write of 1'b0 to this bit is ignored by the EHCI Host Controller. The EHCI host controller will unconditionally set this bit to a zero when: software sets the force port resume (FPR) bit to 1'b0 (from a 1'b1) software sets the port reset (PR) bit to 1'b1 (from a 1'b0). Note: If host software sets this bit to 1'b1 when the port is not enabled (i.e. PEN bit is 1'b0) the results are undefined. Note: This field is zero if port power (PP bit in this register) is zero.

Table 362. PORTSC register bit assignments (continued)

Bit	Name	Reset value	Description
[06]	FPR	1'h0	Force port resume. This bit states whether the port is in suspend, according to encoding: 1'b0 = No resume (K-state) detected/driven on port. 1'b1 = Resume detected/driven on port. The functionality defined for manipulating this bit depends on the value of the suspend bit (see above). For example, if the port is not suspended (S is 1'b0 and PEN is 1'b1) and software transitions this bit to 1'b1, then the effects on the bus are undefined. The EHCI host controller sets the FPR bit to 1'b1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit changes to 1'b1 because a J-to-K transition is detected, the port change detect (PCD) bit in the USBSTS register is also set to 1'b1. Software sets this bit to 1'b1 to drive resume signaling. In this case, the EHCI host controller must not set the port change detect bit. The resume signaling (full-speed 'K') is driven on the port as long as this bit remains a 1'b1. Software must appropriately time the resume and set this bit to a zero when the appropriate amount of time has elapsed. Writing a zero (from one) causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit will remain a one until the port has switched to the high-speed idle. The EHCI host controller must complete this transition within 2 milliseconds of software setting this bit to 1'b0. Note: This field is zero if port power (PP bit in this register) is zero.
[05]	OcC	1'h0	Over-current change. This bit is set to 1'b1 when there is a change in the over-current active (OcA) bit in this register. Software clears this bit by writing a one to this bit position.
[04]	OcA	1'h0	Over-current active. This bit states whether the port has a over-current condition, according to encoding: 1'b0 = This port does not have an over-current condition. 1'b1 = This port currently has an over-current condition. Note: This bit will automatically transition from a 1'b1 to a 1'b0 when the over-current condition is removed.
[03]	PEDC	1'h0	Port enable/disable change. This bit is set to 1'b1 when port enabled/disabled status (reflected by the PEN bit in this register) has changed. Software clears this bit by writing a one to this bit position.

Table 362. PORTSC register bit assignments (continued)

Bit	Name	Reset value	Description
[02]	PEN	1'h0	Port enabled/disabled. This bit states whether the port is enabled, according to encoding: 1'b0 = Disabled. 1'b1 = Enabled. Ports can only be enabled by the EHCI host controller as a part of the reset and enable. Software cannot enable a port by writing a 1'b1 to this field. The EHCI host controller will only set this bit to 1'b1 when the reset sequence determines that the attached device is an high-speed (HS) device. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. When the port is disabled (1'b0), downstream propagation of data is blocked on this port, except for reset. Note: 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 EHCI host controller and bus events. This field is zero if port power (PP bit in this register) is zero.
[01]	CSC	1'h0	Connect status change. This bit is set to indicates that a change has occurred in the port's current connect status (CCS bit in this register). The EHCI host 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 already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to this bit position. This field is zero if port power (PP bit in this register) is zero.
[00]	CCS	1'h0	Current connect status. This bit reflects the current state of the port, according to encoding, and may not correspond directly to the event that caused the CSC bit to be set: 1'b0 = No device is present on port. 1'b1 = Device is present on port. This field is zero if port power (PP bit in this register) is zero.

22.6.17 INSNREG00 register

The INSNREG00 is a RW 14 bit register which allows to reduces the microframe length in simulation.

22.6.18 INSNREG01 register

The INSNREG01 is a RW register which allows to break memory transactions (in both out and in direction) into chunks once a threshold value (in bytes) is reached. Enabling of break

memory feature is driven by the INSNREG03 register. The INSNREG01 register bit assignments are given in *Table 363*.

Table 363. INSNREG01 register bit assignments

Bit	Name	Reset value	Description
[31:16]	OUT	16'h0020	Out transactions threshold (in bytes).
[15:00]	IN	16'h0020	In transactions threshold (in bytes).

22.6.19 INSNREG02 register

The INSNREG02 is a RW 12 bit register which allows to configure the packet buffer depth. As stated by the reset value (12'h080), the buffer depth is 128 x 32 by default.

22.6.20 INSNREG03 register

The INSNREG03 is a RW 1 bit register used in conjunction with INSNREG01 to enable/disable breaking of memory transactions into chunks. The bit description is given in *Table 364*.

Table 364. INSNREG03 register bit assignments

Bit	Name	Reset value	Description
[00]	BMT	1'h0	Setting this bit enables break memory transfer.

22.6.21 INSNREG05 register

The INSNREG05 is a RW 32 bit register which allows to read the UTMI registers from the following signals:

Table 365. INSNREG05 register bit assignments

Bit	Name	Reset value	Description
[31:18]	Reserved	-	Read: undefined. Write: should be zero.
[17]	VBusy	1'h0	Software RO.
[16:13]	VPort	4'h0	Software R/W.
[12]	VControlLoadM	1'h0	1'b0 › Load new control word 1'b1 › NOP (Software R/W).
[11:08]	VControl	4'h0	Vendor control (software R/W).
[07:00]	VStatus	8'h0	Vendor status (software RO).

22.6.22 Register description of OHCI

22.6.23 Operation registers

The Host Controller (HC) contains a set of on-chip operational registers which are mapped into a noncacheable portion of the system addressable space. These registers are used by the Host Controller Driver (HCD). According to the function of these registers, they are

divided into four partitions, specifically for Control and Status, Memory Pointer, Frame Counter and Root Hub. All of the registers should be read and written as Dwords.

Reserved bits may be allocated in future releases of this specification. To ensure interoperability, the Host Controller Driver that does not use a reserved field should not assume that the reserved field contains 0. Furthermore, the Host Controller Driver should always preserve the value(s) of the reserved field. When a R/W register is modified, the Host Controller Driver should first read the register, modify the bits desired, then write the register with the reserved bits still containing the read value. Alternatively, the Host Controller Driver can maintain an in-memory copy of previously written values that can be modified and then written to the Host Controller register. When a write to set/clear register is written, bits written to reserved fields should be 0.

22.6.24 The control and status partition

22.6.25 HcRevision register

Table 366. HcRevision register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:08]					Reserved
[07:00]	REV	8'h10	R	R	Revision This read-only field contains the BCD representation of the version of the HCI specification that is implemented by this HC. For example, a value of 11h corresponds to version 1.1. All of the HC implementations that are compliant with this specification will have a value of 10h.

22.6.26 HcControl register

The HcControl register defines the operating modes for the Host Controller. Most of the fields in this register are modified only by the Host Controller Driver, except HostControllerFunctionalState and RemoteWakeupConnected.

Table 367. HcControl register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:11]	--				Reserved
[10]	RWE	0b	R/W	R	RemoteWakeupEnable This bit is used by HCD to enable or disable the remote wakeup feature upon the detection of upstream resume signaling. When this bit is set and the ResumeDetected bit in HcInterruptStatus is set, a remote wakeup is signaled to the host system. Setting this bit has no impact on the generation of hardware interrupt.

Table 367. HcControl register bit assignments (continued)

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[09]	RWC	0b	R/W	R	RemoteWakeupConnected This bit indicates whether HC supports remote wakeup signaling. If remote wakeup is supported and used by the system it is the responsibility of system firmware to set this bit during POST. HC clears the bit upon a hardware reset but does not alter it upon a software reset. Remote wakeup signaling of the host system is host-bus-specific and is not described in this specification.
[08]	IR	0b	R/W	R	InterruptRouting This bit determines the routing of interrupts generated by events registered in HcInterruptStatus. If clear, all interrupts are routed to the normal host bus interrupt mechanism. If set, interrupts are routed to the System Management Interrupt. HCD clears this bit upon a hardware reset, but it does not alter this bit upon a software reset. HCD uses this bit as a tag to indicate the ownership of HC.
[07:06]	HCFS	00b	R/W	R	HostControllerFunctionalState for USB 00b: USBRESET 01b: USBRESUME 10b: USBOPERATIONAL 11b: USBSUSPEND A transition to USBOPERATIONAL from another state causes SOF generation to begin 1 ms later. HCD may determine whether HC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus. This field may be changed by HC only when in the USBSUSPEND state. HC may move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. The latter also resets the RootHub and asserts subsequent reset signaling to downstream ports.
[05]	BLE	0b	R/W	R	BulkListEnable This bit is set to enable the processing of the Bulk list in the next Frame. If cleared by HCD, processing of the Bulk list does not occur after the next SOF. HC checks this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcBulkCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcBulkCurrentED before re-enabling processing of the list.

Table 367. HcControl register bit assignments (continued)

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[04]	CLE	0b	R/W	R	ControlListEnable This bit is set to enable the processing of the Control list in the next Frame. If cleared by HCD, processing of the Control list does not occur after the next SOF. HC must check this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcControlCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcControlCurrentED before re-enabling processing of the list.
[03]	IE	0b	R/W	R	IsochronousEnable This bit is used by HCD to enable/disable processing of isochronous EDs. While processing the periodic list in a Frame, HC checks the status of this bit when it finds an Isochronous ED (F=1). If set (enabled), HC continues processing the EDs. If cleared (disabled), HC halts processing of the periodic list (which now contains only isochronous EDs) and begins processing the Bulk/Control lists. Setting this bit is guaranteed to take effect in the next Frame (not the current Frame).
[02]	PLE	0b	R/W	R	PeriodicListEnable This bit is set to enable the processing of the periodic list in the next Frame. If cleared by HCD, processing of the periodic list does not occur after the next SOF. HC must check this bit before it starts processing the list.
[01:00]	CBSR	00b	R/W	R	ControlBulkServiceRatio This specifies the service ratio between Control and Bulk EDs. Before processing any of the nonperiodic lists, HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this value. 00 - 1:1 01 - 2:1 10 - 3:1 11 - 4:1

22.6.27 HcCommandStatus register

The HcCommandStatus register is used by the Host Controller to receive commands issued by the Host Controller Driver, as well as reflecting the current status of the Host Controller. To the Host Controller Driver, it appears to be a "write to set" register. The Host Controller must ensure that bits written as ‘1’ become set in the register while bits written as ‘0’ remain unchanged in the register. The Host Controller Driver may issue multiple distinct commands

to the Host Controller without concern for corrupting previously issued commands. The Host Controller Driver has normal read access to all bits.

The SchedulingOverrunCount field indicates the number of frames with which the Host Controller has detected the scheduling overrun error. This occurs when the Periodic list does not complete before EOF. When a scheduling overrun error is detected, the Host Controller increments the counter and sets the SchedulingOverrun field in the HcInterruptStatus register.

Table 368. HcCommandStatus register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:18]					Reserved
[17:16]	SOC	00b	R	R/W	SchedulingOverrunCount These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SchedulingOverrun in HcInterruptStatus has already been set. This is used by HCD to monitor any persistent scheduling problems.
[15:04]					Reserved
[03]	OCR	00b	R/W	R/R	OwnershipChangeRequest This bit is set by an OS HCD to request a change of control of the HC. When set HC will set the OwnershipChange field in HcInterruptStatus. After the changeover, this bit is cleared and remains so until the next request from OS HCD.
[02]	BLF	0b	R/W	R	BulkListFilled This bit is used to indicate whether there are any TDs on the Bulk list. It is set by HCD whenever it adds a TD to an ED in the Bulk list. When HC begins to process the head of the Bulk list, it checks BF. As long as BulkListFilled is 0, HC will not start processing the Bulk list. If BulkListFilled is 1, HC will start processing the Bulk list and will set BF to 0. If HC finds a TD on the list, then HC will set BulkListFilled to 1 causing the Bulk list processing to continue. If no TD is found on the Bulk list, and if HCD does not set BulkListFilled, then BulkListFilled will still be 0 when HC completes processing the Bulk list and Bulk list processing will stop.

Table 368. HcCommandStatus register bit assignments (continued)

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[01]	CLF	0b	R/W	R	ControlListFilled This bit is used to indicate whether there are any TDs on the Control list. It is set by HCD whenever it adds a TD to an ED in the Control list. When HC begins to process the head of the Control list, it checks CLF. As long as ControlListFilled is 0, HC will not start processing the Control list. If CF is 1, HC will start processing the Control list and will set ControlListFilled to 0. If HC finds a TD on the list, then HC will set ControlListFilled to 1 causing the Control list processing to continue. If no TD is found on the Control list, and if the HCD does not set ControlListFilled, then ControlListFilled will still be 0 when HC completes processing the Control list and Control list processing will stop.
[00]	HCR	0b	R/W	R	HostControllerReset This bit is set by HCD to initiate a software reset of HC. Regardless of the functional state of HC, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise; e.g., the InterruptRouting field of HcControl, and no Host bus accesses are allowed. This bit is cleared by HC upon the completion of the reset operation.The reset operation must be completed within 10 ms. This bit, when set, should not cause a reset to the Root Hub and no subsequent reset signaling should be asserted to its downstream ports.

22.6.28 HcInterruptStatus register

This register provides status on various events that cause hardware interrupts. When an event occurs, Host Controller sets the corresponding bit in this register. When a bit becomes set, a hardware interrupt is generated if the interrupt is enabled in the HcInterruptEnable register (see Section 7.1.5) and the MasterInterruptEnable bit is set. The Host Controller Driver may clear specific bits in this register by writing ‘1’ to bit positions to be cleared. The Host Controller Driver may not set any of these bits. The Host Controller will never clear the bit.

Table 369. HcInterruptStatus register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31]					Reserved
[30]	OC	0b	R/W	R/W	OwnershipChange This bit is set by HC when HCD sets the OwnershipChangeRequest field in HcCommandStatus. This event, when unmasked, will always generate an System Management Interrupt (SMI) immediately. This bit is tied to 0b when the SMI pin is not implemented.
[29:07]					Reserved
[06]	RHSC	0b	R/W/	R/W	RootHubStatusChange This bit is set when the content of HcRhStatus or the content of any of HcRhPortStatus[NumberofDownstreamPort] has changed.
[05]	FN0	0b	R/W	R/W	FrameNumberOverflow This bit is set when the MSb of HcFmNumber (bit 15) changes value, from 0 to 1 or from 1 to 0, and after HccaFrameNumber has been updated.
[04]	UE	0b	R/W	R/W	UnrecoverableError This bit is set when HC detects a system error not related to USB. HC should not proceed with any processing nor signaling before the system error has been corrected. HCD clears this bit after HC has been reset.
[03]	RD	0b	R/W	R/W	ResumeDetected This bit is set when HC detects that a device on the USB is asserting resume signaling. It is the transition from no resume signaling to resume signaling causing this bit to be set. This bit is not set when HCD sets the USBRESUME state.
[02]	SF	0b	R/W	R/W	StartofFrame This bit is set by HC at each start of a frame and after the update of HccaFrameNumber. HC also generates a SOF token at the same time.
[01]	WDH	0b	R/W	R/W	WritebackDoneHead This bit is set immediately after HC has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. HCD should only clear this bit after it has saved the content of HccaDoneHead.
[00]	SO	0b	R/W	R/W	SchedulingOverrun This bit is set when the USB schedule for the current Frame overruns and after the update of HccaFrameNumber. A scheduling overrun will also cause the SchedulingOverrunCount of HcCommandStatus to be incremented.

22.6.29 HcInterruptEnable register

Each enable bit in the HcInterruptEnable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptEnable register is used to control which events generate a hardware interrupt. When a bit is set in the HcInterruptStatus register AND the corresponding bit in the HcInterruptEnable register is set AND the MasterInterruptEnable bit is set, then a hardware interrupt is requested on the host bus.

Writing a '1' to a bit in this register sets the corresponding bit, whereas writing a '0' to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Table 370. HcInterruptEnable register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31]	MIE	0b	R/W	R	A ‘0’ written to this field is ignored by HC. A '1' written to this field enables interrupt generation due to events specified in the other bits of this register. This is used by HCD as a Master Interrupt Enable.
[30]	OC	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to Ownership Change.
[29:07]					Reserved
[06]	RHSC	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to Root Hub Status Change.
[05]	FNO	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to Frame Number Overflow.
[04]	UE	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to Unrecoverable Error.
[03]	RD	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to Resume Detect.
[02]	SF	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to Start of Frame.
[01]	WDH	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to HcDoneHead Writeback.
[00]	SO	0b	R/W	R	0 - Ignore 1 - Enable interrupt generation due to Scheduling Overrun.

22.6.30 HcInterruptDisable register

Each disable bit in the HcInterruptDisable register corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptDisable register is coupled with the

HcInterruptEnable register. Thus, writing a '1' to a bit in this register clears the corresponding bit in the HcInterruptEnable register, whereas writing a '0' to a bit in this register leaves the corresponding bit in the HcInterruptEnable register unchanged. On read, the current value of the HcInterruptEnable register is returned.

Table 371. HcInterruptDisable register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31]	MIE	0b	R/W	R	A '0' written to this field is ignored by HC. A '1' written to this field disables interrupt generation due to events specified in the other bits of this register. This field is set after a hardware or software reset.
[30]	OC	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to Ownership Change.
[29:07]					Reserved
[06]	RHSC	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to Root Hub Status Change.
[05]	FNO	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to Frame Number Overflow.
[04]	UE	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to Unrecoverable Error.
[03]	RD	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to Resume Detect.
[02]	SF	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to Start of Frame.
[01]	WDH	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to HcDoneHead Writeback.
[00]	SO	0b	R/W	R	0 - Ignore 1 - disable interrupt generation due to Scheduling Overrun.

22.6.31 Memory pointer partition

22.6.32 HcHCCA register

The HcHCCA register contains the physical address of the Host Controller Communication Area. The Host Controller Driver determines the alignment restrictions by writing all 1s to HcHCCA and reading the content of HcHCCA. The alignment is evaluated by examining the number of zeroes in the lower order bits. The minimum alignment is 256 bytes; therefore, bits 0 through 7 must always return '0' when read. Detailed description can be found in Chapter 4. This area is used to hold the control structures and the Interrupt table that are accessed by both the Host Controller and the Host Controller Driver.

Table 372. HcHCCA register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:08]	HCCA	0h	R/W	R	This is the base address of the Host Controller Communication Area.
[07:00]					Reserved

22.6.33 HcPeriodCurrentED register

The HcPeriodCurrentED register contains the physical address of the current Isochronous or Interrupt Endpoint Descriptor.

Table 373. HcPeriodCurrentED register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:04]	PCED	0h	R	R/W	PeriodCurrentED This is used by HC to point to the head of one of the Periodic lists which will be processed in the current Frame. The content of this register is updated by HC after a periodic ED has been processed. HCD may read the content in determining which ED is currently being processed at the time of reading.
[03:00]					Reserved

22.6.34 HcControlHeadED register

The HcControlHeadED register contains the physical address of the first Endpoint Descriptor of the Control list.

Table 374. HcControlHeadED register bit assignments

Bits	Name	Reset	Read/Write		Descritpion
			HCD	HC	
[31:04]	CHED	0h	R/W	R	ControlHeadED HC traverses the Control list starting with the HcControlHeadED pointer. The content is loaded from HCCA during the initialization of HC.
[03:00]					Reserved

22.6.35 HcControlCurrentED register

The HcControlCurrentED register contains the physical address of the current Endpoint Descriptor of the Control list.

Table 375. HcControlCurrentED register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:04]	CCED	0h	R/W	R/W	ControlCurrentED This pointer is advanced to the next ED after serving the present one. HC will continue processing the list from where it left off in the last Frame. When it reaches the end of the Control list, HC checks the ControlListFilled of in HcCommandStatus. If set, it copies the content of HcControlHeadED to HcControlCurrentED and clears the bit. If not set, it does nothing. HCD is allowed to modify this register only when the ControlListEnable of HcControl is cleared. When set, HCD only reads the instantaneous value of this register. Initially, this is set to zero to indicate the end of the Control list.
[03:00]					Reserved

22.6.36 HcBulkHeadED register

The HcBulkHeadED register contains the physical address of the first Endpoint Descriptor of the Bulk list.

Table 376. HcBulkHeadED register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:04]	BHED	0h	R/W	R	BulkHeadED HC traverses the Bulk list starting with the HcBulkHeadED pointer. The content is loaded from HCCA during the initialization of HC.
[03:00]					Reserved

22.6.37 HcBulkCurrentED register

The HcBulkCurrentED register contains the physical address of the current endpoint of the Bulk list. As the Bulk list will be served in a round-robin fashion, the endpoints will be ordered according to their insertion to the list.

Table 377. HcBulkCurrentED register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:04]	BCED	0h	R/W	R/W	BulkCurrentED This is advanced to the next ED after the HC has served the present one. HC continues processing the list from where it left off in the last Frame. When it reaches the end of the Bulk list, HC checks the ControlListFilled of HcControl. If set, it copies the content of HcBulkHeadED to HcBulkCurrentED and clears the bit. If it is not set, it does nothing. HCD is only allowed to modify this register when the BulkListEnable of HcControl is cleared. When set, the HCD only reads the instantaneous value of this register. This is initially set to zero to indicate the end of the Bulk list.
[03:00]					Reserved

22.6.38 HcDoneHead register

The HcDoneHead register contains the physical address of the last completed Transfer Descriptor that was added to the Done queue. In normal operation, the Host Controller Driver should not need to read this register as its content is periodically written to the HCCA.

Table 378. HcDoneHead register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:04]	DH	0h	R	R/W	DoneHead When a TD is completed, HC writes the content of HcDoneHead to the NextTD field of the TD. HC then overwrites the content of HcDoneHead with the address of this TD. This is set to zero whenever HC writes the content of this register to HCCA. It also sets the WritebackDoneHead of HcInterruptStatus.
[03:00]					Reserved

22.6.39 Frame counter partition

22.6.40 HcFmInterval register

The HcFmInterval register contains a 14 bit value which indicates the bit time interval in a Frame, (i.e., between two consecutive SOFs), and a 15 bit value indicating the Full Speed maximum packet size that the Host Controller may transmit or receive without causing scheduling overrun. The Host Controller Driver may carry out minor adjustment on the FrameInterval by writing a new value over the present one at each SOF. This provides the programmability necessary for the Host Controller to synchronize with an external clocking resource and to adjust any unknown local clock offset.

Table 379. HcFmInterval register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31]	FIT	0b	R/W‘	R	FrameIntervalToggle HCD toggles this bit whenever it loads a new value to FrameInterval.
[30:16]	FSMPS	TBD	R/W	R	FSLargestDataPacket This field specifies a value which is loaded into the Largest Data Packet Counter at the beginning of each frame. The counter value represents the largest amount of data in bits which can be sent or received by the HC in a single transaction at any given time without causing scheduling overrun. The field value is calculated by the HCD.
[15:14]					Reserved
[13:00]	FI	2EDFh	R/W	R	FrameInterval This specifies the interval between two consecutive SOFs in bit times. The nominal value is set to be 11,999. HCD should store the current value of this field before resetting HC. By setting the HostControllerReset field of HcCommandStatus as this will cause the HC to reset this field to its nominal value. HCD may choose to restore the stored value upon the completion of the Reset sequence.

22.6.41 HcFmRemaining register

The HcFmRemaining register is a 14 bit down counter showing the bit time remaining in the current Frame.

Table 380. HcFmRemaining register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31]	FRT	0b	R	R/W	FrameRemainingToggle This bit is loaded from the FrameIntervalToggle field of HcFmInterval whenever FrameRemaining reaches 0. This bit is used by HCD for the synchronization between FrameInterval and FrameRemaining.
[30:14]					Reserved
[13:00]	FR	0h	R	R/W	FrameRemaining This counter is decremented at each bit time. When it reaches zero, it is reset by loading the FrameInterval value specified in HcFmInterval at the next bit time boundary. When entering the USBOPERATIONAL state, HC re-loads the content with the FrameInterval of HcFmInterval and uses the updated value from the next SOF.

22.6.42 HcFmNumber register

The HcFmNumber register is a 16 bit counter. It provides a timing reference among events happening in the Host Controller and the Host Controller Driver. The Host Controller Driver may use the 16 bit value specified in this register and generate a 32 bit frame number without requiring frequent access to the register.

Table 381. HcFmNumber register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:16]					Reserved
[15:00]	FN	0h	R	R/W	FrameNumber This is incremented when HcFmRemaining is re-loaded. It will be rolled over to 0h after ffffh. When entering the USBOPERATIONAL state, this will be incremented automatically. The content will be written to HCCA after HC has incremented the FrameNumber at each frame boundary and sent a SOF but before HC reads the first ED in that Frame. After writing to HCCA, HC will set the StartofFrame in HcInterruptStatus.

22.6.43 HcPeriodicStart register

The HcPeriodicStart register has a 14 bit programmable value which determines when is the earliest time HC should start processing the periodic list.

Table 382. HcPeriodicStart register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:14]					Reserved
[13:00]	PS	0h	R/W	R	PeriodicStart After a hardware reset, this field is cleared. This is then set by HCD during the HC initialization. The value is calculated roughly as 10% off from HcFmInterval. A typical value will be 3E67h. When HcFmRemaining reaches the value specified, processing of the periodic lists will have priority over Control/Bulk processing. HC will therefore start processing the Interrupt list after completing the current Control or Bulk transaction that is in progress.

22.6.44 HcLSThreshold register

The HcLSThreshold register contains an 11 bit value used by the Host Controller to determine whether to commit to the transfer of a maximum of 8-byte LS packet before EOF. Neither the Host Controller nor the Host Controller Driver are allowed to change this value.

Table 383. HcLSThreshold register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:12]					Reserved
[11:00]	LST	0628h	R/W	R	LSThreshold This field contains a value which is compared to the FrameRemaining field prior to initiating a Low Speed transaction. The transaction is started only if FrameRemaining ³ this field. The value is calculated by HCD with the consideration of transmission and setup overhead.

22.6.45 Root hub partition

All registers included in this partition are dedicated to the USB Root Hub which is an integral part of the Host Controller though still a functionally separate entity. The HCD emulates USBD accesses to the Root Hub via a register interface. The HCD maintains many USB-defined hub features which are not required to be supported in hardware. For example, the Hub's Device, Configuration, Interface, and Endpoint Descriptors are maintained only in the HCD as well as some static fields of the Class Descriptor. The HCD also maintains and decodes the Root Hub's device address as well as other trivial operations which are better suited to software than hardware.

The Root Hub register interface is otherwise developed to maintain similarity of bit organization and operation to typical hubs which are found in the system. Below are four register definitions: HcRhDescriptorA, HcRhDescriptorB, HcRhStatus, and HcRhPortStatus[1:NDP]. Each register is read and written as a Dword. These registers are only written during initialization to correspond with the system implementation. The HcRhDescriptorA and HcRhDescriptorB registers should be implemented such that they are writeable regardless of the HC USB state. HcRhStatus and HcRhPortStatus must be writeable during the USBOPERATIONAL state.

Note: IS denotes an implementation-specific reset value for that field.

22.6.46 HcRhDescriptorA register

The HcRhDescriptorA register is the first register of two describing the characteristics of the Root Hub. Reset values are implementation-specific. The descriptor length (11), descriptor type (TBD), and hub controller current (0) fields of the hub Class Descriptor are emulated by the HCD. All other fields are located in the HcRhDescriptorA and HcRhDescriptorB registers.

Table 384. HcRhDescriptorA register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:24]	POTPGT	IS	R/W	R	PowerOnToPowerGoodTime This byte specifies the duration HCD has to wait before accessing a powered-on port of the Root Hub. It is implementation-specific. The unit of time is 2 ms. The duration is calculated as POTPGT * 2 ms.
[23:13]					Reserved
[12]	NOCP	IS	R/W	R	NoOverCurrentProtection This bit describes how the overcurrent status for the Root Hub ports are reported. When this bit is cleared, the OverCurrentProtectionMode field specifies global or per-port reporting. 0: Over-current status is reported collectively for all downstream ports 1: No overcurrent protection supported
[11]	OCPM	IS	R/W	R	OverCurrentProtectionMode This bit describes how the overcurrent status for the Root Hub ports are reported. At reset, this fields should reflect the same mode as PowerSwitchingMode. This field is valid only if the NoOverCurrentProtection field is cleared. 0: over-current status is reported collectively for all downstream ports 1: over-current status is reported on a per-port basis
[10]	DT	0b	R	R	DeviceType This bit specifies that the Root Hub is not a compound device. The Root Hub is not permitted to be a compound device. This field should always read/write 0.
[09]	NPS	IS	R/W	R	NoPowerSwitching These bits are used to specify whether power switching is supported or port are always powered. It is implementationspecific. When this bit is cleared, the PowerSwitchingMode specifies global or per-port switching. 0: Ports are power switched 1: Ports are always powered on when the HC is powered on

Table 384. HcRhDescriptorA register bit assignments (continued)

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[08]	PSM	IS	R/W	R	PowerSwitchingMode This bit is used to specify how the power switching of the Root Hub ports is controlled. It is implementation-specific. This field is only valid if the NoPowerSwitching field is cleared. 0: all ports are powered at the same time. 1: each port is powered individually. This mode allows port power to be controlled by either the global switch or perport switching. If the PortPowerControlMask bit is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, then the port is controlled only by the global power switch (Set/ClearGlobalPower).
[07:00]	NDP	IS	R/W	R	NumberDownstreamPorts These bits specify the number of downstream ports supported by the Root Hub. It is implementation-specific. The minimum number of ports is 1. The maximum number of ports supported by OpenHCI is 15.

22.6.47 HcRhDescriptorB register

The HcRhDescriptorB register is the second register of two describing the characteristics of the Root Hub. These fields are written during initialization to correspond with the system implementation. Reset values are implementation-specific.

Table 385. HcRhDescriptorB register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31:16]	PPCM	IS	R/W	R	PortPowerControlMask Each bit indicates if a port is affected by a global power control command when PowerSwitchingMode is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (PowerSwitchingMode=0), this field is not valid. bit 0: Reserved bit 1: Ganged-power mask on Port #1 bit 2: Ganged-power mask on Port #2 ... bit15: Ganged-power mask on Port #15
[15:00]	DR	IS	R/W	R	DeviceRemovable Each bit is dedicated to a port of the Root Hub. When cleared, the attached device is removable. When set, the attached device is not removable. bit 0: Reserved bit 1: Device attached to Port #1 bit 2: Device attached to Port #2 ... bit15: Device attached to Port #15

22.6.48 HcRhStatus register

The HcRhStatus register is divided into two parts. The lower word of a Dword represents the Hub Status field and the upper word represents the Hub Status Change field. Reserved bits should always be written '0'.

Table 386. HcRhStatus register bit assignments

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[31]	CRWE		W	R	(write) ClearRemoteWakeupEnable Writing a '1' clears DeviceRemoveWakeupEnable. Writing a '0' has no effect.
[30:18]					Reserved
[17]	OCIC	0b	R/W	R/W	OverCurrentIndicatorChange This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing a ‘1’. Writing a ‘0’ has no effect.

Table 386. HcRhStatus register bit assignments (continued)

Bits	Name	Reset	Read/Write		Description
			HCD	HC	
[16]	LPSC	0b	R/W	R	(read) LocalPowerStatusChange The Root Hub does not support the local power status feature; thus, this bit is always read as '0'.(write) SetGlobalPower In global power mode (PowerSwitchingMode=0), This bit is written to '1' to turn on power to all ports (clear PortPowerStatus). In per-port power mode, it sets PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a '0' has no effect.
[15]	DRWE	0b	R/W	R	(read) DeviceRemoteWakeupEnable This bit enables a ConnectStatusChange bit as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the ResumeDetected interrupt. 0 = ConnectStatusChange is not a remote wakeup event. 1 = ConnectStatusChange is a remote wakeup event. (write) SetRemoteWakeupEnable Writing a '1' sets DeviceRemoveWakeupEnable. Writing a '0' has no effect.
[14:02]					Reserved
[01]	OCI	0b	R	R/W	OverCurrentIndicator This bit reports overcurrent conditions when the global reporting is implemented. When set, an overcurrent condition exists. When cleared, all power operations are normal. If per-port overcurrent protection is implemented this bit is always '0'
[00]	LPS	0b	R/W	R	(read) LocalPowerStatus The Root Hub does not support the local power status feature; thus, this bit is always read as '0'. (write) ClearGlobalPower In global power mode (PowerSwitchingMode=0), This bit is written to '1' to turn off power to all ports (clear PortPowerStatus). In per-port power mode, it clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a '0' has no effect.

22.6.49 HcRhPortStatus[1:NDP] register

The HcRhPortStatus[1:NDP] register is used to control and report port events on a per-port basis. NumberDownstreamPorts represents the number of HcRhPortStatus registers that are implemented in hardware. The lower word is used to reflect the port status, whereas the upper word reflects the status change bits. Some status bits are implemented with special write behavior (see below). If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change must be postponed until the transaction completes. Reserved bits should always be written '0'.

Table 387. HcRhPortStatus register bit assignments

Bits	Name	Reset	Read/write		Description
			HCD	HC	
[31:21]					Reserved
[20]	PRSC	0b	R/W	R/W	PortResetStatusChange This bit is set at the end of the 10-ms port reset signal. The HCD writes a '1' to clear this bit. Writing a '0' has no effect. 0 = port reset is not complete 1 = port reset is complete
[19]	OCIC	0b	R/W	R/W	PortOverCurrentIndicatorChange This bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when Root Hub changes the PortOverCurrentIndicator bit. The HCD writes a '1' to clear this bit. Writing a '0' has no effect. 0 = no change in PortOverCurrentIndicator 1 = PortOverCurrentIndicator has changed
[18]	PSSC	0b	R/W	R/W	PortSuspendStatusChange This bit is set when the full resume sequence has been completed. This sequence includes the 20-s resume pulse, LS EOP, and 3-ms resychronization delay. The HCD writes a '1' to clear this bit. Writing a '0' has no effect. This bit is also cleared when ResetStatusChange is set. 0 = resume is not completed 1 = resume completed
[17]	PESC	0b	R/W	R/W	PortEnableStatusChange This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Changes from HCD writes do not set this bit. The HCD writes a '1' to clear this bit. Writing a '0' has no effect. 0 = no change in PortEnableStatus 1 = change in PortEnableStatus
[16]	CSC	0b	R/W	R/W	ConnectStatusChange This bit is set whenever a connect or disconnect event occurs. The HCD writes a '1' to clear this bit. Writing a '0' has no effect. If CurrentConnectStatus is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected. 0 = no change in CurrentConnectStatus 1 = change in CurrentConnectStatus Note: If the DeviceRemovable[NDP] bit is set, this bit is set only after a Root Hub reset to inform the system that the device is attached.
[15:10]					Reserved

Table 387. HcRhPortStatus register bit assignments (continued)

Bits	Name	Reset	Read/write		Description
			HCD	HC	
[09]	LSDA	xb	R/W	R/W	(read) LowSpeedDeviceAttached This bit indicates the speed of the device attached to this port. When set, a Low Speed device is attached to this port. When clear, a Full Speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set. 0 = full speed device attached 1 = low speed device attached (write) ClearPortPower The HCD clears the PortPowerStatus bit by writing a '1' to this bit. Writing a '0' has no effect.
[08]	PPS	0b	R/W	R/W	(read) PortPowerStatus This bit reflects the port's power status, regardless of the type of power switching implemented. This bit is cleared if an overcurrent condition is detected. HCD sets this bit by writing SetPortPower or SetGlobalPower. HCD clears this bit by writing ClearPortPower or ClearGlobalPower. Which power control switches are enabled is determined by PowerSwitchingMode and PortPortControlMask[NDP]. In global switching mode (PowerSwitchingMode=0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PowerSwitchingMode=1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset. 0 = port power is off 1 = port power is on (write) SetPortPower The HCD writes a '1' to set the PortPowerStatus bit. Writing a '0' has no effect. Note: This bit is always reads '1b' if power switching is not supported.
[07:05]					Reserved

Table 387. HcRhPortStatus register bit assignments (continued)

Bits	Name	Reset	Read/write		Description
			HCD	HC	
[04]	PRS	0b	R/W	R/W	(read) PortResetStatus When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. 0 = port reset signal is not active 1 = port reset signal is active (write) SetPortReset The HCD sets the port reset signaling by writing a '1' to this bit. Writing a '0' has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to reset a disconnected port.
[03]	POCI	0b	R/W	R/W	(read) PortOverCurrentIndicator This bit is only valid when the Root Hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal 0 = no overcurrent condition. 1 = overcurrent condition detected. (write) ClearSuspendStatus The HCD writes a '1' to initiate a resume. Writing a '0' has no effect. A resume is initiated only if PortSuspendStatus is set.
[02]	PSS	0b	R/W	R/W	(read) PortSuspendStatus This bit indicates the port is suspended or in the resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatusChange is set at the end of the port reset or when the HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC. 0 = port is not suspended 1 = port is suspended (write) SetPortSuspend The HCD sets the PortSuspendStatus bit by writing a '1' to this bit. Writing a '0' has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus; instead it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port.

Table 387. HcRhPortStatus register bit assignments (continued)

Bits	Name	Reset	Read/write		Description
			HCD	HC	
[01]	PES	0b	R/W	R/W	(read) PortEnableStatus This bit indicates whether the port is enabled or disabled. The Root Hub may clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error such as babble is detected. This change also causes PortEnabledStatusChange to be set. HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable. This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if not already, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. 0 = port is disabled 1 = port is enabled (write) SetPortEnable The HCD sets PortEnableStatus by writing a '1'. Writing a '0' has no effect. If CurrentConnectStatus is cleared, this write does not set PortEnableStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to enable a disconnected port.
[00]	CCS	0b	R/W	R/W	(read) CurrentConnectStatus This bit reflects the current state of the downstream port. 0 = no device connected 1 = device connected (write) ClearPortEnable The HCD writes a '1' to this bit to clear the PortEnableStatus bit. Writing a '0' has no effect. The CurrentConnectStatus is not affected by any write. Note: This bit is always read '1b' when the attached device is nonremovable (DeviceRemoveable[NDP]).

23 HS_USB 2.0 device

23.1 Overview

In addition to single independent USB 2.0 hosts, within its high-speed (HS) connection subsystem SPEAr300 provides a USB 2.0 Device which is fully compliant with the universal serial bus specification (version 2.0), and offering an interface to the industry-standard AHB bus.

Main features provided by the USB 2.0 device are listed:

- A PHY interface implementing a USB 2.0 transceiver macrocell interface (UTMI) fully compliant with UTMI specification (version 1.05), to execute serialization and deserialization of transmissions over the USB line.
- Unidirectional/bidirectional 8-/16 bit UTMI data bus interfaces are supported.
- A USB plug detect (UPD) which detects the connection of a device (detailed in *Section 23.7*).
- A USB device controller (UDC) which is connected to the AHB bus and generates the commands for the UTMI PHY. Hereafter the UDC along with AHB interface is referred to UDC-AHB subsystem.
- The UDC-AHB supports the 480 Mbps high-speed (HS) for USB 2.0, as well as the 12 Mbps full-speed (FS) for USB 1.1.
- The UDC-AHB supports 16 physical endpoints (listed in *Table 388*), and proper configurations to achieve logical endpoints.
- Both DMA mode and slave-only mode supported (detailed in *Section 23.4*).
- In DMA mode, the UDC-AHB supports descriptor-based memory structures in application memory.
- In both modes, an AHB slave is provided by UDC-AHB, acting as programming interface to access to memory-mapped control and status registers (CSRs).
- An AHB master for data transfer to system memory, supporting 8, 16, and 32 bit wide data transactions on the AHB bus.

Table 388. Endpoints assignment

Endpoint number	Endpoint direction	Transfer type
0	IN/OUT	Control.
1-3-5-7-9-11-13-15	IN	Software configurable to: Bulk Interrupt Isochronous.
2-4-6-8-10-12-14	OUT	Software configurable to: Bulk Interrupt Isochronous.

23.2 Block diagram

Figure 38 shows the block diagram of the UDC-AHB subsystem.

Figure 38. UDC-AHB subsystem block diagram within the USB 2.0 device

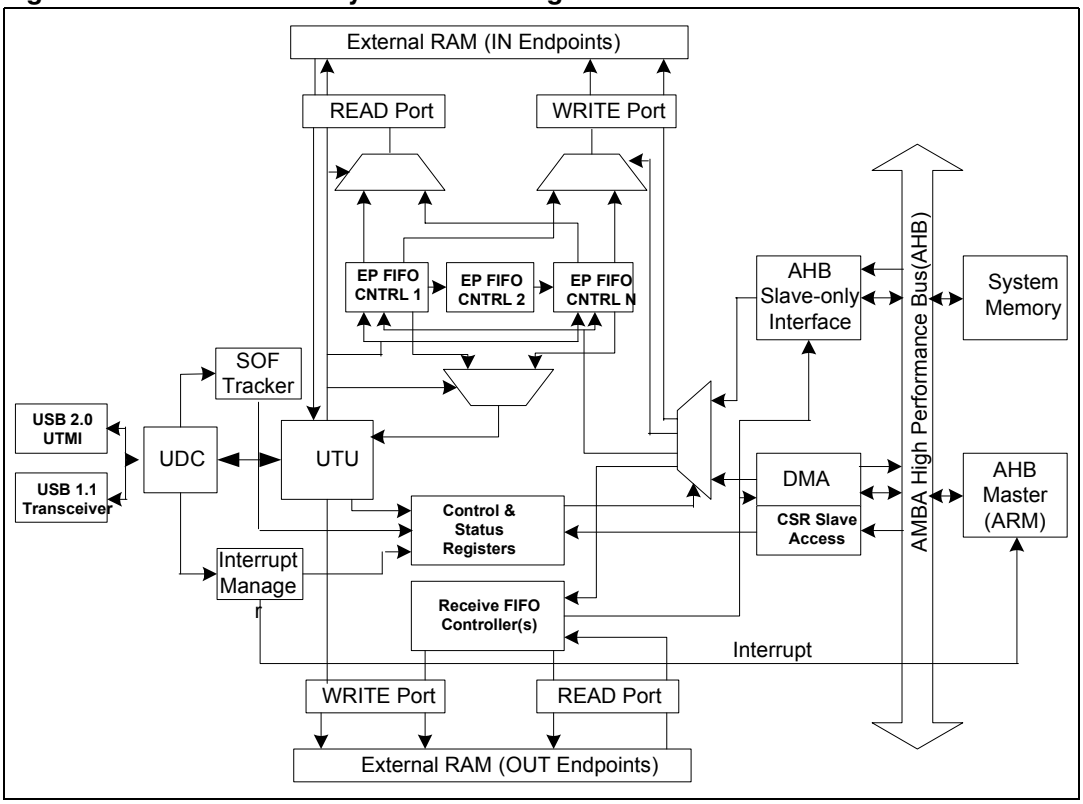
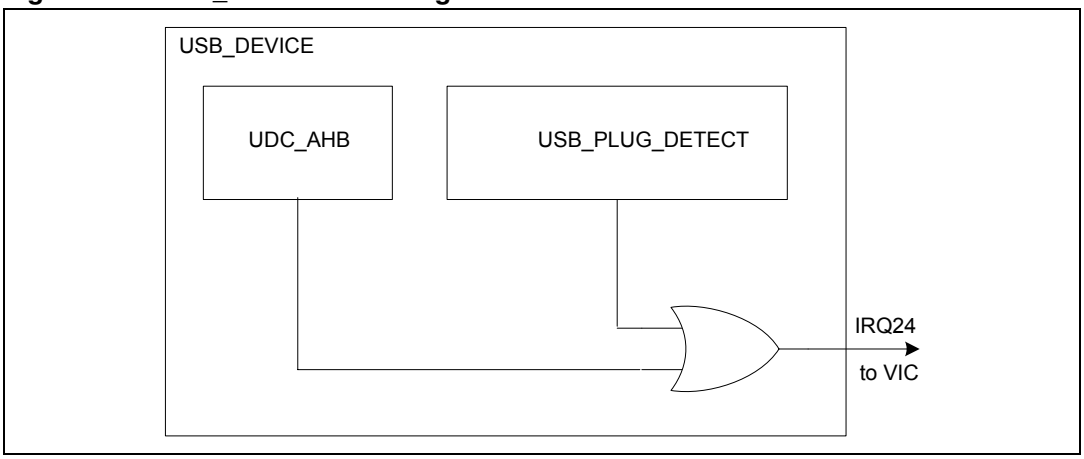


Figure 39. UDC_Device block diagram



23.3 Main functions description

23.3.1 UTLI

The USB *transaction layer interface* (UTLI) of the UDC-AHB subsystem interfaces with the UDC and the FIFOs to handle data reception/transmission with and USB host.

Main tasks of UTLI are:

- Interfaces to the UDC,
- Interfaces to endpoint-specific TxFIFOs (*Section 23.3.5: Endpoint FIFO controller (Transmit FIFO controller)*) when transmitting data in response to in requests from USB host,
- Interfaces to the common RxFIFO (*Section 23.3.4: Receive FIFO controller*) when receiving out data from the USB host,
- Works with the CSRs block (*Section 23.3.6: Control and status registers*) to maintain correct status and control,
- Works with the interrupt manager block (*Section 23.3.2: Interrupt manager*) to generate proper interrupts to the application,
- Interfaces to the SOF tracker (*Section 23.3.3: SOF tracker*) to ensure that isochronous data is transmitted in intended frame.

In particular, during data reception from an USB Host (that is, an out transaction), the UTLI directly read incoming data from the UDC and writes them to the receive FIFO (*Section 23.3.4: Receive FIFO controller*). Besides, for data transmission to an USB Host (that is, an in transaction), the UTLI reads data to be transmitted from relevant endpoint FIFO (*Section 23.3.5: Endpoint FIFO controller (Transmit FIFO controller)*) and provides them to UDC.

23.3.2 Interrupt manager

The *interrupt manager* block controls the generation of interrupts to the application. In particular, exchanging information with the UTLI, an interrupt is issued by the *interrupt mana*ger when any of the following device-level events occurs:

- Reception of a SOF token from the USB Host,
- Detection of a USB suspend,
- Detection of a USB reset,
- Completion of speed enumeration,
- Reception of a Set Interface command (defined in USB specification),
- Reception of a Set Configuration command (defined in USB specification).

In addition, the interrupt manager also triggers an interrupt when any of the following endpoint-specific events occurs:

- Reception of a request for in data,
- Reception of an out data packet,
- Reception of 8 bytes of SETUP data packet,
- An application error resulting in an AHB error response.

The interrupt manager block maintains the device interrupt register (*Device interrupt register on page 495*) and the device interrupt mask register (*Device interrupt mask register on*

page 497), which are mapped into the address space of the global control and status registers (CSRs, Section 23.3.6: Control and status registers).

Interrupt (IRQ24) issued will be the OR of all active events defined above, plus the plug detect interrupt.

23.3.3 SOF tracker

The USB host sends start-of-frame (SOF) packets to USB 2.0 device every 1 ms for full-speed (FS) operation, and every 125 µs for high-speed (HS) operation. Each SOF token represents the start of every frame (for FS) or micro-frame (for HS) respectively, in case of isochronous (ISO) data synchronization.

The start-of-frame (SOF) tracker block within the UDC-AHB subsystem is intended to track any incoming SOF packets from the USB Host. With this aim, the SOF tracker runs internal frame counters according to the operation rate (that is, 1 ms for FS and 125 µs for HS).

When a SOF packet is received from the USB Host, the UDC gets the 11 bit frame number from the packet, and gives it to the back end within a single clock pulse, indicating the reception of a SOF token.

In contrast, if a missing SOF packet is detected, the SOF tracker generates an event that is used by the ISO in FIFOs to clear residual data from the previous frame, whereas UDC-AHB subsystem moves to the next frame to provide synchronization.

In order to provide backward-compatibility with the FS 1 ms frame of USB 1.1, in HS mode the frame number is incremented by UDC once every eight 125 µs micro-frames only. As a consequence, the SOF tracker module generates the correct 14 bit micro-frame number by adding a 3 bit micro-frame counter (operated by the SOF tracker itself) to the 11 bit frame number provided by the UDC.

Note: In order to provide backward-compatibility with the FS 1 ms frame of USB 1.1, in HS mode the frame number is incremented by UDC once every eight 125 µs micro-frames only. As a consequence, the SOF tracker module generates the correct 14 bit micro-frame number by adding a 3 bit micro-frame counter (operated by the SOF tracker itself) to the 11 bit frame number provided by the UDC.

23.3.4 Receive FIFO controller

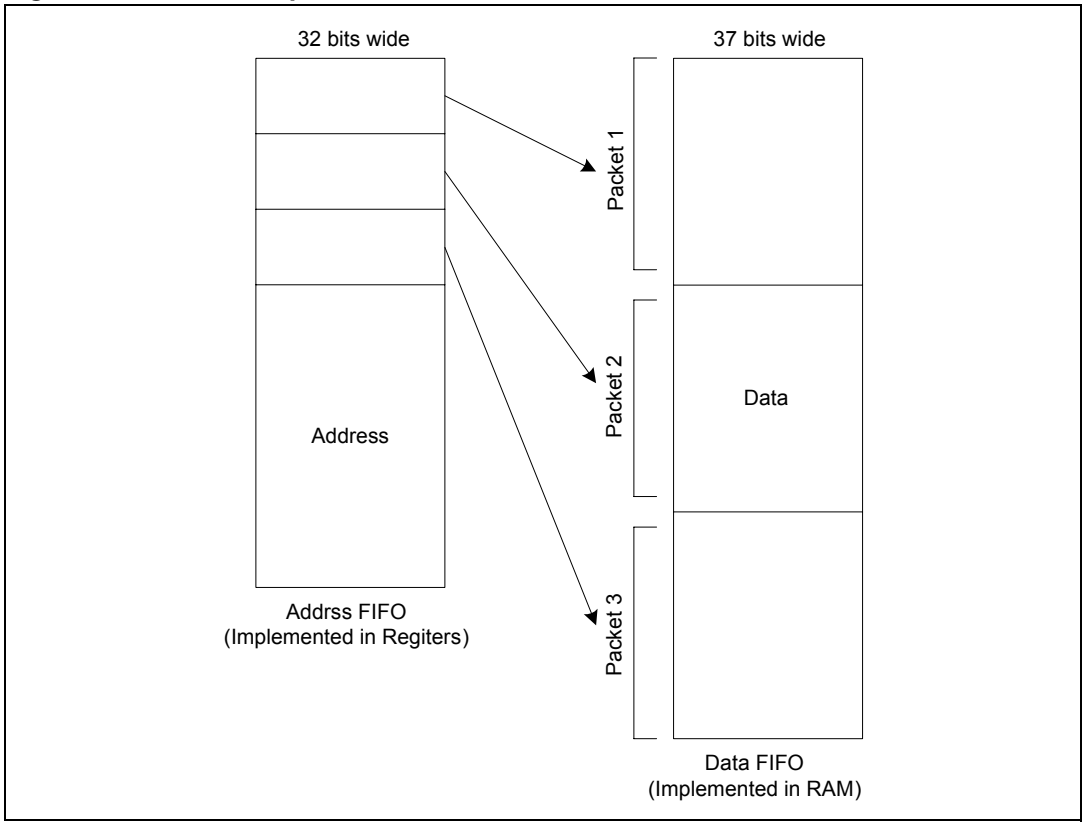
All out endpoints (dedicated to transactions coming from the USB Host) share a common receive FIFO (RxFIFO), which is managed by multiple *receive FIFO controller*. In particular, the RxFIFO provides the UTLI with enough space to either accept the incoming packet from the USB Host or send a NYET (UDC20 only) or a NAK handshake packet.

In particular, the RxFIFO consists of two individual FIFOs, one for the data and one for the addresses. As depicted in Figure 40, the data FIFO is implemented as RAM, whereas the address FIFO is implemented using registers. Each 32 bit wide entry in the address FIFO corresponds to a received out packet, and it is associated to both the destination endpoint number and a flag to distinguish regular data from the 8 bytes of SETUP data.

Note: The total data FIFO size is 4KB. Out of the 37 bits wide data 32 bits is OUT/IN data and rest 5 bits is status information. For RxFIFO the maximum depth of address FIFO is 4, hence at a given time maximum 4 OUT packets can be accommodated simultaneously. The depth of the data RXFIFO is limited to 2kB. So during simultaneous storing of 4 OUT packet, each packet would not be more than 512 bytes. But if OUT packets are of 1024 bytes (maximum size) then only two OUT packets can be accommodated. Hence number of OUT packet

accommodation is limited by the size of the OUT packets. Rest of the data FIFO i.e. 2 KB (out of total 4KB dada FIFO) is used for TXFIFO.

Figure 40. RxFIFO implementation



Upon receiving an out packet, UTLI strobes this data into the data FIFO (37 bit wide), and the UDC sends a status bit indicating whether or not the data was received without errors. If data reception was error-free, the UTLI confirms the data in the RxFIFO by writing the relevant endpoint number and associated flag into the address FIFO. In contrast, if data was received with errors, the *receive FIFO controller* rolls back the data FIFO pointers as if nothing had been received.

Then, when an external AHB master tries to access the packet received for a particular out endpoint, at first it must read the relevant endpoint status register (*Endpoint status register on page 499*) to determine the number of bytes to be transferred, before to start the appropriate AHB transfers with an appropriate `HSIZE` (i.e 32, 16, or 8).

Note: Any attempt to write the RxFIFO via the AHB interface results in an AHB error.

The RxFIFO also requires a confirming signal when a packet is written to or read from it. This confirmation is used by the *receive FIFO controller* to propagate pointer information from one domain to another and to calculate different RxFIFO status signals.

23.3.5 Endpoint FIFO controller (Transmit FIFO controller)

An *endpoint FIFO controller* block manages the FIFO of a specific in endpoint (dedicated to transactions to the USB Host) supported by the UDC-AHB subsystem. In particular, each in endpoint is associated to a Transmit FIFO (TxFIFO) which is mapped in external RAM, and each TxFIFO is in charge of an *endpoint FIFO controller*.

Each *endpoint FIFO controller* maintains the write and read pointers to access the memory where relevant TxFIFO is located. Besides, these controllers need both the base address and the buffer size of each endpoint TxFIFO to implement adaptive buffer management. This feature allows to tailor the size of each TxFIFO depending on specific buffering requirements.

In particular, the base address of each TxFIFO results from the upper limit of the previous TxFIFO which, in turn, depends on the buffer size set by the CSRs for this endpoint (buffer size register, *Endpoint buffer size and received packet frame number register on page 501*). That is, the base address of the TxFIFO associated to n^{th} in endoint added to the size of the TxFIFO of the same n^{th} in endpoint represents the base address for the TxFIFO associated to the $(n+1)^{th}$ in endpoint.

Note: Any attempt to read from TxFIFO via the AHB interface results in an AHB error.

Like *r*eceive FIFO controller, the transmit *endpoint FIFO controller* also requires a confirmation signal indicating a successful transfer to TxFIFO. This confirmation signal allows the controller to export the FIFO pointers to other domains.

23.3.6 Control and status registers

The *control and status registers* (CSRs) allow to exchange control information with the application, as well as provide a means for the application to control the UDC-AHB Subsystem.

23.3.7 AHB slave-only interface

The *AHB slave-only interface* block is active only when the UDC-AHB subsystem is configured as a slave on the AHB.

In this scenario, all endpoint FIFOs are mapped to the system memory, and the application writes the data directly to the endpoint FIFOs. Similarly, the RxFIFO is also mapped to the system memory, and the application reads directly from the RxFIFO.

23.3.8 DMA (AHB master interface)

Enabling the UDC-AHB Subsystem to become an AHB master (that is, entering the DMA mode, *Section 23.4.1: DMA mode*), the *DMA* block receives the required data pointers from the values programmed in the CSRs (*Section 23.3.6: Control and status registers*), and it can transfer data with the system memory.

In particular, the *DMA* supports a true scatter/gather memory distribution, where each endpoint memory structure is implemented as a linked-list (as detailed in *Section 23.4.1: DMA mode*).

The DMA block (which is inactive in slave-only mode) consists of three basic components:

- The DMA transfer engine, which moves the actual data,
- The DMA controller, which controls the movement of the data,
- The AHB interface, which manages the flow of data between the DMA and AHB for both data transfer and CSRs accesses.

23.3.9 DMA transfer engine

The *DMA transfer engine* is a slave to the DMA by the DMA controller for the actual data transfer to and from system memory.

In case of a memory access, the *DMA transfer engine* interfaces with the FIFOs and the AHB interface module of DMA, and indicates to the DMA whether or not the transfer was successful. If the data transfer was unsuccessful, the *DMA transfer engine* also indicates how many bytes were successfully transferred to the destination, so that the DMA can decide whether to retry the transaction.

In case of data transfer from the FIFOs to system memory, the *DMA transfer engine* depends on status signals from the FIFOs' respective FIFO controllers. In case of transferring the data to system memory, the *DMA transfer engine* registers the data, then waits for the request from the DMA controller and decides on the direction of the transfer. It completes the transfer whether the transaction is completed successfully or if there is an error during the data transfer.

23.3.10 DMA controller

The *DMA controller* is in charge of all data exchange between FIFOs and system memory. Specifically, the *DMA controller* actually consists of two distinct controllers with the aim to manage both in (transmit) and out (receive) transactions simultaneously, although transmit and receive functions cannot be performed simultaneously.

From a functional perspective, the *DMA controller* parses the descriptor structures and then commands the other subsystem blocks to perform data transfers accordingly. The descriptors fetched from memory are stored by the *DMA controller* in a proper descriptors buffer.

23.3.11 AHB interface

This block contains all the subsystem's AHB protocol logic. In particular, the *AHB interface* has two functional states where it is able to act:

- As a AHB slave, when the application programs the CSRs of either the UDC-AHB Subsystem or the UDC (*Section 23.3.7*),
- As a AHB master, when the DMA performs data transfers.

Acting as AHB master, the UDC-AHB subsystem accesses the application memory for descriptors and data buffers. When the subsystem is in slave-only mode, the *AHB interface* also acts as a slave. In this mode, all the FIFOs are memory-mapped, and the application writes directly to the FIFOs.

23.3.12 CSRs slave access

The *CSRs slave access* block is active in DMA mode only (*Section 23.4.1: DMA mode*) and, acting as an AHB slave, it responds to any CSRs access from the application (which acts as an AHB master).

In DMA mode CSR registers are accessible through CSR slave access block as this time AHB slave only block is de-activated. This AHB slave only block is activated only in slave mode.

23.4 Theory of operation

The UDC-AHB Subsystem supports two distinct operation modes:

- DMA mode (detailed in *Section 23.4.1*), a DMA-based implementation where the UDC-AHB Subsystem acts as an AHB master for data transfers.
- Slave-Only mode (detailed in *Section 23.4.2*), where the UDC-AHB Subsystem is slaved to the application and any application AHB master reads data from, or writes data to the memory-mapped FIFOs provided by the device.

In both modes, all data transfers are interrupt-driven.

23.4.1 DMA mode

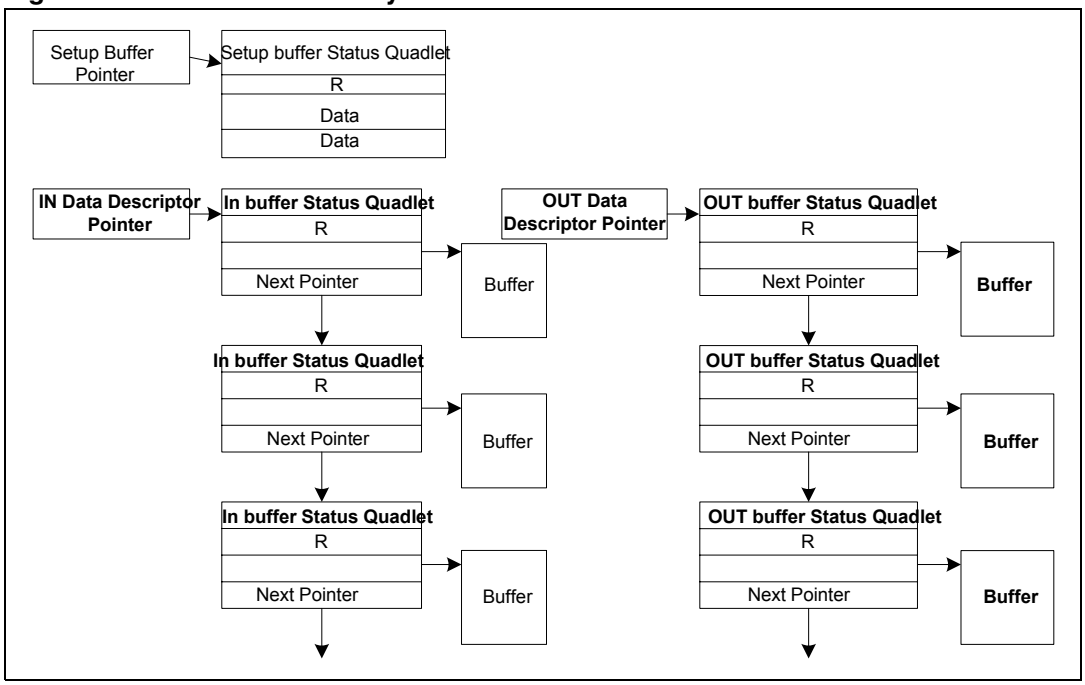
In general, a major advantage of DMA-based implementations is that they spare the main processor's computing power from involvement in data transfer tasks. Moreover, use of a scatter-gather DMA helps applications to make efficient and optimal use of system memory, which is indeed a major design constraint on portable systems.

Specifically, in DMA mode, the UDC-AHB subsystem implements a true scatter-gather memory distribution, in which memory structures are scattered over the system memory. As illustrated in *Figure 41*, each in/out endpoint memory structure is implemented as a linked-list, where each element of the list is a data buffer of a predefined size.

In addition to data (both in and out), each buffer also has a status quadlet and a pointer to the next buffer. The last element of such a linked list can point either to a null pointer or, if the linked list is implemented as a ring buffer, to the first element of the list. Data buffer structure for both in and out endpoint is described in *Section 23.5.2* and *Section 23.5.3*, respectively.

Besides, all control endpoints implement an additional 16-byte buffer to store SETUP data. The SETUP data structure (detailed in *Section 23.5.1*) does not implement a linked-list structure.

Figure 41. Linked-list memory structure in DMA mode



In DMA mode, before starting any action, the application must both initialize the buffer descriptor chains in the DMA data memory structure for all active endpoints of the UDC-AHB subsystem, and configure the required CSRs during the USB reset.

A brief description of both in and out operation in DMA mode.

23.4.2 In operation (Data transfer To USB host)

If the UDC-AHB subsystem receives an in token from a USB host for a non-isochronous endpoint (such as, bulk, interrupt or control), it checks the corresponding TxFIFO for data availability. If data is available, the TxFIFO is read and the data is provided to the UDC for transfer to USB host. In contrast, if the TxFIFO is empty (no data), the UDC-AHB subsystem sends an interrupt to the application and the UDC sends a NAK handshake to the USB Host connected to that endpoint.

On receiving the interrupt, at first the application probes the endpoint interrupt register, *Endpoint interrupt register on page 497* to determine which endpoint has requested the interrupt. Having determined this endpoint, then the application probes the endpoint status register, *Endpoint status register on page 499* to determine the interrupt's cause.

Upon notification that this is an in token for a particular endpoint, the application updates the addressed endpoint's system memory buffer with data. Besides, the application reports to the DMA the availability of such data by setting the poll demand bit in the subsystem's CSRs.

Note: Each endpoint has a dedicated poll demand bit within CSRs, specifically in the endpoint-specific endpoint control register, Endpoint control register on page 498.

Now the DMA transfers the data from the system memory to the relevant endpoint FIFO. As shown in *Figure 41*, these endpoint buffers are RAM-based implementations with programmable sizes. When the USB host retries with another in token, the UDC-AHB subsystem provides the data to the UDC reading the endpoint buffers for transmission to

USB host. When the transmission is complete, the status is written back into the buffer descriptor's status quadlet. Then, the subsystem clears the endpoint-specific poll demand bit once the descriptor chain reaches the last descriptor.

Note: The application can read the poll demand bit to determine if the descriptor chain is serviced or not.

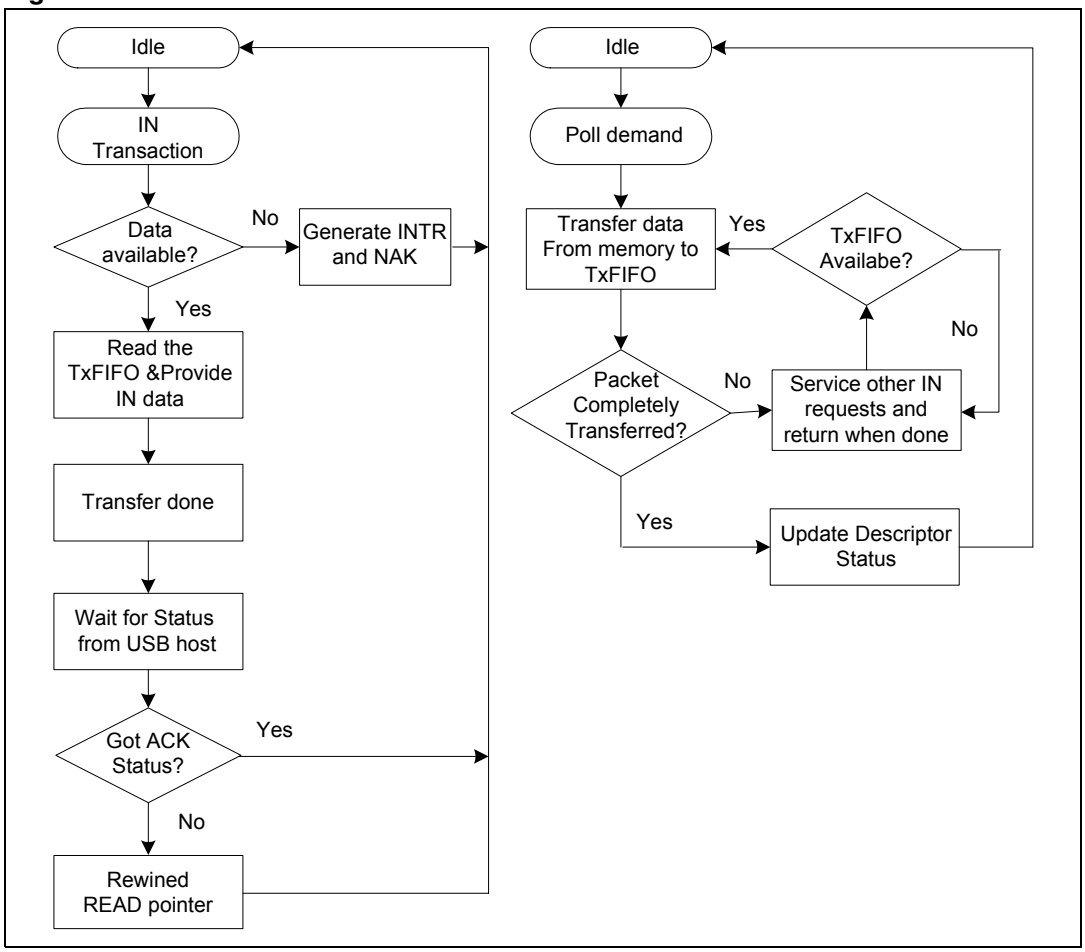
In transfers with ISO endpoints are handled similarly. In this case, the transfer of in data from the application memory to the endpoint FIFO is not initiated by request tokens from the USB host, but the application sets the poll demand bit of CSR as soon as data is available.

Following bit setting, the UDC-AHB subsystem tags data for isochronous endpoints with a frame number. The UDC, which maintains the frame counter, sends the isochronous data in the intended frame, whereas the SOF tracker module (*SOF tracker on page 466*) tracks the incoming SOFs and their frame numbers. Three distinct scenarios can be raised up:

- if the incoming frame number matches the frame number in the buffer, the UDC is allowed to transfer the frame from the appropriate data buffer.
- if the frame number in the SOF is greater than the frame number in the frame counter of UDC, the DMA module skips the buffers to align to the correct frame number.
- if the frame number in the SOF is less than the subsystem's frame number, the DMA waits for a few frames to align to the correct frame number.

Hooks are provided for the application to flush the subsystem's FIFOs in case of missing SOFs. The transaction flow of in data from the USB Host to the application memory is given in *Figure 42*.

Figure 42. In transaction flow in DMA mode



23.4.3 Out operation (Data transfer from USB host)

In the out direction, as soon as the UDC-AHB subsystem receives an out (or SETUP) data from the USB host (that is, when a packet of data is completed or - if thresholding is enabled - a threshold is reached), it transfers the data to the buffers allocated to the endpoint in application memory. Once the data is transferred, the subsystem updates the status of the received data to the buffer's status quadlet.

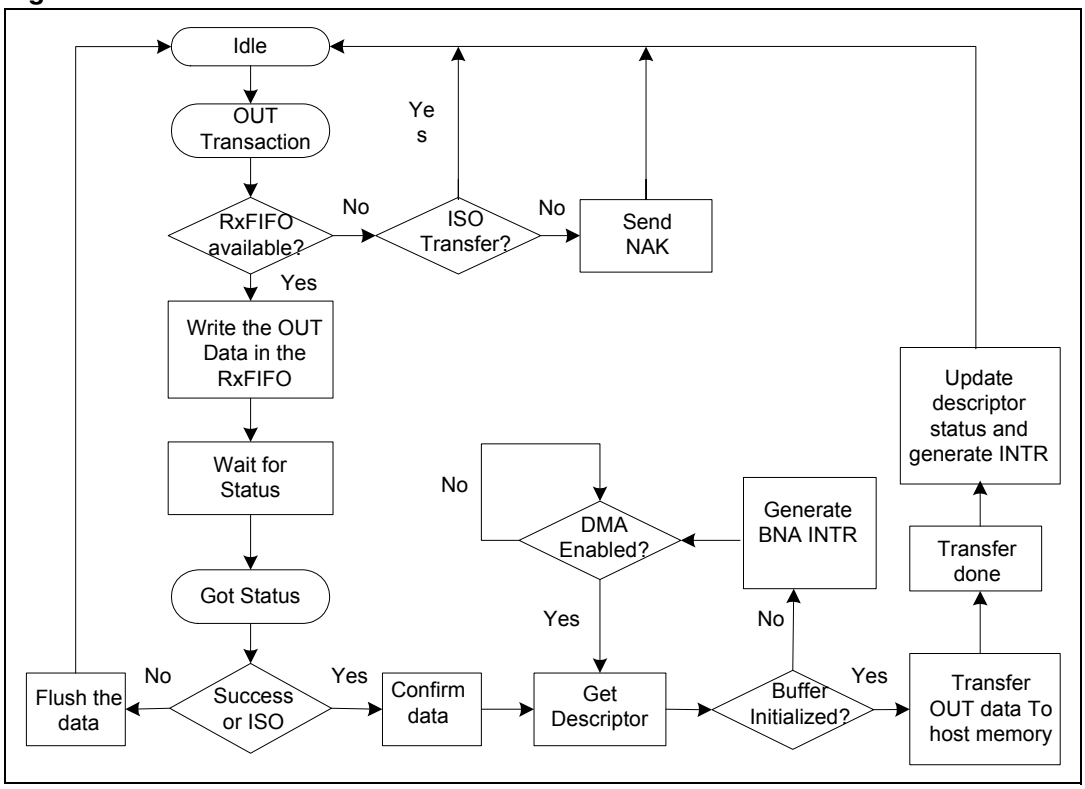
SETUP data is transferred to a 16-byte SETUP buffer. The pointer for this buffer is indicated in the SETUP buffer pointer register, *Endpoint setup buffer pointer register on page 503*). Out data is transferred to the buffers indicated by the descriptor, and the pointer for these descriptors is programmed in the CSRs.

Note: The SETUP data directly addresses the buffers, while regular out data addresses the out data buffers indirectly.

The transaction flow for all out endpoints is similar. The only difference is that isochronous (iso-out) data is tagged with the frame number when the packet is received.

The transaction flow of out data from the USB host to the application memory is given in *Figure 43*.

Figure 43. Out transaction flow in DMA mode



High-bandwidth isochronous (ISO) transfers

In case of out packets (that is, coming from USB Host), each descriptor stores one maximum packet size of data. The data PID associated with the packet is available in the PID field, bits [15:14], of the out data memory buffer status (*OUT data memory structure on page 479*). If the microframe contains three packets, data and the corresponding data PID are stored in three descriptors.

In case of in packets (transmitted to the USB Host), the application creates data of one maximum packet size per descriptor. If the application must send three packets in the microframe, the application must then create three descriptors. Data PID information must be provided in the PID field, bits [15:14], of the in data memory buffer (*IN data memory structure on page 481*).

23.4.4 Slave-only mode

In slave-only implementation, the application acts as an AHB master to read data from or to write data to the memory-mapped subsystem FIFOs, and the UDC-AHB subsystem operates as a AHB slave for both data and CSRs transfers.

The USB host initiates USB traffic and the application responds to all the USB host's commands. In this mode, the UDC-AHB subsystem can only be used in device-type applications, and before any operation the application must completely configure the necessary CSRs. All data transfers are interrupt-driven, except iso-in and interrupt-in transfers, which are periodic.

The slave-only mode is typically implemented either in applications with limited complexity software, or when the subsystem has a dedicated master for data processing.

In operation (Data transfer to USB host)

If the UDC-AHB subsystem receives an in token from an USB Host for a non-isochronous endpoint (such as, bulk, interrupt or control), it checks the associated TxFIFO (*Endpoint FIFO controller (Transmit FIFO controller) on page 467*) for data availability. If data is available, the UDC reads the data from the TxFIFO, otherwise if the TxFIFO does not contain data, the UDC sends an interrupt to the application, and the USB host retries the in token.

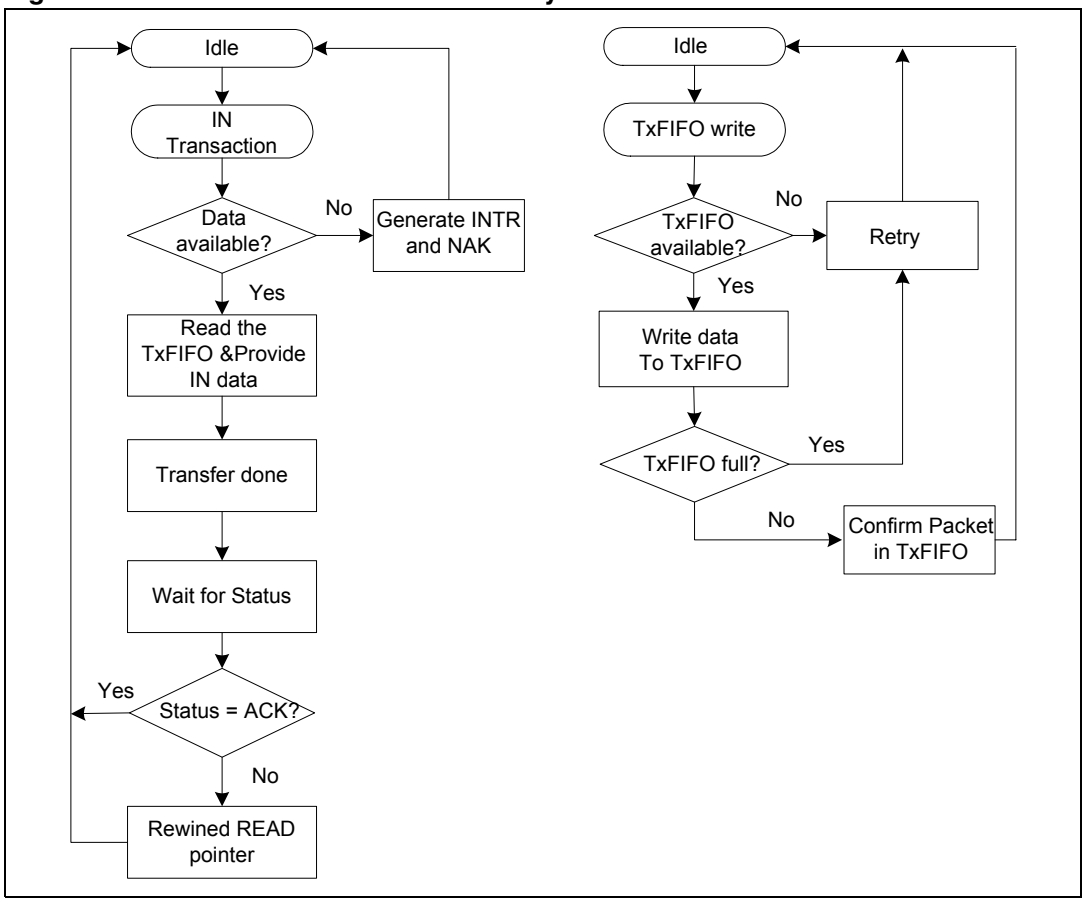
Upon receiving the interrupt, at first the application reads the endpoint interrupt register (*Endpoint interrupt register on page 497*) to determine which endpoint requested the interrupt, and then probes the endpoint status register (*Endpoint status register on page 499*) to determine the interrupt's cause.

Once the application determines that an in token for the endpoint requested the interrupt, it writes the packet directly to the address where the associated TxFIFO is mapped. As soon as the packet data has been completely written into the FIFO, the application performs then a single write to a predefined address (pointed by the write confirmation register, see *Table 396*, of the relevant in endpoint) indicating to the subsystem that the packet transfer is done.

When the USB host retries the in token, the subsystem provides the associated endpoint TxFIFO data to the UDC for transmission to the USB host. The sequence of these events for a non-isochronous (interrupt, bulk, or control) endpoint is shown in *Figure 44*.

Note: The application does not receive status update regarding the packet, because the subsystem must transmit this data. However, the application may flush the packet from the relevant TxFIFO by setting the F bit in the endpoint control register (Endpoint control register on page 498).

Figure 44. In transaction flow in slave-only mode



Isochronous-in endpoints are handled similarly. In this case, the transfer of in data from the application memory to the endpoint FIFO is not initiated by USB Host request tokens, but by the application filling the TxFIFOs as soon as data is available. Isochronous endpoint data must be filled in the TxFIFO only if the buffer is set for the current frame, which the application can determine by reading the current active frame number from the CSR.

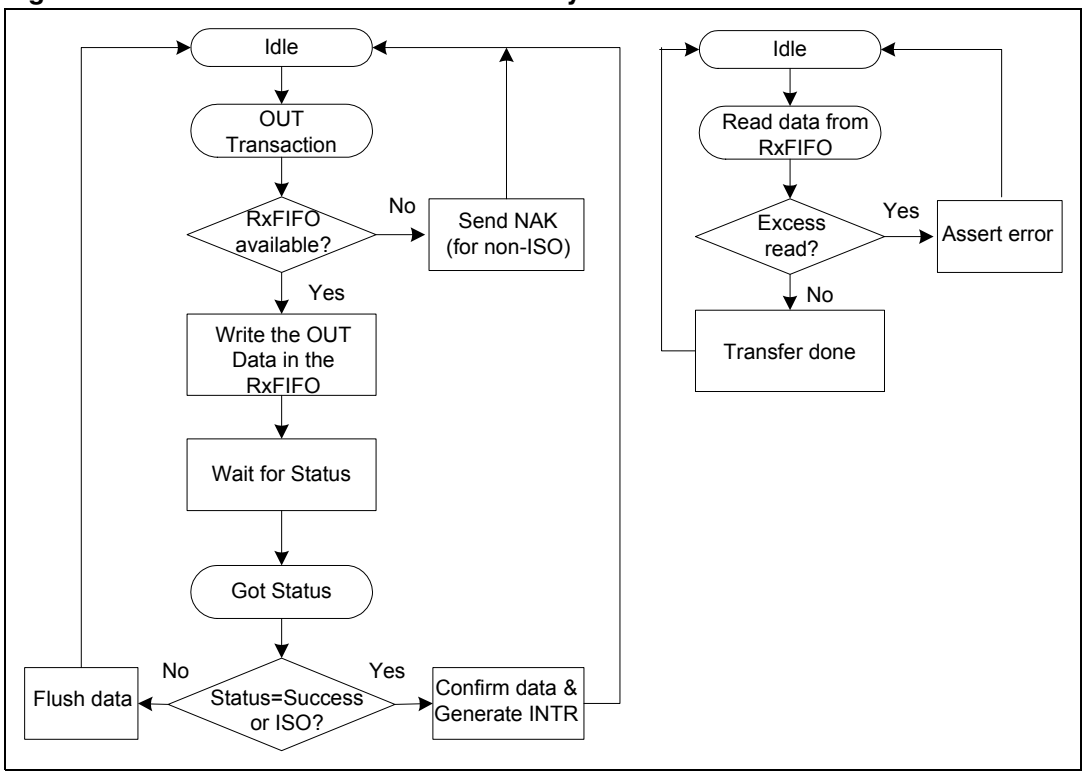
Out operation (Data transfer from USB host)

When the UDC-AHB subsystem receives out data from the USB Host, it transfers the data to the receive FIFO (RxFIFO) within the subsystem - if it has space available for the packet - . If no space is available, the packet is retried. SETUP out packets are stored in a temporary subsystem register before being loaded to the receive FIFO.

Once the packet is transferred to the RxFIFO, the subsystem sends an interrupt to the application for the received packet. Then, the application reads the addressed endpoint's interrupt and status registers (*Endpoint interrupt register on page 497* and *Endpoint status register on page 499*) and it is able to determine the number of bytes received by the UDC-AHB Subsystem in the packet. After that, the application reads from the RxFIFO this number of bytes.

Note: The application reads from the address where the RxFIFO is mapped. The transaction flow of out data from the USB host to the application is shown in Figure 45.

Figure 45. Out transaction flow in slave-only mode



High-bandwidth isochronous (ISO) transfers

In outcase of out packets (that is, coming from USB Host), the data PID received for a high-bandwidth transaction is available in the endpoint buffer size (in) / receive packet frame number (out) register (*Endpoint buffer size and received packet frame number register on page 501*), specifically in the ISO PID field (bit [17:16]).

This 2 bit field indicates the data PID for the current packet available in the receive FIFO. For example, if the USB host sends three packets with a PID sequence of MDATA and DATA2, when the application receives the interrupt for the first packet, the ISO PID field of the register indicates an MDATA PID (2‘b11). After the application reads out the first two maximum-size packets, this register will indicate DATA2 (2‘b10).

In case of in packets (that is, transmitted to the USB host), the transfer is described by the following example flow:

- At first, write the initial data PID (bit [17:16]) in the endpoint buffer size in register (*Endpoint buffer size and received packet frame number register on page 501*). For example, to send three packets in a microframe, write 2’b11 to the ISO PID field (in).
- Write the data one maximum-size packet at a time. After writing one maximum-size packet, perform a confirm cycle.
- Wait for the UDC-AHB subsystem to send the iso in done interrupt, setting the iso in done (bit [23]) field in the endpoint status register (*Endpoint status register on page 499*) after the entire high-bandwidth transaction is completed on the USB.
- To change the packet number to be sent in the next microframe, the application can now modify the iso pid (in) field in the endpoint buffer size in register (*Endpoint buffer size and received packet frame number register on page 501*).

23.5 Data memory structure in DMA mode

23.5.1 SETUP data memory structure

The memory structure for SETUP data is given in *Figure 46*. The 16-byte buffer consists of 4 fields of 32 bits each: the status quadlet (its bit assignments are given in *Table 389*), a reserved one and the 2 last fields for the 8 bytes of SETUP data.

Figure 46. SETUP data memory

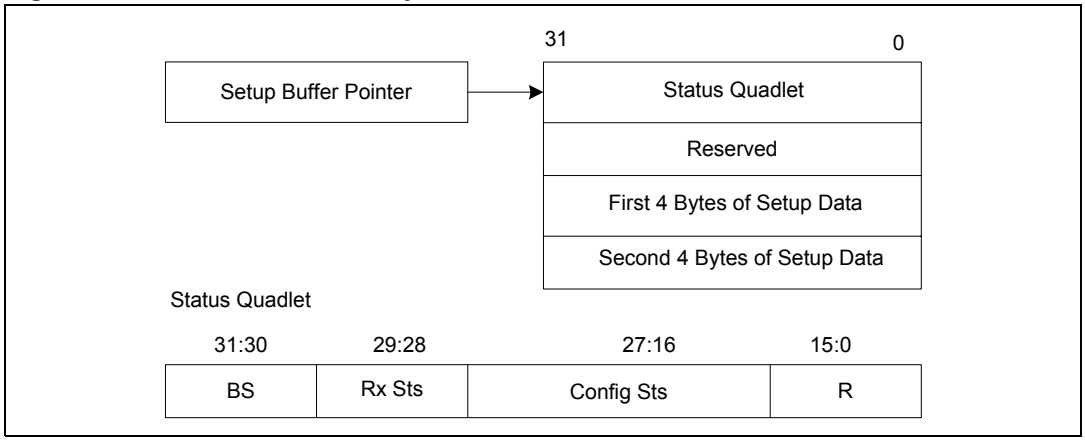


Table 389. SETUP data memory: status quadlet bit assignments

Bit	Name	Description
[31:30]	BS	Buffer status. This 2 bit field reports the status of the SETUP data buffer, according to encoding: – 2'b00 Host ready. = The descriptor is available to be processed by DMA. – 2'b01 DMA busy. = The DMA is still processing the descriptor. – 2'b10 DMA done. = Buffer data transfer completed by DMA. – 2'b11 Host busy. = The application is processing the descriptor.
[29:28]	Rx Sts	Receive status. This 2 bit field reports the status of the received SETUP data (according to encoding), reflecting whether the SETUP data has been correctly received or some errors occurred: – 2'b00 = Success. – 2'b01 = DESERR (descriptor transfer error). – 2'b10 = Reserved. – 2'b11 = BUFFER (data transfer error).
[27:16]	Config Sts	Configuration status. This 12 bit field echoes the status of the current configuration associated with the SETUP packet for a control endpoint. Bits assignments for this field are given: – [27:24] = Configuration number. – [23:20] = Interface number. – [19:16] = Alternate setting number.
[15:00]	Reserved	Read: undefined. Write: should be zero.

23.5.2 OUT data memory structure

All endpoints that support out direction transactions (that is, endpoints receiving data from the USB Host) must implement a memory structure according to the following characteristics:

- Each data buffer must have an associated descriptor which provides the status of the buffer. Indeed, the buffer itself contains only raw data.
- Each buffer descriptor is 4-quadlet length.

The out data memory structure is given in *Figure 47*. *Table 390* reports the bits assignments for out buffer status quadlet.

If the buffer status of the first descriptor is set to host ready (see BS field in *Table 390*), the DMA fetches and processes its data buffer. Otherwise, the DMA skips to the next descriptor until it reaches the end of the descriptor chain.

Figure 47. Out data memory

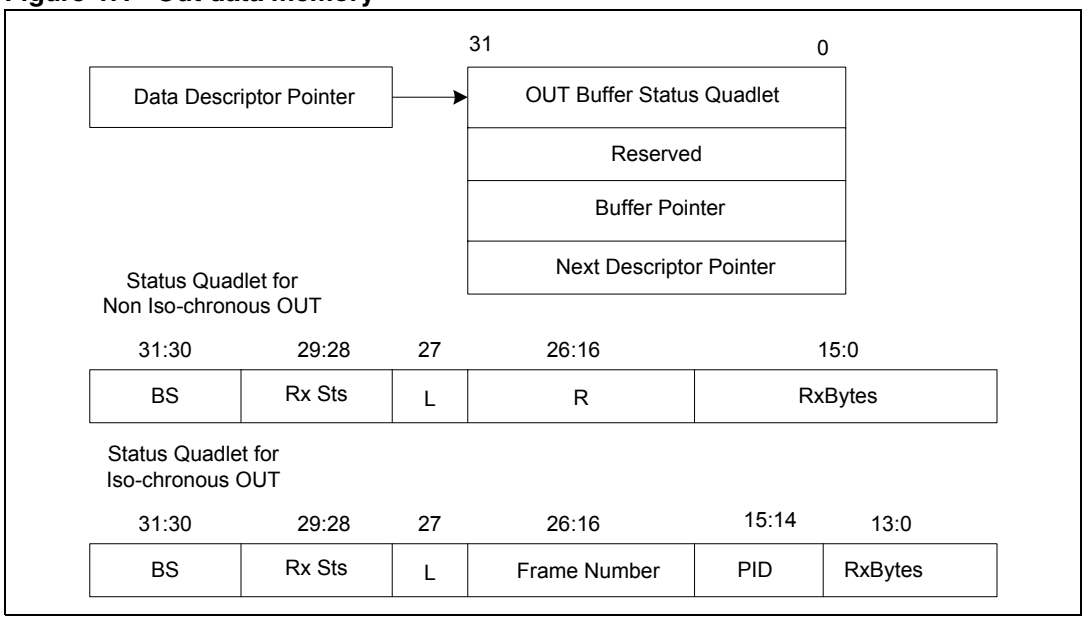


Table 390. Out data memory: buffer status quadlet bit assignments (for Non-Isochronous OUT)

Bit	Name	Description
[31:30]	BS	Buffer status. This 2 bit field reports the status of the out buffer, according to encoding: – 2'b00 Host ready. = The descriptor is available to be processed by DMA. – 2'b01 DMA busy. = The DMA is still processing the descriptor. – 2'b10 DMA done. = Buffer data transfer completed by DMA. – 2'b11 Host busy. = The application is processing the descriptor.
[29:28]	Rx Sts	Receive status. This 2 bit field reports the status of the received out data (according to encoding), reflecting whether the out data has been correctly received or some errors occurred: – 2'b00 = Success. – 2'b01 = DESERR (descriptor transfer error). – 2'b10 = Reserved. – 2'b11 = BUFFER (data transfer error). Note: In particular, a DESERR receive status indicates that the out buffer status is something other than Host ready (that is, BS not equal to 2’b00) during descriptor fetch.
[27]	L	If set, it indicates that this descriptor is the last one of the chain.
[26:16]	Reserved (Non ISO)	Read: undefined. Write: should be zero.
[15:00]	Rx Bytes (Non ISO)	Received number of bytes. Its 16 bit width allows values ranging from 0 to 64 kbytes, depending on the packet size of data received from the USB host.

Table 391. Out data memory: buffer status quadlet bit assignments (for Isochronous OUT)

Bit	Name	Description
[31:30]	BS	Buffer status. This 2 bit field reports the status of the out buffer, according to encoding: – 2'b00 Host ready. = The descriptor is available to be processed by DMA. – 2'b01 DMA busy. = The DMA is still processing the descriptor. – 2'b10 DMA done. = Buffer data transfer completed by DMA. – 2'b11 Host busy. = The application is processing the descriptor.
[29:28]	Rx Sts	Receive status. This 2 bit field reports the status of the received out data (according to encoding), reflecting whether the out data has been correctly received or some errors occurred: – 2'b00 = Success. – 2'b01 = DESERR (descriptor transfer error). – 2'b10 = Reserved. – 2'b11 = BUFFER (data transfer error). Note: In particular, a DESERR receive status indicates that the out buffer status is something other than Host ready (that is, BS not equal to 'b00) during descriptor fetch.
[27]	L	If set, it indicates that this descriptor is the last one of the chain.
[26:16]	Frame Number	11 bit frame number in which the current iso-out packet is received.
[15:14]	PID	ISO received data PID. This 2 bit field indicates the data PID (according to encoding) for the received ISO packet which is contained in the descriptor: – 2'b00 = DATA0 – 2'b01 = DATA1 – 2'b10 = DATA2 – 2'b11 = MDATA Note: The PID field is for HS ISO transactions only. For FS ISO transactions, this field is don't care.
[13:00]	Rx Bytes	Received number of bytes. The value of this field gives the received number of bytes.

23.5.3 IN data memory structure

All endpoints that support in direction transactions (that is, endpoints transmitting data to the USB Host) must implement the memory structure given in *Figure 48*, where each in buffer must be associated to a descriptor. *Table 392* reports the bits assignments for in buffer status quadlet.

The application fills the data buffer, then updates its status in the descriptor, and sets the poll demand bit. Besides, the DMA fetches this descriptor and processes it, moving on in this fashion until it reaches the end of the descriptor chain.

Figure 48. In data memory

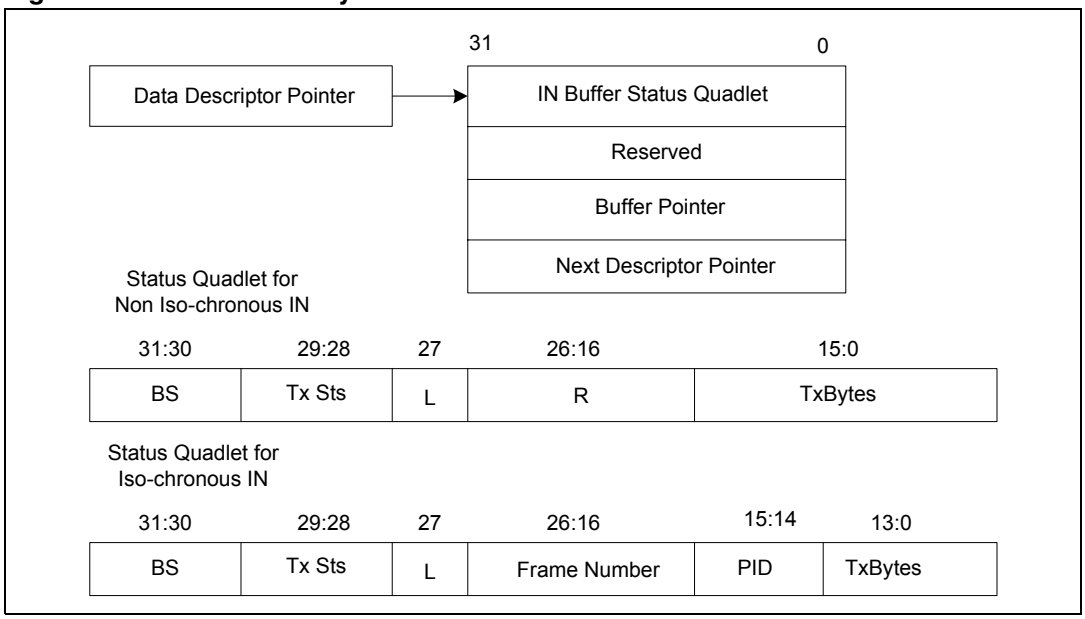


Table 392. In data memory: buffer status quadlet bit assignments (for Non-Isochronous IN)

Bit	Name	Description
[31:30]	BS	Buffer status. This 2 bit field reports the status of the in buffer, according to encoding: – 2'b00, Host ready. The descriptor is available to be processed by DMA. – 2'b01, DMA busy. The DMA is still processing the descriptor. – 2'b10, DMA done. Buffer data transfer completed by DMA. – 2'b11, Host busy. The application is processing the descriptor.
[29:28]	Tx Sts	Transmit status. This 2 bit field reports the status of the transmitted in data (according to encoding), reflecting whether the in data has been correctly transmitted or some errors occurred: – 2'b00 = Success. – 2'b01 = DESERR (descriptor transfer error). – 2'b10 = Reserved. – 2'b11 = BUFFER (data transfer error).
[27]	L	If set, it indicates that this descriptor is the last one of the chain.
[26:16]	Reserved	Read: undefined. Write: should be zero.
[15:00]	Tx Bytes	Number of bytes to be transmitted. Its 16 bit width allows values ranging from 0 to 64 kbytes. for iso in transactions, a maximum value of 16 kbytes is allowed by the 14 bit length of the Tx bytes field.

Table 393. In data memory:buffer status quadlet bit assignments (for Isochronous)

Bit	Name	Description
[31:30]	BS	Buffer status. This 2 bit field reports the status of the in buffer, according to encoding: – 2'b00, Host ready. The descriptor is available to be processed by DMA. – 2'b01, DMA busy. The DMA is still processing the descriptor. – 2'b10, DMA done. Buffer data transfer completed by DMA. – 2'b11, Host busy. The application is processing the descriptor.
[29:28]	Tx Sts	Transmit status. This 2 bit field reports the status of the transmitted in data (according to encoding), reflecting whether the in data has been correctly transmitted or some errors occurred: – 2'b00 = Success. – 2'b01 = DESERR (descriptor transfer error). – 2'b10 = Reserved. – 2'b11 = BUFFER (data transfer error).
[27]	L	If set, it indicates that this descriptor is the last one of the chain.
[26:16]	Frame Number (ISO)	11-bsit frame number in which the current iso-out packet is transmitted. This 11 bit field gives the frame number in which the current iso-in packet is transmitted, according to the following bits assignments: [26:19] = Millisecond frame number. [18:16] = Micro-frame number.
[15:14]	PID	Number of packets per microframe. This 2 bit field indicates the number of packets per microframe for isochronous (iso) in transfers during high-speed (hs) operation. The application must program these bits in the descriptor (and they must be the same for all descriptors of the same microframe) according to encoding, such that the subsystem core returns an isochronous packet with an appropriate data PID per frame: – 2'b00 = 1 – 2'b01 = 1 – 2'b10 = 2 – 2'b11 = 3 Note: The PID field is for HS ISO transactions only. For FS ISO transactions, this field is reserved.
[13:00]	Tx Bytes (ISO)	Number of bytes to be transmitted. The value of this field gives the number of bytes to be transmitted to USB host. In case of non-iso in, its 16 bit length allows values ranging from 0 to 64 kbytes. for iso in transactions, a maximum value of 16 kbytes is allowed by the 14 bit length of the Tx bytes field.

23.6 Operation modes In DMA mode

23.6.1 Packet-per-buffer mode

In *packet-per-buffer* mode (alternate to buffer fill mode, *Buffer fill mode (OUT) on page 484*), the DMA transfers packet by packet to various addresses as indicated by the descriptor, implementing then a true scatter-gather mechanism.

A descriptor update can happen either at the end of each packet transfer or at the end of the descriptor chain only. As a results, the application may be interrupted either after processing each descriptor or at the end of a descriptor chain, respectively. In particular, setting to 1‘b1 the DU bit of the global CSRs’ *Device control register on page 492*, it enables descriptor updating and application interrupt to the software at the end of each packet.

23.6.2 Buffer fill mode (OUT)

Enabling the *buffer fill* mode (setting the bit BF in the global CSRs’ *Device control register on page 492*, the DMA transfers all packets to the large buffer whose address is indicated by the single out data memory structure descriptor. This DMA mode of operation requires fewer memory accesses than packet-per-buffer with descriptor update mode *Packet-per-buffer mode on page 484* above, increasing the throughput.

The DMA controller updates buffer status when a short packet is received, and simultaneously sends an interrupt to the application.

23.6.3 Buffer fill mode (IN)

In case of in transactions, the DMA buffer fill mode can be entered by using the packet-per-buffer mode with only one descriptor indicating:

- The system memory starting address,
- The number of bytes to be transferred to the USB host (the tx bytes can be greater than one packet),
- The L bit set in the status quadlet.

The DMA controller updates buffer status after all data has been transferred to the TxFIFO. The UDC-AHB subsystem then sends an interrupt to the application.

23.6.4 Threshold enable

The *threshold enable* feature is used for transferring packets in the out direction for DMA operation only. There is no threshold enable for the in direction.

Note: In this context, thresholding means emptying the out RxFIFO as soon as it receives a certain number (the threshold value) of 32 bit words.

Thresholding is enabled by setting to 1‘b1 the THE bit in the global CSRs’ *Device control register on page 492*. Moreover, the threshold value is programmed by setting the THLEN 8 bit in the same device control register. As mentioned, the threshold value is the number of 32 bit words (quadlets) that must be received by the RxFIFO before the DMA can start the transfer.

When thresholding is disabled (bit THE set to 1‘b0), then the DMA waits for the complete packet before starting the data transfer. In contrast, if thresholding is enabled, the transfer of the packet to host memory starts before the validity of the packet is assessed. If the packet

is found to be corrupt at the end of the transfer, the descriptor is not updated and the next clean packet overwrites the previous corrupted one. This conceals the USB error from the application.

23.6.5 Burst split enable

When *burst split* is enabled, all AHB transfers (from the DMA to system memory and from system memory to the TxFIFO) are splitted into bursts of a specified length.

The *burst split* is enabled by setting to 1‘b1 the BREN bit in the global CSRs’ *Device control register on page 492*. Moreover, the burst length value is programmed by setting the BRLEN 8 bit in the same device control register. As mentioned, this value indicates the number of 32 bit transfers that should happen in a single burst.

Note: When thresholding and burst splitting are both enabled, the threshold length (THLEN) should be either greater than multiples of burst length (BRLEN) or equal to BRLEN.

23.7 USB plug detect

The USB plug detect block (UPD) allows to detect when an USB host is either connected or disconnected to the USB 2.0 device. This is done through the VBUS pad which is driven high when the USB host is attached. In particular, a plug interrupt is raised by the UPD block when the USB host is attached/detached. This interrupt is putted in OR with the output of the Interrupt Manager block and the output of he OR logic goes to the VIC block.

Two 32 bit RW registers are associated to the UPD block, namely the plug status register and the plug pending register, whose bit assignments are given in *Table 394* and *Table 395*, respectively. These registers can be accessed at the base address 0xE1200000

As soon as the USB Host is connected to the device, the VBUS signal goes high enabling the UPD internal counter, which generates an interrupt 10 ms after the connection. The software routine handling the interrupt reads as 1‘b1 the intend field of the plug pending register (*Table 395*), and as 1‘b1 the state field of the plug status register (*Table 394*): the USB 2.0 PHY reset is then released (phy_rst set to 1‘b0 in plug status register) and the USB 2.0 PHY is placed in normal mode (phy_mode set to 1‘b0 in plug status register).

In contrast, when the USB host is detached, the VBUS signal goes low and after 10 ms an interrupt is generated by the PD block (intpend field set to 1‘b1 in plug pending register). Then, the interrupt handler reads as 1‘b0 the state field of the plug status register, so the reset is asserted (phy_rst set to 1‘b1 in Plug Status register) and the USB 2.0 PHY is placed in non-driving state (phy_mode set to 1‘b1 in plug status register).

Table 394. Plug status register bit assignments

Bit	Name	Reset value	Description
[31:04]	Reserved	-	Read: undefined. Write: should be zero.
[03]	phy_mode	1'h1	USB PHY mode. This bit allows to set the physical terminations of PHY, according to encoding: 1'b0 = Normal (UDC is allowed to drive the USB 2.0 PHY). 1'b1 = Tri-state (the USB 2.0 PHY is in non-driving mode).
[02]	phy_rst	1'h1	USB PHY reset. If set, this bit indicates that the USB PHY is in reset mode, otherwise it is in normal mode.
[01]	state	1'h0	USB host connection state. This RO bit reports the connection status of the USB Host, according to encoding: 1'b0 = Disconnected. 1'b1 = Connection detected.
[00]	enable	1'h0	Plug interrupt. If set, this bit enables an interrupt to be raised when the USB host is attached/detached.

Table 395. Plug pending register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read: undefined. Write: should be zero.
[00]	intpend	1'h0	Plug interrupt. This bit is set when the UPD block generates a plug interrupt. It is cleared when the CPU reads it.

23.8 Programming model

23.8.1 External pin connection

Signal name	Pin	Description
DEV_DP	M1	Device, positive data line
DEV_DM	M2	Device, negative data line
DEV_VBUS	G3	Device, VBUS detection line

23.8.2 Register map

The 32 bit wide CSRs of the UDC-AHB subsystem (*Control and status registers on page 468*) provide a high degree of control, making the device both configurable and scalable. These CSRs can be accessed at the base address 0xE110_0000.

The CSRs can be grouped in two basic categories:

- Global CSRs (listed in *Table 398*), which are specific to the UDC-AHB subsystem.
- Endpoint CSRs (listed in *Table 396* and in *Table 397*), which are specific to a particular endpoint within the UDC-AHB subsystem. Specifically, each endpoint supported by the UDC-AHB subsystem is associated to a set of specific 32 bit CSRs for each direction (in/out).

As explained by the memory map in *Figure 49*, these CSRs are mapped in the 0x0000 to 0x04FC offset address space (with respect to the base address above). Apart from these device-level CSRs, the UDC itself contains other specific CSRs which are mapped in the 0x0500 to 0x07FC offset address space.

Moreover, the FIFOs are mapped at base address 0xE100_0800. Offset addresses from 0x0800 up to a 0x1800 host the data in the RxFIFO (*Receive FIFO controller on page 466*), which are followed by the memory space allocated to TxFIFOs.

Table 396. In endpoint-specific CSRs summary

Endpoint	Name	Offset	Type	Reset value
0	Control	0x0000	RW	32'h0
	Status	0x0004	RO	32'h0
	Buffer size	0x0008	RW	32'h0
	Maximum packet size	0x000C	RW	32'h0
	Reserved	0x0010	-	-
	Data description pointer	0x0014	RW	32'h0
	Reserved	0x0018	-	-
	Write confirmation	0x001C	RW	-
1	As Endpoint 0	0x0020 - 0x003C	As Endpoint 0	
	Reserved	0x0040 - 0x005C		
3	As Endpoint 0	0x0060 - 0x007C	As Endpoint 0	
	Reserved	0x0080 - 0x009C		
5	As Endpoint 0	0x00A0 - 0x00BC	As Endpoint 0	
	Reserved	0x00C0 - 0x00DC		
7	As Endpoint 0	0x00E0 - 0x00FC	As Endpoint 0	
	Reserved	0x0100 - 0x011C		
9	As Endpoint 0	0x0120 - 0x013C	As Endpoint 0	
	Reserved	0x0140 - 0x015C		
11	As Endpoint 0	0x0160 - 0x017C	As Endpoint 0	
		0x0180 - 0x019C		
13	As Endpoint 0	0x01A0 - 0x01BC	As Endpoint 0	
		0x01C0 - 0x01DC		
15	As Endpoint 0	0x01E0 - 0x01FC	As Endpoint 0	

Table 397. Out endpoint-specific CSRs summary

Endpoint	Name	Offset	Type	Reset value
0	Control	0x0200	RW	32'h0
	Status	0x0204	RO	32'h0
	Packet frame number	0x0208	RW	32'h0
	Buffer size	0x020C	RW	32'h0
	SETUP buffer pointer	0x0210	RW	32'h0
	Data description pointer	0x0214	RW	32'h0
	Reserved	0x0218	-	-
	Read confirmation	0x021C	RW	-
	Reserved	0x0220 - 0x023C		
2	As Endpoint 0	0x0240 -0x025C	As Endpoint 0	
	Reserved	0x0260 - 0x027C		
4	As Endpoint 0	0x0280 - 0x029C	As Endpoint 0	
	Reserved	0x02A0 - 0x02BC		
6	As Endpoint 0	0x02C0 - 0x02DC	As Endpoint 0	
	Reserved	0x02E0 - 0x02FC		
8	As Endpoint 0	0x0300 - 0x031C	As Endpoint 0	
	Reserved	0x0320 - 0x033C		
10	As Endpoint 0	0x0340 - 0x035C	As Endpoint 0	
	Reserved	0x0360 -0x037C		
12	As Endpoint 0	0x0380 -0x039C	As Endpoint 0	
	Reserved	0x03A0 -0x03BC		
14	As Endpoint 0	0x03C0 - 0x03DC	As Endpoint 0	
	Reserved	0x03E0 -0x03FC		

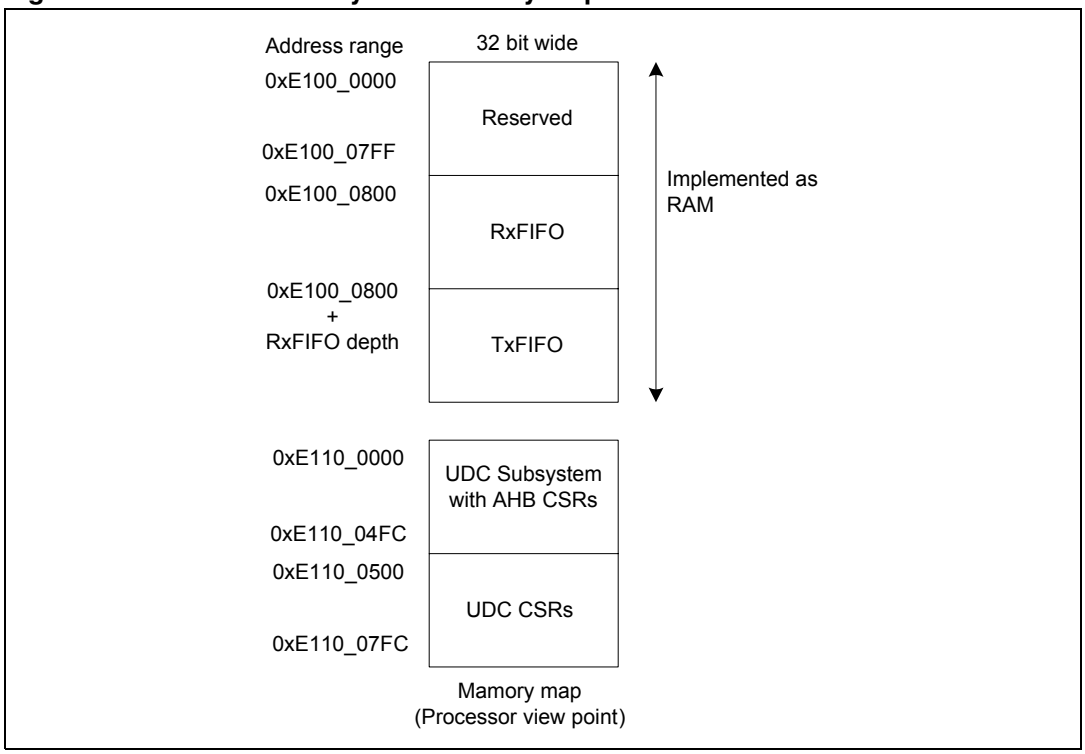
Table 398. Global CSRs summary

Name	Offset	Type	Reset value
Device configuration	0x0400	RW	32'h0
Device control	0x0404	RW	32'h0
Device status	0x0408	RO	32'h0
Device interrupt	0x040C	RW	32'h0
Device interrupt mask	0x0410	RW	32'h0
Endpoint interrupt	0x0414	RW	32'h0
Endpoint interrupt mask	0x0418	RW	32'h0
Reserved	0x041C to 0x04fc		

Table 399. UDCI CSRs summary

Endpoint	Name	Offset	Type	Reset value
	Reserved	0x0500		
0	UDC20 Endpoint register	0x0504	RW	32'h0
1	UDC20 Endpoint register	0x0508	RW	32'h0
2	UDC20 Endpoint register	0x050C	RW	32'h0
3	UDC20 Endpoint register	0x0510	RW	32'h0
4	UDC20 Endpoint register	0x0514	RW	32'h0
5	UDC20 Endpoint register	0x0518	RW	32'h0
6	UDC20 Endpoint register	0x051C	RW	32'h0
7	UDC20 Endpoint register	0x0520	RW	32'h0
8	UDC20 Endpoint register	0x0524	RW	32'h0
9	UDC20 Endpoint register	0x0528	RW	32'h0
10	UDC20 Endpoint register	0x052C	RW	32'h0
11	UDC20 Endpoint register	0x0530	RW	32'h0
12	UDC20 Endpoint register	0x0534	RW	32'h0
13	UDC20 Endpoint register	0x0538	RW	32'h0
14	UDC20 Endpoint register	0x053C	RW	32'h0
15	UDC20 Endpoint register	0x0540	RW	32'h0
	Reserved	0x0544 To 0x07FC		

Figure 49. UDC-AHB subsystem memory map



23.8.3 Register description

23.8.4 Device configuration register

The Device configuration is a RW register which allows to configure the USB 2.0 device.

Table 400. Device configuration register bit assignments

Bit	Name	Reset value	Description
[31:19]	Reserved	-	Read: undefined. Write: should be zero.
[18]	SET_DESC	1'h0	Set descriptor requests support. This bit states how the USB device replies to set Descriptor request, according to encoding: 1‘b0 = A STALL handshake is sent back to the USB Host. 1‘b1 = The SETUP packet passes to the application.
[17]	CSR_PRG	1'h0	Dynamic UDC register programming support. Setting this bit, the application is able to dynamically program the UDC CSRs whenever an interrupt is received for either a set configuration or a set interface request. In this case, the USB device returns a NAK handshake during the status in stage of both the set configuration and set interface requests until the application sets the CSR_DONE bit of the *Device control register on page 492*.

Table 400. Device configuration register bit assignments (continued)

Bit	Name	Reset value	Description
[16]	HALT STATUS	1'h0	Reply to USB host Clear_Feature request for endpoint 0. This bit indicates whether the USB device must respond with either a STALL (bit set to 1‘b1) or an ACK (bit set to 1‘b0) handshake when a Clear_Feature (ENDPOINT_HALT) request for Endpoint 0 has been issued by the USB host.
[15:13]	HS_TIMEOUT CALIB	3‘h0	Timeout counter in HS operation. This 3 bit field indicates the integer number of PHY clocks to the USB device’s timeout counter in high-speed (HS) operation. The application uses this value to increase the timeout value (736 to 848 bit times in HS operation), which depends on the PHY’s delay in generating a line state condition. The default timeout value is 736 bit times.
[12:10]	FS_TIMEOUT CALIB	3‘h0	Timeout counter in FS operation. This 3 bit field indicates the integer number of phy clocks to the USB device’s timeout counter in full-speed (FS) operation. The application uses this value to increase the timeout value (16 to 18 bit times in FS operation), which depends on the PHY’s delay in generating a line state condition. The default timeout value is 16 bit times.
[09]	PHY_ERROR DETECT	1'h0	PHY error detection. Setting this bit, the USB device detects either the phy_rxvalid or the phy_rxactive input signal to be continuously asserted for 2 ms, indicating a PHY error.
[08]	STATUS_1	1'h0	See description.
[07]	STATUS	1'h0	STATUS_1, STATUS These 2 bits together provide an option for the USB device to respond to the USB host with a STALL or an ACK handshake if the USB host has issued a non-zero-length data packet during the status-out stage of a control transfer. Refer to USB device technical documentation for more information.
[06]	DIR	1'h0	UTMI data bus interface direction. This bit states the direction of the UTMI data bus interface, according to encoding: 1‘b0 = Unidirectional. 1‘b1 = Bidirectional.
[05]	PI	1'h0	UTMI PHY interface. This bit indicates the interface size which the UTMI PHY must support, according to encoding: 1‘b0 = 16 bit. 1‘b1 = 8 bit.
[04]	SS	1'h0	If set, the USB device supports Sync frame.
[03]	SP	1'h0	If set, the USB device is self-powered.

Table 400. Device configuration register bit assignments (continued)

Bit	Name	Reset value	Description
[02]	RWKP	1'h0	If set, the USB device is remote wake up capable.
[01:00]	SPD	2'h0	Device speed. These 2 bits give the expected speed the application programs for the USB device, according to encoding: – 2'b00 HS – 2'b01 FS – 2'b10 LS – 2'b11 Reserved However, the actual speed at which the USB device operates depends on the enumerated speed field (ENUM SPD) of the device status register (see *Device status register on page 494*). Note: The UDC11-AHB subsystem uses only the LSB (bit 0) of SPD field, whereas bit 1 is don't care (2'bx0 = LS, 2'bx1 = FS).

23.8.5 Device control register

The device control is a RW register which allows to control (at runtime) the USB 2.0 device after device configuration. The device control register bit assignments are given in *Table 401*.

Table 401. Device control register bit assignments

Bit	Name	Reset value	Description
[31:24]	THLEN (1)	8'h00	Threshold length. This 8 bit field indicates the number (THLEN + 1) of 32 bit entries in the RxFIFO before the DMA can start data transfer (in an out transaction in DMA mode when thresholding is enabled, *Threshold enable on page 484*). The 8'h00 reset value means that only one entry in RxFIFO is enough to start the DMA data transfer.
[23:16]	BRLEN (1)	8'h00	Burst length. This 8 bit field indicates the length of a single burst on the AHB bus as an integer number (BRLEN + 1) of 32 bit data transfers, when burst split features of DMA mode is enabled (*Burst split enable on page 485*). The 8'h00 reset value means then a burst length of (1 · 32) bits.
[15:14]	Reserved	-	Read: undefined. Write: should be zero.
[13]	CSR_DONE	1'h0	CSR programming completion notification. This bit is used by the application to notify the UDC-AHB subsystem that all required CSRs configuration has been completed (bit set to 1'b1). Then, the UDC-AHB subsystem can acknowledge (ACK reply) the current set configuration or set interface command.

Table 401. Device control register bit assignments (continued)

Bit	Name	Reset value	Description
[12]	DEVNAK	1'h0	NAK handshake. setting this bit, the udc-ahb Subsystem returns a NAK handshake to all out endpoints, avoiding then to set the SNAK bit of each endpoint control register (*Endpoint control register on page 498*).
[11]	SCALE	1'h0	Scale down. Setting this bit, the timer values inside the UDC-AHB subsystem are scaled down when running gate-level simulation only, aiming to reduce simulation time. Clear the bit for normal operation.
[10]	SD	1'h0	Soft disconnect. This bit is used by the software application to signal the UDC to soft-disconnect. In particular, setting this bit causes the UDC-AHB Subsystem to enter the disconnected state.
[09]	MODE	1'h0	Operation mode. This bit allows to select the operation mode of the UDC-AHB subsystem (*Theory of operation on page 470*), according to encoding: 1‘b0 = slave-only mode. 1‘b1 = DMA mode.
[08]	BREN (1)	1'h0	Burst transfer to AHB bus enable. Setting this bit, the DMA burst split (*Burst split enable on page 485*) is enabled, and burst length is programmed by the BRLEN field in this register.
[07]	THE (1)	1'h0	Thresholding enable. Setting this bit, the DMA threshold (*Threshold enable on page 484*) is enabled, and a number of quadlets equal to the threshold value (field THLEN in this register) are transferred from the RxFIFO to the memory in an out transaction in DMA mode.
[06]	BF (1)	1'h0	Buffer fill mode enable. Setting this bit, the DMA buffer fill mode (*Buffer fill mode (OUT) on page 484*) is enabled, and the data are transferred into contiguous locations pointed to by the buffer address.
[05]	BE (1)	1'h0	Endianness bit. Setting this bit, the system byte ordering can be changed from little endian (default, BE set to 1‘b0) to big endian. Note: Only data accesses are endian-sensitive (in both slave-only and DMA mode). Descriptor and CSR accesses are always in little endian mode.
[04]	DU (1)	1'h0	Descriptor update. Setting this bit, the DMA updates the descriptor at the end of each packet processed.
[03]	TDE (1)	1'h0	DMA transmission. Setting this bit, the transmit DMA is enabled.

Table 401. Device control register bit assignments (continued)

Bit	Name	Reset value	Description
[02]	RDE (1)	1'h0	DMA receive. Setting this bit, the receive DMA is enabled.
[01]	Reserved	-	Read: undefined. Write: should be zero.
[00]	RES	1'h0	Resuming signaling on the USB. This bit is used by the software application to perform a remote wake-up resume. Setting this bit, the UDC-AHB subsystem signals the USB host to resume the USB bus. however, the application must first set RWKP bit in the device configuration register, (indicating that the UDC-AHB subsystem supports the remote wake-up feature), and the USB host must already have issued a set feature request to enable the device's remote wake-up feature.

1. Field supported in DMA mode only.

23.8.6 Device status register

The device status is a RO register which echoes status information needed to service some of the interrupts. The device status register bit assignments are given in *Table 402*.

Table 402. Device status register bit assignments

Bit	Name	Reset value	Description
[31:18]	TS	14'h0000	Frame number of the received SOF. This 14 bit field indicates the frame number of the received SOF, according to the following rules: High-Speed (HS) operation [31:21] = Millisecond frame number. [20:18] = Microframe number Full-Speed (FS) operation [31:29] = Reserved. [28:18] = Millisecond frame number.
[17]	Reserved	-	Read: undefined. Write: should be zero.
[16]	PHY ERROR	1'h0	PHY Error. This bit is set when either the phy_rxvalid or phy_rxactive input signal is detected to be continuously asserted for 2 ms. It results that the UDC-AHB subsystem goes to the suspend state. When the application serves the early suspend interrupt (ES bit of the device interrupt register, *Device interrupt register on page 495*) it also must check this bit to determine if the early suspend interrupt was generated due to PHY error detection. Note: This bit is reserved for the UDC11-AHB subsystem.

Table 402. Device status register bit assignments (continued)

Bit	Name	Reset value	Description
[15]	RXFIFO EMPTY	1'h0	Receive FIFO empty status. This bit is set as soon as DMA data transfer has been completed and no new packets have been received. In contrast, this bit is cleared after receiving a valid packet from the USB. It is set according to the encoding: – 1‘b0 = Not empty. – 1‘b1 = Empty.
[14:13]	ENUM SPD	2’h0	Enumerated speed. These 2 bits give the speed at which the subsystem comes up after the speed enumeration, according to the encoding: – 2‘b00 = HS. – 2‘b01 = FS. – 2‘b10 = LS. – 2‘b11 = Reserved. If the expected speed is HS (field SPD = 2‘b00 in the device configuration register and the udc-ahb Subsystem is connected to a USB 1.1 host controller, then after speed enumeration, these bits indicates that the subsystem is operating in FS mode (2‘b01). Besides, if SPD states HS again but the UDC-AHB subsystem is connected to a USB 2.0 host controller, then after speed enumeration, these bits indicate that the subsystem is operating in HS mode (2‘b00). Finally, if the expected speed is either LS (SPD = 2‘b10) or FS (SPD = 2‘b01) and the UDC-AHB subsystem is connected to either a USB 1.1 or a USB 2.0 host controller, then after speed enumeration, these bits indicate that the subsystem is operating in either LS mode (2‘b10) or FS mode (2‘b01, respectively).
[12]	SUSP	1'h0	Suspend status. This bit is set according the encoding: – 1‘b0 = Not detected. – 1‘b1 = Detected on USB.
[11:08]	ALT	4’h0	Alternate setting. Please refer to USB standard for more details.
[07:04]	INTF	4’h0	Interface. Please refer to USB standard for more details.
[03:00]	CFG	4’h0	Configuration. Please refer to USB standard for more details.

23.8.7 Device interrupt register

The device interrupt is a RW register whose bits are set when there are system-level events. Indeed interrupts are used by the software application to make system-level decisions. The device interrupt register bit assignments are given in *Table 403*.

Note: After checking this register, the application must clear the interrupt by writing a 1'b1 to the corresponding bit.

Table 403. Device interrupt register bit assignments

Bit	Name	Reset value	Description
[31:07]	Reserved	-	Read: undefined. Write: should be zero.
[06]	ENUM	1'h0	Speed enumeration completed. If set, this bit indicates that speed enumeration is completed. Note: This bit is only used for the UDC20.
[05]	SOF	1'h0	SOF token detected. If set, this bit indicates that a SOF token is detected on the USB.
[04]	US	1'h0	Suspend state detected. If set, this bit indicates that a suspend state is detected on the USB for duration of 3 milliseconds, following the 3 millisecond ES interrupt activity due to an idle state. Note: For the UDC20, there is no suspend interrupt to the application if the PHY clock is suspended via the Suspendm signal.
[03]	UR	1'h0	Reset detected. If set, this bit indicates that a reset is detected on the USB. Note: If the application didn't serve this interrupt, the UDC-AHB subsystem returns a NAK handshake for all transactions except the 8 SETUP packet bytes from the USB host.
[02]	ES	1'h0	Idle state detected. If set, this bit indicates that an idle state is detected on the USB for duration of 3 milliseconds. Note: This interrupt bit is used by the application firmware to finish its job before the subsystem generates a true suspend (US) interrupt (that is, 3 milliseconds after the ES interrupt).
[01]	SI	1'h0	Set interface command. If set, this bit indicates that a set interface command has been received from the USB host. Note: If the application didn't serve this interrupt, the UDC-AHB subsystem returns a NAK handshake for all transactions except the 8 SETUP packet bytes from the USB host.
[00]	SC	1'h0	Set configuration command. If set, this bit indicates that a set configuration command has been received from the USB host. Note: If the application didn't serve this interrupt, the UDC-AHB subsystem returns a NAK handshake for all transactions except the 8 SETUP packet bytes from the USB host.

23.8.8 Device interrupt mask register

The device interrupt mask is a RW register which allows to mask the system levels interrupts. Setting to 1‘b1 the appropriate bit position in the register the designated interrupt is masked.

If masked, the corresponding interrupt signal will not reach the application and its interrupt bit will not be set in the Device Interrupt register (*Device interrupt register on page 495*). The device interrupt mask register bit assignments are given in *Table 404*.

Note: The mask mapping reflects masking on device interrupts register bits e.g. LSB masking through this register signifies SC interrupt of device interrupt register is masked.

Table 404. Device interrupt mask register bit assignments

Bit	Name	Reset value	Description
[31:07]	Reserved	-	Read: undefined. Write: should be zero.
[06:00]	MASK	7’h0	Mask equivalent device interrupt bit.

23.8.9 Endpoint interrupt register

The endpoint interrupt is a RW register whose bits are set when there are endpoint-level events. The MSB 16 bits of the register are allocated to out endpoints, and the LSB 16 bits to in endpoints. The endpoint interrupt register bit assignments are given in *Table 405*.

Note: After checking this register, the application must clear the interrupt by writing a ‘b1 to the corresponding bit.

Table 405. Endpoint interrupt register bit assignments

Bit	Name	Reset value	Description
[31:16]	OUT EP	16’h0000	One bit per out endpoint.
[15:00]	IN EP	16’h0000	One bit per in endpoint.

23.8.10 Endpoint interrupt mask register

The endpoint interrupt mask is a RW register which allows to mask the endpoint-level interrupts. Setting to 1‘b1 the appropriate bit position in the register, the designated interrupt is masked.

If masked, the corresponding interrupt signal will not reach the application and its interrupt bit will not be set in the endpoint interrupt register (*Endpoint interrupt register on page 497*). The endpoint interrupt mask register bit assignments are given in *Table 406*.

Table 406. Endpoint interrupt mask register bit assignments

Bit	Name	Reset value	Description
[31:16]	OUT EP MASK	16’h0000	One bit per out endpoint.
[15:00]	IN EP MASK	16’h0000	One bit per in endpoint.

23.8.11 Endpoint control register

The endpoint control is an endpoint-specific RW register which allows to setup the endpoint as required by the application. The endpoint control register bit assignments are given in *Table 407*.

Note: If the corresponding endpoint is bidirectional (both in and out), there will be two such endpoint control registers.

Table 407. Endpoint control register bit assignments

Bit	Name	Reset value	Description
[31:10]	Reserved	-	Read: undefined. Write: should be zero.
[09]	RRDY	1'h0	Receive ready. This bit is set by the application (at any time), or receiving an out packet, the DMA sends the packet to system memory. This bit is cleared at the end of packet if the descriptor update bit, DU, is set in the *Device control register on page 492*. In contrast, this bit is cleared at the end of payload if the DU bit is set to 1‘b0. If the DMA is busy transferring the data, the application cannot clear this bit.
[08]	CNAK	1'h0	Clear NAK. This bit is used by the application to clear the NAK bit in this register. For example, after a SETUP packet has been decoded as a valid command by the application, then the application must set the CNAK bit to clear the NAK bit. The application also must clear the NAK bit (through CNAK) whenever the subsystem sets it (i.e., the STALL bit in this register is set by the application). Note: The application is allowed to clear this bit only when either the RxFIFO is empty (for single RxFIFO implementation) or when the RxFIFO corresponding to the same logical is empty (for multiple RxFIFO implementation).
[07]	SNAK	1'h0	Set NAK. This bit is used by the application to set the NAK bit in this register. Note: The application must not set the NAK bit for an in endpoint until an in token has been received indicating that the TxFIFO is empty.
[06]	NAK	1'h0	NAK handshake. If set, this bit forces the endpoint to reply to the USB Host with a NAK handshake. Setting and clearing of NAK bit are allowed by SNAK and CNAK bits respectively. For example, after a SETUP packet (preliminarily decoded by the application) has been received by the core, the core sets the NAK bit for all control in and out endpoints. Besides, NAK bit is also set after a STALL response for the endpoint. Note: A SETUP packet is sent to the application regardless of whether the NAK bit is set.

Table 407. Endpoint control register bit assignments (continued)

Bit	Name	Reset value	Description
[05:04]	ET	2'h0	Endpoint type. This 2 bit field gives the endpoint type, according to encoding: – 2'b00 = Control. – 2'b01 = Isochronous (ISO). – 2'b10 = Bulk. – 2'b11 = Interrupt.
[03]	P	1'h0	Poll demand. If set, this bit indicates a poll demand from the application. Note: This bit is reserved for out endpoints only.
[02]	SN	1'h0	Snoop mode. Enabling this bit, the subsystem does not check the correctness of out packets before transferring them to application memory (snoop mode). Note: This bit is reserved for in endpoints only.
[01]	F	1'h0	Flush the TxFIFO. Setting this bit, it flushes the TxFIFO. Note: This bit is reserved for out endpoints only.
[00]	S	1'h0	STALL handshake. If set, this bit forces the endpoint to reply to the USB Host with a STALL handshake. For example, on successful reception of a SETUP packet (preliminarily decoded by the application), the subsystem clears both in and out stall bits, and sets both in and out NAK bits. In case of non-SETUP packets, the subsystem clears either in or out stall bit if a STALL handshake is sent back to the USB Host, and set the corresponding NAK bit. Besides, a STALL handshake for next transactions of a stalled endpoint is returned until the USB Host issues a Clear_Feature command to clear it. Note: The application must check for RxFIFO emptiness before setting the in and out stall bit.

23.8.12 Endpoint status register

The endpoint status is a endpoint-specific RO register which reports the current status of the associated endpoint. The Endpoint Status register bit assignments are given in *Table 408*.

Note: If the corresponding endpoint is bidirectional (both in and out), there will be two such endpoint status registers.

Table 408. Endpoint status register bit assignments

Bit	Name	Reset value	Description
[31:24]	Reserved	-	Read: undefined.
[23]	ISO IN DONE	1'h0	Isochronous in transaction is completed. This bit indicates that an isochronous (iso) in transaction for this endpoint has been completed. the application can use this information to program the iso in data for the next microframe. Note: The iso in done bit is used in slave-only mode, and it is reserved for the UDC11 only.
[22:11]	RX PKT SIZE	12'h000	Receive packet size. This 12 bit field indicates the number of bytes in the current receive packet which the RxFIFO is receiving. In case of incoming SETUP data, this field doesn't report the corresponding number of bytes (8 byte every time), but the configuration status as follows: [22:19] = Configuration number. [18:15] = Interface number. [14:11] = Alternate setting number Note: This field is used in slave-only mode. In DMA mode, the application must check the status from the endpoint data descriptor.
[10]	TDC	1'h0	Transmit DMA completion. If set, this bit indicates that transmit DMA has been completed, transferring a descriptor chain's data to the TxFIFO. After servicing the corresponding interrupt, the application must clear this bit.
[09]	HE	1'h0	Error response on the AHB. If set, this bit indicates that an error response on the AHB occurs, during data transfer, descriptor fetch or descriptor update for this endpoint. After servicing the corresponding interrupt, the application must clear this bit.
[08]	Reserved	-	Read: undefined.
[07]	BNA	1'h0	Buffer not available. This bit is set if the descriptor status is either host busy or DMA done, stating that the descriptor was not ready at the time tried to access. After servicing the corresponding interrupt, the application must clear this bit.
[06]	IN	1'h0	In token reception. If set, this bit states that an in token has been received by the endpoint. After servicing the corresponding interrupt, the application must clear this bit. This bit is reserved in case of out endpoints.

Table 408. Endpoint status register bit assignments (continued)

Bit	Name	Reset value	Description
[05:04]	OUT	2‘h0	Out packet reception. This 2 bit field states that if an out packet has been received by the endpoint. The type of the incoming data is given by encoding: – 2‘b00 = None. – 2‘b01 = Data. – 2‘b10 = SETUP data (8 bytes). – 2‘b11 = Reserved. In order to clear these bits, the application must write the same values.
[03:00]	Reserved	-	Read: undefined.

23.8.13 Endpoint buffer size and received packet frame number register

This is a dual-function endpoint-specific RW register which gives either the buffer size or the TxFIFO associated to an in endpoint, or the frame number in which a packet is received an out endpoint (useful to handle ISO traffic).

Table 409. Endpoint buffer size/received packet frame number register bit assignments

Bit	Name	Reset value	Description
[31:18]	Reserved	-	Read: undefined. Write: should be zero.

Table 409. Endpoint buffer size/received packet frame number register bit assignments (continued)

Bit	Name	Reset value	Description
[17:16]	ISO PID (IN/OUT)	2'h0	Initial data PID to be sent for a high-bandwidth ISO transaction. For IN These 2 bits indicate the initial data PID to be transmitted for an high-bandwidth ISO transaction, according to encoding: – 2'b00 = DATA0. – 2'b01 = DATA0. – 2'b10 = DATA1. – 2'b11 = DATA2. For OUT, These 2 bits indicate the initial data PID of the packet received (that is, available in the RxFIFO) for an high-bandwidth ISO transaction, according to encoding: – 2'b00 = DATA0. – 2'b01 = DATA1. – 2'b10 = DATA2. – 2'b11 = MDATA. Note: The ISO PID field is used in slave-only mode, and it is reserved for the UDC11.
[15:00]	BUFF SIZE (IN)	16'h0000	For IN Buffer size required for this endpoint. This 16 bit field represents the size of the buffer associated to that in endpoint as an integer number of 32 bit words. Resulting flexibility in buffer size allows the application to cope with interface or configuration changes. For OUT Frame number in which the packet is received. This 16 bit field states the frame number in which an incoming packet has been received by the RxFIFO for that out endpoint, as follows: -High-Speed (HS) operation [15:14] Reserved. [13:3] Millisecond frame number. [2:0] Micro-frame number. Full-Speed (FS) operation [15:11] Reserved. [10:0] Millisecond frame number

23.8.14 Endpoint maximum packet size and buffer size register

This is an endpoint-specific RW register which gives both the buffer size in the RxFIFO associated to an out endpoint, and the maximum packet size an endpoint (both in and out) should support. This maximum packet size is used to calculate whether the RxFIFO has sufficient space to accept a packet. The register bit assignments are given in *Table 410*.

Note: When the maximum packet size for a specific endpoint is changed, the application must also properly set the UDC register space.

Table 410. Endpoint maximum packet size/buffer size register bit assignments

Bit	Name	Reset value	Description
[31:16]	BUFF SIZE	16'h0000	Buffer size required for this endpoint. This 16 bit field represent the size of the buffer in the RxFIFO associated to that out endpoint as an integer number of 32 bit words. Resulting flexibility in buffer size allows the application to cope with interface or configuration changes.
[15:00]	MAX PKT SIZE	16'h0000	Maximum packet size for the endpoint (in bytes).

23.8.15 Endpoint setup buffer pointer register

The endpoint SETUP buffer pointer is an endpoint-specific RW register which contains the SETUP buffer pointer used in SETUP commands. The endpoint SETUP buffer pointer register bit assignments are given in *Table 411*.

Note: 1 The endpoint SETUP buffer pointer register is used in DMA mode only.

2 The endpoint SETUP buffer pointer register is applicable to control endpoints only, whereas it is reserved for all other endpoints.

Table 411. Endpoint SETUP buffer pointer register bit assignments

Bit	Name	Reset value	Description
[31:00]	SUBPTR	32'h0000	SETUP buffer pointer.

23.8.16 Endpoint data description pointer register

The endpoint data description pointer is an endpoint-specific RW register which contains data descriptor pointer. Both in and out endpoints have a data descriptor pointer each (32 bit wide).

The endpoint data description pointer register bit assignments are given in *Table 412*.

Table 412. Endpoint data description pointer register bit assignments

Bit	Name	Reset value	Description
[31:00]	DESPTR	32'h0000	Data descriptor pointer.

23.8.17 UDC20 endpoint register

The UDC20 endpoint register is used by SW to describe the endpoint characteristics.

Table 413. Endpoint register bit assignments

Bit	Name	Reset value	Description
[31:30]	Reserved	-	Read: undefined. Write: should be zero.
[29:19]	MaxPackSize	11'h000	Maximum endpoint packet size.
[18:15]	AltSetting	4'h0	Alternate setting to which this endpoint belongs.
[14:11]	InterfNumber	4'h0	Interface number to which this endpoint belongs.

Table 413. Endpoint register bit assignments (continued)

Bit	Name	Reset value	Description
[10:07]	ConfNumber	4'h0	Configuration number to which this endpoint belongs.
[06:05]	EPType	2'h0	Endpoint type. The possible options are: – 2'b00: Control – 2'b01: Isochronous – 2'b10: Bulk – 2'b11: Interrupt
[04]	EPDir	1'h0	Endpoint direction. The possible options are: – 1'b0: out – 1'b1: in
[03:00]	EPNumber	4'h0	Logical endpoint number

Note: The UDC 2.0 registers won't be accessible in SUSPEND mode as the UDC2.0 registers operate on PHY clock which is disabled in the SUSPEND mode.

24 HS_Media independent interface (MII)

24.1 Overview

Within its high-speed (HS) connection subsystem, the device provides an **Ethernet MAC 10/100 Universal** (commonly referred as MAC-UNIV), enabling to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2002 standard.

In particular, MAC-UNIV in the device is configured to offer an AHB-interfaced native DMA (also referred as MAC-AHB) over a MAC core. Main features offered by MAC core are listed in *Section 24.1.1*. The MAC-AHB allows transfer data by DMA with system memory through AHB interface. Main features of MAC-AHB are summarized in *Section 24.1.2*.

24.1.1 MAC core main features

- It supports the default Media Independent Interface (MII) defined in the IEEE 802.3 specifications.
- It supports 10/100 Mbps data transfer rates with the PHY interfaces above.
- It supports both half-duplex and full-duplex operation. In half-duplex operation, CSMA/CD protocol is provided.
- Programmable frame length to support both standard and Jumbo Ethernet frames with size up to 16 Kbytes.
- A variety of flexible address filtering modes are supported.
- A set of control and status registers (CSRs) to control MAC Core operation.
- Complete network statistics with RMON Counters (MMC, MAC Management Counters).

24.1.2 MAC-AHB main features

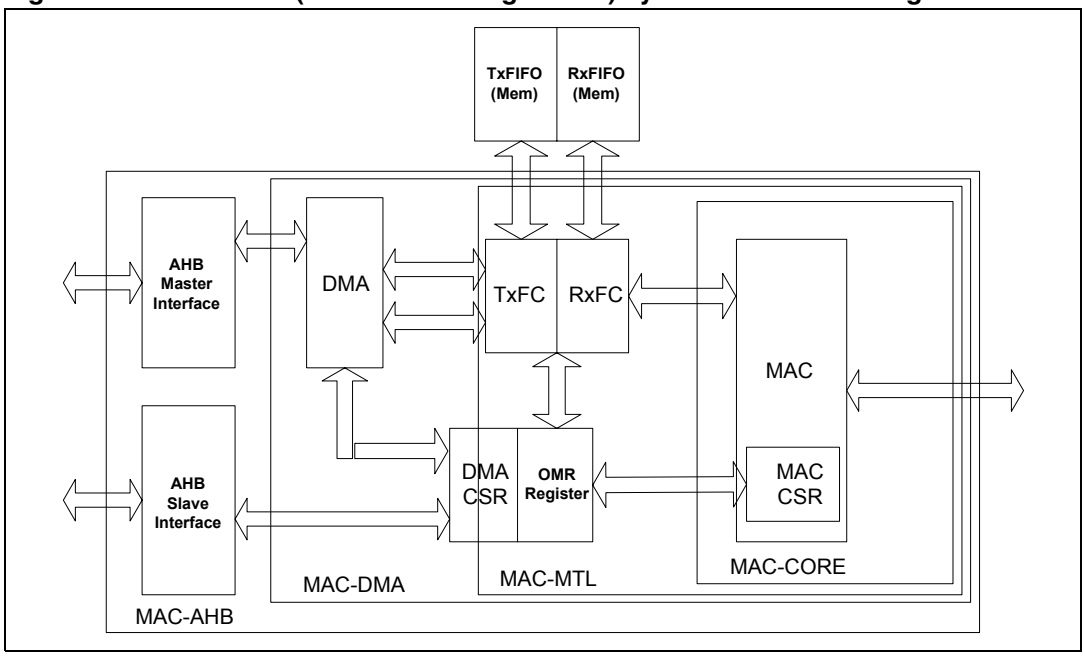
- DMA implements dual-buffer (ring) or linked-list (chained) descriptor chaining.
- A set of CSR's to control DMA operation.
- An AHB slave acting as programming interface to access all CSR's, for both DMA and MAC Core subsystems.
- An AHB master for data transfer to system memory.
- It support both big-endian and little-endian.
- Power Management Module (PMT) with Remote Wake-up and Magic Packet frame processing options;

24.2 Functional description

24.2.1 Block diagram

Figure 50 shows the system-level block diagram of MAC-UNIV in MAC-AHB configuration.

Figure 50. MAC-UNIV (MAC-AHB configuration) system-level block diagram



24.3 Main functions description

24.3.1 AHB slave interface

The AHB Slave Interface block allows the host CPU to access all the DMA and MAC control and status registers (CSRs).

32 bit, 16 bit and 8 bit write/read transfers to CSRs are all supported by AHB Slave Interface, but 32 bit access to CSRs are recommended to avoid any software synchronization problems.

24.3.2 AHB master interface

In MAC-AHB configuration, the MAC-UNIV transfers data by DMA to system memory through the AHB master interface.

In particular, the AHB master interfaces with the DMA controller and converts the internal DMA request cycles into AHB cycles. Both fixed burst length (SINGLE, INCR4, INCR8, INCR16) and unspecified burst length (SINGLE, INCR) transfer types are supported.

Note: 1 The DMA Controller should request an AHB Burst Read transfer only when it has the capability of accepting the received burst data completely. Indeed, data read from AHB is always pushed into the DMA without any delay.

2 The DMA Controller should request an AHB Burst Write transfer only when it has the sufficient data to transfer the burst completely. Indeed, the AHB interface assumes that it always has data available to push into the AHB bus.

24.3.3 DMA controller

A native DMA is available within the MAC-AHB, and its DMA controller interfaces both with the host through the AHB interface and with the MAC core (*Section 24.1.1*).

The DMA controller has Independent Transmit and Receive engines. The Transmit Engine transfers data from system memory (through the AHB master interface) to MAC core, while the Receive Engine transfers data from the MAC core to the system memory (through AHB master interface).

Apart from DMA CSRs, the DMA controller communicates with the host using both **descriptor lists** and **data buffers.**

The *descriptor lists* are used by the *DMA Controller* to efficiently move data from source to destination with minimal host CPU involvement. The descriptor lists (detailed in *Section 24.4*) reside in the host physical memory space, and they act as pointers to the data buffer (each descriptor can point to two buffers maximum).

A data buffer consists of an entire frame or part of a frame, but cannot exceed a single frame. Data buffers reside in the host physical memory space, and they are used by the *DMA Controller* to write to **(Receive Buffer)** and read from **(Transmit Buffer)** frames which have been received or have to be transmitted, respectively.

Note: Only data are contained in data buffer, whereas buffer status is maintained in the relevant descriptor.

24.3.4 Transmit and receive FIFO

The Transmit FIFO (TxFIFO) buffers data read from system memory by the DMA, before transmission by the MAC core.

Similarly, the receive FIFO (RxFIFO) is intended to store the frames received from the ethernet until they are transferred to system memory by the DMA.

These are asynchronous FIFOs, as they transfer data between the application clock domain and the MAC line clock.

24.3.5 MAC management counters

The MMC (MAC Management Counters) Module provides a mechanism compliant with the standard RMON (Remote Networking Monitoring) specification. This standard defines a set of statistics and functions that can be exchanged between RMON-compliant console systems and network probes.

The counters in the MMC module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers (listed in the *Table 423: MAC-UNIV MAC global registers summary* in the section Register Map) for gathering statistics on the received and transmitted frames. These include a control register for controlling the behaviour of the registers, two 32 bit registers containing interrupts generated (receive and transmit) and two 32 bit registers containing mask for the interrupt register (receive and transmit).

The organization of these registers is shown in the section MMC Control Register and following ones. The MMCs are accessed using transactions, in the same way the CSR address space is accessed.

24.3.6 Power management module (PMT)

This section describes the power management (PMT) mechanism as supported by the MAC. PMT supports the reception of network (remote) wake-up frames and Magic Packet frames. PMT does not perform the clock gate function, but generates interrupts for wake-up frames and Magic Packets received by the MAC. The PMT block sits on the receiver path of the MAC and is enabled with remote wake-up frame enable and Magic Packet enable. These enables are in the **PMT Control and Status Register (Register11, MAC)** and are programmed by the Application.

24.4 DMA descriptors

The DMA Controller operates two descriptor lists, one for transmission and one for reception. The base address of each list is written into DMA Register3 (Receive Descriptor List Address, *Section 24.7.6*) and Register4 (Transmit Descriptor List Address, *Section 24.7.7*), respectively.

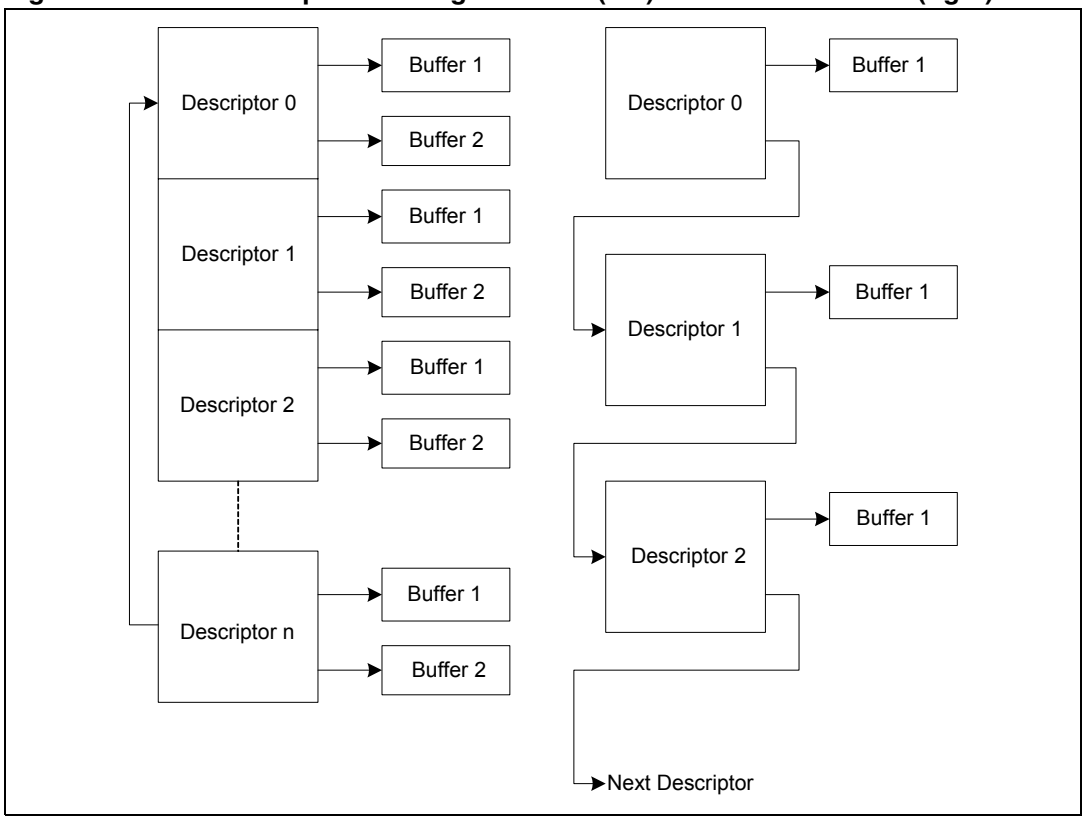
Each descriptor list (both for transmission and for reception) consists of a set of descriptor, and each descriptor is provided with the following fields:

- The status of the received/transmitted frames together with descriptor ownership information,
- A set of control bits,
- The byte-count of the two pointed data buffers,
- The address pointers of the two data buffers.

Depending on the contents of the 2nd data buffer of the last descriptor in the list, two different lists can result (as depicted in *Figure 51.*)

- **A ring structure**: the list is implicitly forward linked, each descriptor points to two data buffers, and the last descriptor points back to the first entry of the list;
- **A chain structure**: the list is explicitly forward linked by using the 2nd address pointer of each descriptor to point the next descriptor in the list.

Figure 51. DMA descriptor list: ring structure (left) and chain structure (right).



Note: The descriptor addresses must be aligned to the bus-width used (32/64/128 bit buses).

24.4.1 Transmit descriptors

According to *Figure 52.*, there are four different Transmit Descriptors:

- Transmit Descriptor 0, **TDES0** (*Table 414*): it contains the status of the transmitted frame and the descriptor ownership information along with control bits for controlling the descriptor structure and the frame being transferred;
- Transmit Descriptor 1, **TDES1** (*Table 415*): it contains the data buffer sizes;
- Transmit Descriptor 2, **TDES2** (*Table 416*): it contains the address pointer to the first data buffer in the descriptor;
- Transmit Descriptor 3, **TDES3** (*Table 417*): it contains the address pointer to the second data buffer in the descriptor or the next descriptor (in case of chained structure)

Figure 52. DMA descriptor format (Transmit Descriptor, 32 bit)

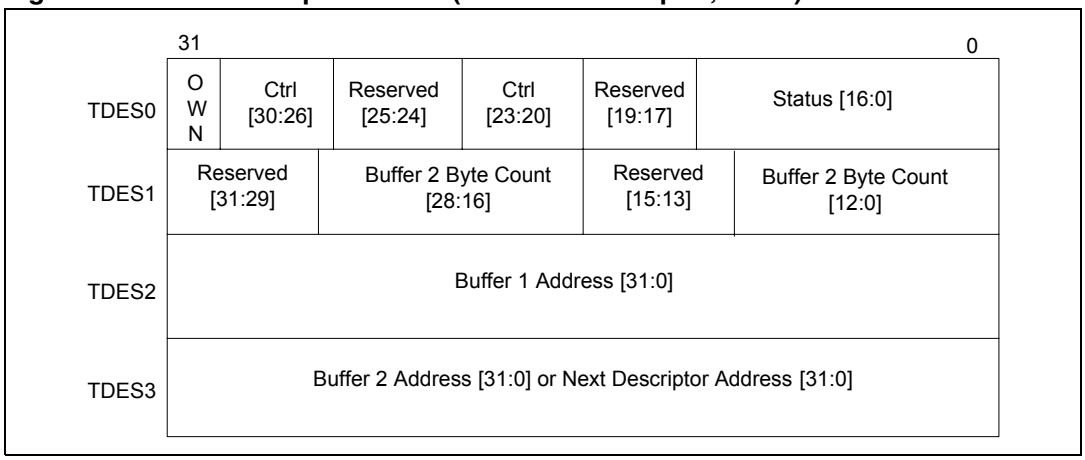


Table 414. Transmit descriptor 0 (TDES0)

Bit	Name	Description
[31]	OWN	Own Bit. If set, indicates that the descriptor is owned by the DMA of MAC. If cleared, the descriptor is owned by the host.
[30]	IC	Interrupt on Completion. Setting this bit, the TI bit of the Status register (DMA Register5) is set after the present frame has been transmitted.
[29]	LS	Last Segment. If set, it indicates that the buffer contains the last segment of the frame.
[28]	FS	First Segment. If set, it indicates that the buffer contains the first segment of the frame.
[27]	DC	Disable CRC. Setting this bit, the MAC doesn't automatically add padding to a framer shorter than 64 bytes.
[26]	DP	Disable Padding. Setting this bit, the MAC doesn't automatically add padding to a frame shorter than 64 bytes. If cleared, the DMA automatically adds padding and CRC to a frame shorter that 64 bytes and the CRC field is added despite the state if the DC bit (TDES0[27]). Valid only if (TDES0[28]) is set.
[25]	TTSE	Reserved
[24]	Reserved	-
[23:22]	CIC	Checksum Insertion Control. These bits control the checksum calculation and insertion. Bit encodings are as shown below. 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 calculated in hardware. 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware.
[21]	TER	Transmit End of Ring. If set, it indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring structure.

Table 414. Transmit descriptor 0 (TDES0) (continued)

Bit	Name	Description
[20]	TCH	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. TDES0[21] takes precedence over TDES0[20].
[19:18]	Reserved	-
[17]	TTSS	Transmit Time Stamp Status. This field is used as a status bit to indicated that a time stamp was captured for the described transmit frame. When this bit is set, TDES2 and TDES3 have a time stamp value captured for the transmit frame. This field is only valid when the descriptor's Last Segment Control bit (TDES0[29]) is set.
[16]	IHE	IP Header Error. When set, this bit indicates that the MACMAC transmitter 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. 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. For IP4 frames, an error status is also indicated if the Header Length field has a value less than 0x5.
[15]	ES	Error Summary. It is the logical OR of the following bits of this descriptor. [1], [2], [8], [9], [10], [11], [13] and [14].
[14]	JT	Jabber Timeout. If set, it indicates that the MAC transmitter has experienced a jabber timeout.
[13]	FF	Frame Flushed. If set, it indicates that the DMA flushed the frame due to a software flush command given by the CPU.
[12]	IPE	IP Payload Error When set, this bit indicates that MAC transmitter detected
[11]	LC	Loss of Carrier. If set, it indicates that loss of carrier occured during frame transmission. This is valid only for the frames transmitted without collision when the MAC operates in Half-Duplex mode.
[10]	NC	No Carrier. If set, it indicates that the carrier sense signal
[09]	LC	Late Collision. If set, it indicates that the frame transmission was aborted due to a collision after the collision window. This bit is not valid if the UnderFlow Error bit is set.
[08]	EC	Excessive Collision. If set, it indicates that the frame transmission was aborted after 16 successive collisions while attempting to transmit the current frame. Note: if the DR (Disable Retry) bit in the MAC Configuration register (Register 0) is set, the EC bit is set after the first collision and the transmission is aborted.
[07]	VF	VLAN Frame. If set, it indicates that the transmitted frame was a VLAN-type frame.
[06:03]	CC	Collision Count. This 4 bit counter value reports the number of collisions occuring before the frame was transmitted. Note: The count is not valid when the EC bit (TDES0[8]) is set.

Table 414. Transmit descriptor 0 (TDES0) (continued)

Bit	Name	Description
[02]	ED	Excessive Deferral. If set, it indicates that the transmission has ended because of excessive deferral of over 24288 bit times (155680 bit times in Jumbo Frame Enabled Mode), if the DC (Deferral Check) bit in the MAC Configuration register (Register 0) is set.
[01]	UF	Underflow Error. If set, it indicates that the transmission has ended because data arrived late from the host memory. This means that DMA encountered an empty Transmit Buffer while transmitting the frame.
[00]	DB	Deferred Bit. If set, it indicates that the MAC defers before transmission because of the presence of carrier. Valid in half-duplex mode only.

Table 415. Transmit descriptor 1 (TEDS1)

Bit	Name	Description
[31:29]	Reserved	-
[28:16]	TBS2	Transmit Buffer 2 Size. These 13 bit field reports the size (in bytes) of the second data buffer. Note: This field is not valid if TDES0[20] is set.
[15:13]	Reserved	-
[12:00]	TBS1	Transmit Buffer 1 Size. These 13 bit field reports the size (in bytes) of the first data buffer. Note: If TBS1 value is 13h'0, then the DMA ignores this buffer and uses the 2nd buffer or next descriptor (depending on TCH bit value).

Table 416. Transmit descriptor 2 (TDES2)

Bit	Name	Description
[31:00]	Buffer 1 Address Pointer	This field indicates the physical address of the 1st data buffer pointed to by this descriptor.

Table 417. Transmit descriptor 3 (TDES3)

Bit	Name	Description
[31:00]	Buffer 2 Address Pointer	This field indicates the physical address of the 2nd data buffer pointed to by this descriptor, if descriptor chaining is used. If TDES1[24] bit is set, then this field contains the pointer to the physical memory when the next descriptor is present.

24.4.2 Receive descriptors

As for Transmit Descriptors above, there are four different Receive Descriptors:

- Receive descriptor 0, **RDES0** (*Table 418*): it contains the status of the received frame, the frame length and the descriptor ownership information.
- Receive descriptor 1, **RDES1** (*Table 419*): it contains the data buffer sizes and other bits which controls the descriptor structure (chain/ring).
- Receive descriptor 2, **RDES2** (*Table 420*): it contains the address pointer to the first data buffer in the descriptor.
- Receive descriptor 3, **RDES3** (*Table 421*): it contains the address pointer to the second data buffer in the descriptor or the next descriptor (in case of chained structure).

Figure 53. DMA descriptor format (Receive Descriptor, 32 bit)

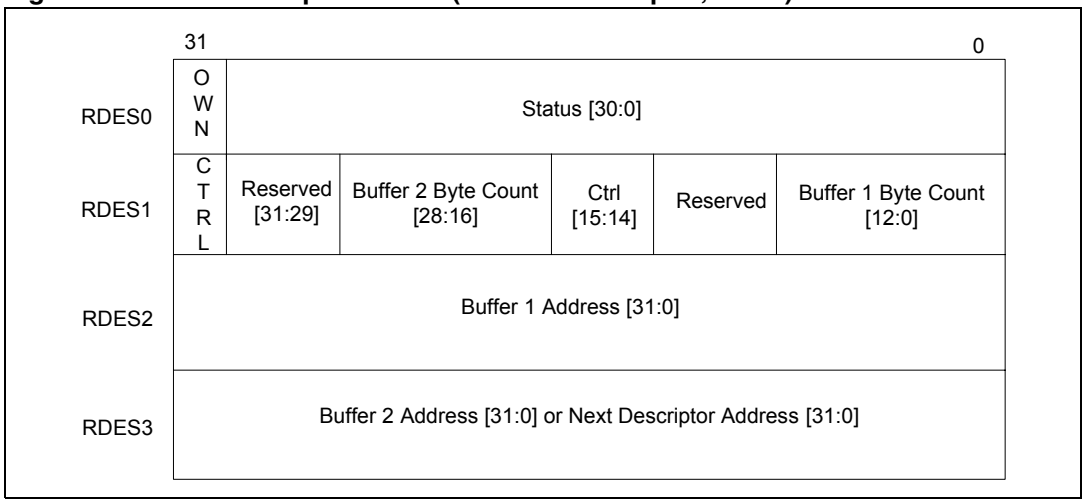


Table 418. Receive descriptor 0 (RDES0)

Bit	Name	Description
[31]	OWN	**Own Bit.** If set, it indicates that the descriptor is owned by the DMA of MAC. If cleared, the descriptor is owned by the host.
[30]	AFM	**Destination Address Filter Fail.** If set, it indicates a frame that failed in the DA filter in the MAC core.
[29:16]	FL	**Frame Length.** These 14 bit field reports the byte length of the received frame (including CRC and the 2 bytes appended to the frame when IP checksum calculation is enabled and the received frame is not a MAC control frame).
[15]	ES	**Error Summary.** It is the logical OR of the following bits of this descriptor: [1], [3], [4], [6], [10], [11] and [14].
[14]	DE	**Descriptor Error.** If set, it 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.
[13]	SAF	**Source Address Filter Fail.** If set, it indicates a frame that failed in the SA filter in the MAC core.
[12]	LE	**Length Error.** If set, it states that the actual length of the frame received and that the length/type field does not match.

Table 418. Receive descriptor 0 (RDES0) (continued)

Bit	Name	Description
[11]	OE	**Overflow Error.** If set, it indicates that received frame was damaged due to buffer overflow in MAC core.
[10]	VLAN	**Van Tag.** If set, it indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the MAC core.
[09]	FS	**First Descriptor.** If set, this descriptor contains the first buffer of the frame.
[08]	LS	**Last Descriptor.** If set, it indicates that the buffers pointed to by this descriptor are the last buffers of the frame.
[07]	IPC checksum error	If set, it indicates that the 16 bit IP header checksum calculated by the MAC core did not match the received checksum bytes.
[06]	LC	**Late Collision**. If set, it indicates that late collision has occurred while receiving the frame in half-duplex mode.
[05]	FT	**Frame type**. If set, it indicates that the received frame is an Ethernet-type frame, whereas the received frame is an IEEE802.3 frame.
[04]	RWT	Receive watchdog timeout. If set, it indicates that the receive watchdog timer has expired while receiving the current frame and the current frame is truncated after the watchdog timeout.
[03]	RE	Receive Error.
[02]	DE	**Dribble Bit Error.** If set, it indicates that the received frame has a non-integer multiple of bytes (odd nibbles). Valid only in MII mode.
[01]	CE	CRC Error.
[00]	Rx MAC address	If set, it indicates that the Rx MAC address value (register1 to register15) matched the DA field of the frame. If cleared, it indicates that Rx MAC Address0 matched the DA field.

Table 419. Receive descriptor 1 (RDES1)

Bit	Name	Description
[31]	DIC	Disable Interrupt on Completion Setting this bit will prevent the setting of the RI bit of the Status register (*Table 422*) for the received frame that ends in the buffer pointer to by this descriptor. This, in turn, will disable the assertion of the interrupt to the host due to RI.
[30:29]	Reserved	-
[28:16]	RBS2	**Receive Buffer 2 Size.** These 13 bit field reports the size (in bytes) of the second data buffer. **Note:** The RBS2 value must be a multiple of 4/8/16 depending on the width of the bus otherwise the resulting behaviour is undefined.
[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].

Table 419. Receive descriptor 1 (RDES1) (continued)

Bit	Name	Description
[13]	Reserved	-
[12:00]	RBS1	**Receive Buffer 1 Size.** These 11 bit field reports the size (in bytes) of the first data buffer. **Note:** The RBS1 value must be a multiple of 4/8/16 depending on the bus width otherwise the resulting behaviour is undefined. **Note:** If RBS1 value is 13'h0, then the DMA ignores this buffer and uses the 2nd buffer or next descriptor (depending on RCH bit value).

Table 420. Receive descriptor 2 (RDES2)

Bit	Name	Description
[31:00]	Buffer 1 Address Pointer	This field indicates the physical address of the 1st data buffer pointed to by this descriptor.

Table 421. Receive descriptor 3 (RDES3)

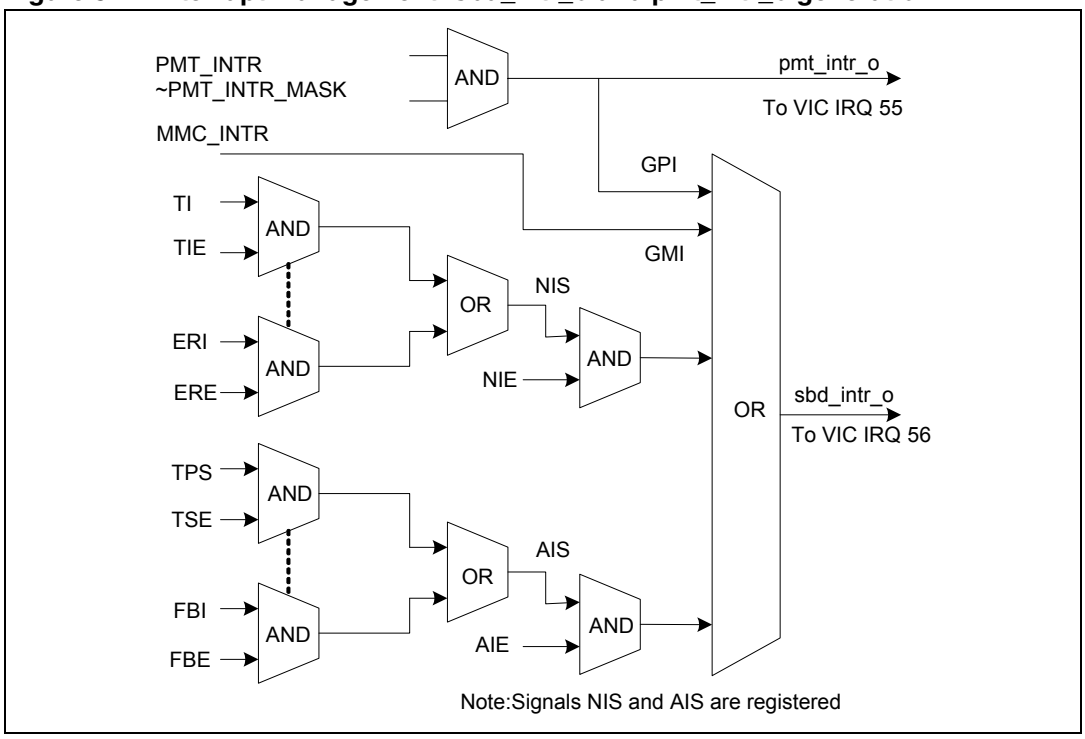
Bit	Name	Description
[31:00]	Buffer 2 Address Pointer (Next Descriptor Address)	This field indicates the physical address of the 2nd data buffer pointed to by this descriptor, if descriptor chaining is used. If RDES1[24] bit is set, then this field contains the pointer to the physical memory when the next descriptor is present.

24.5 How to initialize DMA

1. Write to DMA register0 (Bus Mode) to set the host bus access parameters.
2. Write to DMA register7 (Interrupt Enable) to mask needless interrupt causes.
3. Create the Transmit And Receive descriptor lists into the host physical memory space.
4. Write to both DMA register3 (Receive Descriptor List Address *Section 24.7.6*) and register4 (Transmit Descriptor List Address, *Section 24.7.7*) providing the DMA with the starting address of each descriptor list.
5. Write to MAC register1 (MAC Frame Filter, *Section 24.7.17*), register2 (Hash Table High, *Section 24.7.18*) and register3 (Hash Table Low, *Section 24.7.19*) for described filtering options.
6. Write to MAC register0 (MAC configuration, *Section 24.7.16*) 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).
7. Write to DMA register6 (operation mode, *Section 24.7.9*) setting bits [1] SR and [13] ST to start reception and transmission, respectively.

24.6 Interrupt management

Figure 54. Interrupt management: sbd_intr_o and pmt_intr_o generation



The Ethernet MAC provides two interrupt lines to VIC (Vectored Interrupt Controller):

- sbd_intr_o: general interrupt signal connected to the VIC IRQ 56 line;
- pmt_intr_o: interrupt signal generated from PMT (Power Management) module connected to VIC IRQ 55 line;

The signal pmt_intr_o reflects the combination of the value PMT_INTR in the MAC interrupt status register (Interrupt Status Register (Register14, MAC), bit 3) and the value PMT_INTR_MASK in the MAC interrupt mask register (Interrupt Mask Register (Register 15,MAC), bit 3).

Interrupts can be generated from the MAC Core as a result of various events in the optional modules in it (for example MMC and PMT modules). These interrupt events are combined with events in the DMA on the sbd_intr_o signal. Infact the signal pmt_intr_o is also used to drive the signal sbd_intr_o (GPI in the Figure 4.)

In the same way the MMC block through MMC_INTR (controlled by Interrupt Status Register (Register14, MAC), bit 4) drives the sbd_intr_o signal (GMI in the Figure 4).

The other events in the DMA that drive the sbd_intr_o signal are produced by the combinations of status register bits and interrupt mask bits (listed correspondingly in the Status Register (Register5, DMA) and Interrupt Enable Register (Register7, DMA)).

24.7 Programming model

24.7.1 Register map

The MAC-UNIV can be fully configured by programming a set of 32 bit wide registers which can be accessed at the base address 0xE080_0000.

The MAC-UNIV registers can be grouped in two different classes:

- **DMA registers** (listed in *Table 422*.)
- **MAC registers** (listed in *Table 423*.)

Table 422. MAC-UNIV DMA registers summary

Name	Offset	Reset value	Description
Register 0	0x1000	32'h0	Bus Mode Register.
Register 1	0x1004	32'h0	Transmit Poll Demand Register.
Register 2	0x1008	32'h0	Receive Poll Demand Register.
Register 3	0x100C	32'h0	Receive Descriptor List Address Register.
Register 4	0x1010	32'h0	Transmit Descriptor List Address Register.
Register 5	0x1014	32'h0	Status Register.
Register 6	0x1018	32'h0	Operation Mode Register.
Register 7	0x101C	32'h0	Interrupt Enable Register.
Register 8	0x1020	32'h0	Missed Frame And Buffer Overflow Counter Register.
-	0x1024 to 0x1044	-	Reserved.
Register 18	0x1048	32'h0	Current Host Transmit Descriptor Register.
Register 19	0x104C	32'h0	Current Host Receive Descriptor Register.
Register 20	0x1050	32'h0	Current Host Transmit Buffer Address Register.
Register 21	0x1054	32'h0	Current Host Receive Buffer Address Register.

Table 423. MAC-UNIV MAC global registers summary

Name	Offset	Reset value	Description
Register 0	0x0000	32'h0	Mac Configuration Register.
Register 1	0x0004	32'h0	Mac Frame Filter Register.
Register 2	0x0008	32'h0	Hash Table High Register.
Register 3	0x000C	32'h0	Hash Table Low Register.
Register 4	0x0010	32'h0	Mii Address Register.
Register 5	0x0014	32'h0	Mii Data Register.
Register 6	0x0018	32'h0	Flow Control Register.
Register 7	0x001C	32'h0	Vlan Tag Register.

Table 423. MAC-UNIV MAC global registers summary (continued)

Name	Offset	Reset value	Description
Register 8	0x0020	8'h10	Version Register (RO).
-	0x0024	-	Reserved.
Register 10	0x0028	-	Pointer To Wake-up Frame Filter Registers.
Register 11	0x002C	32'h0	Pmt Control And Status Register.
-	0x0030 to 0x0034	-	Reserved.
Register 14	0x0038	32'h0	Interrupt Register
Register 15	0x003C	32'h0	Interrupt Mask Register
Register16	0x0040	32'h8000FFFF	Mac Address0 High Register.
Register17	0x0044	32'hFFFFFFFF	Mac Address0 Low Register.
Register18	0x0048	32'h0000FFFF	Mac Address1 High Register.
Register19	0x004C	32'hFFFFFFFF	Mac Address1 Low Register.
Register 20 to Register 47	0x0050 to 0x00BC	As for Register18/19	Mac Address1 High/low Registers (With I = 2...15).
-	0x00C0 to 0x00D8	-	Reserved.
-	0x00DC to 0x00FC	-	Reserved.
Register 64 to Register 127	0x0100 to 0x01FC	-	MMC Registers. (Described in *Table 423: MAC-UNIV MAC global registers summary* below)

Table 424. MMC (MAC management counters) registers

Name	Offset	Reset Value	Description
Register 64	0x0100	32'h0	Mmc_cntrl establishes the operating mode of MMC
Register 65	0x0104	32'h0	Mmc_intr_rx maintains the interrupts generated from all of the receive statistics counters.
Register 66	0x0108	32'h0	Mmc_intr_tx maintains the interrupts generated from all of the transmit statistics counters.
Register 67	0x010C	32'h0	Mmc_intr_mask_rx maintains the mask for the interrupt generated from all of the received statistics counters.
Register 68	0x0110	32'h0	Mmc_intr_mask_tx maintains the mask for the interrupt generated from all of the transmit statistics counters.
Register 69	0x0114	32'h0	Txoctetcount_gb is the number of bytes, exclusive of preamble and retried bytes, in good and bad frames
Register 70	0x0118	32'h0	Txframecount_gb is the number of good and bad frames transmitted, exclusive of retried frames.

Table 424. MMC (MAC management counters) registers (continued)

Name	Offset	Reset Value	Description
Register 71	0x011C	32'h0	Txbroadcastframes_g is the number of good broadcast frames transmitted
Register 72	0x0120	32'h0	Txmulticastframes_g is the number of good multicast frames transmitted
Register 73	0x0124	32'h0	Tx64octects_gb is the number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames.
Register 74	0x0128	32'h0	Tx65to127octects_gb is the number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.
Register 75	0x012C	32'h0	Tx128to255octects_gb is the number of good and bad frames transmitted with length between 127 and 255 (inclusive) bytes, exclusive of preamble and retried frames.
Register 76	0x0130	32'h0	Tx256to511octects_gb is the number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.
Register 77	0x0134	32'h0	Tx512to1023octects_gb is the number of good and bad frames transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried frames.
Register 78	0x0138	32'h0	Tx1024tomaxoctects_gb is the number of good and bad frames transmitted with length between1024and mqaxsize (inclusive) bytes, exclusive of preamble and retried frames.
Register 79	0x013C	32'h0	Txunicastframes_gb is the number of good and bad unicast frames transmitted.
Register 80	0x0140	32'h0	Txmulticastframes_gb is the number of good and bad multicast frames transmitted.
Register 81	0x0144	32'h0	Txbroadcastframes_gb is the number of good and bad broadcast frames transmitted.
Register 82	0x0148	32'h0	Txunderflowerror is the number of frames aborted due to frame underflow error.
Register 83	0x014C	32'h0	Txsinglecol_g is the number of successfully transmitted frames after a single collision in Half-duplex mode.
Register 84	0x0150	32'h0	Txsinglecol_g is the number of successfully transmitted frames after a single collision in Half-duplex mode.
Register 85	0x0154	32'h0	Txdeferred is the number of successfully transmitted frames after a deferral in Half-duplex mode.
Register 86	0x0158	32'h0	Txlatecol is the number of frames aborted due to late collision error.
Register 87	0x015C	32'h0	Txexesscol is the number of frames aborted due to excessive (16) collision errors.

Table 424. MMC (MAC management counters) registers (continued)

Name	Offset	Reset Value	Description
Register 88	0x0160	32'h0	Txcarriererror is the number of frames aborted due to carrier sense error (no carrier or loss of carrier).
Register 89	0x0164	32'h0	Txoctetcount_g is the number of bytes transmitted, exclusive of preamble, in good frames only.
Register 90	0x0168	32'h0	Txframecount_g is the number of good frames transmitted.
Register 91	0x016C	32'h0	Txexcessdef is the number of frames aborted due to excessive deferral error (deferred for more than two max-sized frame times).
Register 92	0x0170	32'h0	Txpauseframes is the number of good PAUSE frames transmitted.
Register 93	0x0174	32'h0	Txvlanframes_g is the number of good VLAN frames transmitted, exclusive of retried frames.
Register 94	0x0178	32'h0	Reserved
Register 95	0x017C	32'h0	Reserved
Register 96	0x0180	32'h0	Rxframecount_gb is the number of good and bad frames received.
Register 97	0x0184	32'h0	Rxoctetcount_gb is the number of bytes received exclusive of preamble, in good and bad frames.
Register 98	0x0188	32'h0	Rxoctetcount_g is the number of bytes received exclusive of preamble, only in good frames.
Register 99	0x018C	32'h0	Rxbroadcastframes_g is the number of good broadcast frames received.
Register 100	0x0190	32'h0	Rxmulticastframes_g is the number of good multicast frames received.
Register 101	0x0194	32'h0	Rxcrcerror is the number of frames received with CRC error.
Register 102	0x0198	32'h0	Rxalignmenterror is the number of frames received with alignment (dribble) error. Valid only in 10/100 mode.
Register 103	0x019C	32'h0	Rxrunterror is the number of frames received with runt (<64 bytes and CRC error) error.
Register 104	0x01A0	32'h0	Rxjabbererror is the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes with VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, the frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.
Register 105	0x01A4	32'h0	Rxundersize_g is the number of frames received with length less than 64 bytes, without any errors.
Register 106	0x01A8	32'h0	Rxoversize_g is the number of frames received with length greater than the maxsize (1,518 0r 1,522 for VLAN tagged frames) without errors.

Table 424. MMC (MAC management counters) registers (continued)

Name	Offset	Reset Value	Description
Register 107	0x01AC	32'h0	Rx64octects_gb is the number of good and bad frames received with length 64 bytes, exclusive of preamble.
Register 108	0x01B0	32'h0	Rx65to127octects_gb is the number of good and bad frames received with length between 127 and 255 (inclusive) bytes, exclusive of preamble.
Register 109	0x01B4	32'h0	Rx128to255octects_gb is the number of good and bad frames transmitted with length between 127 and 255 (inclusive) bytes, exclusive of preamble.
Register 110	0x01B8	32'h0	Rx256to511octects_gb is the number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble.
Register 111	0x01BC	32'h0	Rx512to1023octects_gb is the number of good and bad frames transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble.
Register 112	0x01C0	32'h0	Rx1023tomaxoctects_gb is the number of good and bad frames transmitted with length between 1023 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.
Register 113	0x01C4	32'h0	Rxunicastframes_g is the number of good unicast frames received.
Register 114	0x01C8	32'h0	Rxlengtherror is the number of frames received with length error (length type field!= frame size) for all frames with valid length field.
Register 115	0x01CC	32'h0	Rxoutofrangetype is the number of frames received with length field not equal to the valid frame size (greater than 1500 but less than 1536).
Register 116	0x01D0	32'h0	Rxpauseframes is the number of good and valid PAUSE frames received.
Register 117	0x01D4	32'h0	Rxfifooverflow is the number of missed received frames due to FIFO overflow.
Register 118	0x01D8	32'h0	Rxvlanframes_gb is the number of good and bad VLAN frames received.
Register 119	0x01DC	32'h0	Rxwatchdogerror is the number of frames received with error due to watchdog timeout error (frames with a data load larger than 2,048 bytes).
Register 120-127	0x01E0-0x01FC	32'h0	Reserved

24.7.2 Register description

24.7.3 Bus mode register (Register0, DMA)

The bus mode is a register which establishes the bus operating mode for the DMA. The bus mode bit assignments are given in *Table 425*

Table 425. Bus mode register bit assignments

Bit	Name	Reset Value	Type	Description
[31:17]	Reserved	-	RO	Read: undefined
[16]	*FB*	1'h0	RW	Fixed Burst
[15:14]	*PR*	2'h0	RW	Rx:Tx Priority Ratio.
[13:08]	*PBL*	6'h0	RW	Programmable Burst Length.
[07]	Reserved	-	RO	Read: undefined
[06:02]	*DSL*	5'h0	RW	Descriptor Skip Length.
[01]	*DA*	1'h0	RW	DMA Arbitration Scheme
[00]	*SWR*	1'h0	RW	Software Reset

- FB

  Setting this bit, the AHB Master interface performs only fixed bursts transfers (SINGLE, INCR4, INCR8 or INCR16). In contrast, the AHB will use SINGLE and INCR burst only.
- PR

  This 2 bit field indicates the ratio of the RxDMA requests given priority over TxDMA request, according to encoding below:

VALUE	Rx:Tx RATIO
2'b00	1:1
2'b01	2:1
2'b10	3:1
2'b11	4:1

- PBL

  This 6 bit field states the maximum number of beats to be transferred in one DMA transmission. Each time DMA starts a burst transfer on the host bus, it will always attempt to burst as specified by PBL value. Valid values for PBL are 1, 2, 4, 8, 16 and 32, and any other value will result in undefined behavior.
- DSL

  This 5 bit field specifies the number of Word/Dword/Long (depending on 32(64/128 bit bus) to skip between two unchained descriptors. If DSL is zero (5'h0, default) then the descriptor table is taken as contiguous by the DMA in ring mode.
- DA

  This bit allows the selection of the DMA arbitration scheme, according to encoding below:

VALUE	ARBITRATION SCHEME
1‘b0	Round robin with Rx:Tx priority given in PR field.
1‘b1	Rx has priority over Tx.

- SWR

  Setting this bit, the DMA Controller resets all MAC internal registers and logic. This bit is automatically cleared after the reset has completed.

24.7.4 Transmit poll demand register (Register1, DMA)

The Transmit Poll Demand is a register which enables the transmit DMA to check whether or not the current descriptor is owned by DMA. The transmit poll demand bit assignments are given in *Table 426*.

Table 426. Transmit poll demand register bit assignments

Bit	Name	Reset Value	Type	Description
[31:00]	TPD	32’h0	RW	Transmit Poll Demand.

- **TPD**

  When these bits are written with any value, the DMA reads the current descriptor pointed to by DMA Register18 (**Current Host Transmit Descriptor)**. If the pointed descriptor is available the transmission resumes, otherwise (that is, the descriptor is owned by the host), transmission returns to suspend state and TU bit in DMA Register5 (**Status)** is asserted.

24.7.5 Receive poll demand register (Register2, DMA)

The Receive Poll Demand is a register which enables the receive DMA to check for new descriptors. The Receive Poll Demand bit assignments are given in *Table 427*

Table 427. Receive poll demand register bit assignments

Bit	Name	Reset Value	Type	Description
[31:00]	*RPD*	32’h0	RW	Receive Poll Demand.

- RPD

  When these bits are written with any value, the DMA reads the current descriptor pointed to by DMA Register19 **(Current Host Receive Descriptor)**. If the pointed descriptor is available the reception resumes, otherwise (that is, the descriptor is owned by the host), reception returns to suspend state and RU bit in DMA Register5 **(Status)** is asserted.

24.7.6 Receive descriptor list address register (Register3, DMA)

The receive descriptor list address is a register which points to the start of the Receiver Descriptor List. The Receive Descriptor List address bit assignments are given in *Table 428.*

Note: Writing to this register is permitted only when reception is stopped. When stopped, the register must be written to before the receive Start command is given.

Table 428. Receive descriptor list address register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	SRL	32'h0	RW	Start of receive list.

24.7.7 Transmit descriptor list address register (Register4, DMA)

The Transmit Descriptor List Address is a register which points to the start of the Transmit Descriptor List. The Transmit Descriptor List Address bit assignments are given in *Table 429*.

Note: Writing to this register is permitted only when transmission is stopped. When stopped, the register can be written to before the transmission Start command is given.

Table 429. Transmit descriptor list address register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	STL	32'h0	RW	Start of transmit list.

24.7.8 Status register (Register 5, DMA)

The Status is a RO register which contains all the status bit that the DMA reports to the host, and it is usually read by the software driver during an interrupt service routine or polling. The Status bit assignments are given in *Table 430*.

Note: The Status register bits are not cleared when read. Unreserved bits [16:0] in this register are cleared writing 1'b1 to them, whereas writing 1'b0 has no effect. The same [16:0] bits can be masked by the appropriate bits in Register 7 (Interrupt Enable Register).

Table 430. Status register bit assignments

Bit	Name	Reset Value	Type	Description
[31:29]	Reserved	-	RO	Read:undefined
[28]	*GPI*	1'h0	RO	MAC PMT Interrupt
[27]	*GMI*	1'h0	RO	MAC MMC Interrupt
[26]	Reserved	-	RO	Read:undefined
[25:23]	*EB*	3'h0	RO	Error bits
[22:20]	*TS*	3'h0	RO	Transmit Process State.
[19:17]	*RS*	3'h0	RO	Receive Process State.
[16]	*NIS*	1'h0	RW	Normal Interrupt Summary.
[15]	*AIS*	1'h0	RW	Abnormal Interrupt Summary
[14]	*ERI*	1'h0	RW	Early Receive Interrupt.
[13]	*FBI*	1'h0	RW	Fatal Bus Error Interrupt.
[12:11]	Reserved	-	RO	Read: undefined
[10]	*ETI*	1'h0	RW	Early Transmit Interrupt.
[09]	*RWT*	1'h0	RW	Receive Watchdog Timeout
[08]	*RPS*	1'h0	RW	Receive Process Stopped
[07]	*RU*	1'h0	RW	Receive Buffer Unavailable
[06]	*RI*	1'h0	RW	Receive Interrupt.
[05]	*UNF*	1'h0	RW	Transmit Underflow
[04]	*OVF*	1'h0	RW	Receive Overflow
[03]	*TJT*	1'h0	RW	Transmit Jabber Timeout
[02]	*TU*	1'h0	RW	Transmit Buffer Unavailable
[01]	*TPS*	1'h0	RW	Transmit Process Stopped
[00]	*TI*	1'h0	RW	Transmit Interrupt.

- GPI

  This bit reflects the **pmt_intr_o** signal output of the MAC Core, in the frame of PMT (Power Management) module.

Note: The corresponding registers in MAC Core must be read to get the exact cause of this interrupt and clear the source.

- GMI

  This bit reflects an interrupt event in the MMC (MAC Management Counters) module of the MAC Core.

Note: The corresponding registers in MAC Core must be read to get the exact cause of this interrupt and clear the source.

- EB

  This 3 bit field indicates the type of error that caused a Bus Error response on the AHB interface, according to encoding below:

Table 431. EB field bit assignments

Bit 23	Bit 24	Bit 25	Error
1‘b0	-	-	During data transfer by RxDMA.
1‘b1	-	-	During data transfer by TxDMA.
-	1‘b0	-	During write transfer.
-	1‘b1	-	During read transfer.
-	-	1‘b0	During data buffer access.
-	-	1‘b1	During descriptor access.

This field does not generate an interrupt. This field is valid only when FBI bit in this register is set.

- **TS**

  This 3 bit field reflects the state of the Transmit DMA FSM, according to encoding below:

Table 432. TS filed bit assignments

Value	State	Description
3‘b000	Stopped	Reset or stop transmission 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 (TxFIFO).
3‘b100	Reserved	-
3‘b101	Reserved	-
3‘b110	Suspended	Transmit descriptor unavailable or transmit buffer underflow.
3‘b111	Running	Closing transmitting descriptor.

- RS

  This 3 bit field reflects the state of the Receive DMA FSM, according to encoding below:

Table 433. RS field bit assignments

Value	State	Description
3‘b000	Stopped	Reset or stop reception command issued.
3‘b001	Running	Fetching receive transfer descriptor.
3‘b010	Reserved	-
3‘b011	Running	Waiting for receive packet.
3‘b100	Suspended	Receive descriptor unavailable.
3‘b101	Running	Closing receiving descriptor.
3‘b110	Reserved	-
3‘b111	Running	Transferring the receive packet data from receiver buffer to host memory.

- NIS

  The value of this bit is the logical OR of the following bits in this register (if corresponding interrupt bits are enabled in DMA Register7, section 1.4.2.8, that is only unmasked bits affect NIS):

Table 434. NIS field bit assignments

FIELD		Bit
Transmit Interrupt	TI	0
Transmit Buffer Unavailable	TU	2
Receive Interrupt	RI	6
Early Receive Interrupt	ERI	14

Note: This bit must be cleared (writing a 1'b1) each time a corresponding bit that causes NIS to be set is cleared.

- AIS

  The value of this bit is the logical OR of the following bits in this register (if corresponding interrupt bits are enabled in DMA Register7, section 1.4.2.8, that is only unmasked bits affect AIS):

Table 435. AIS field bit assignments

Field		Bit
Transmit Process Stopped	TPS	1
Transmit Jabber Timeout	TJT	3
Receive FIFO Overflow	OVF	4
Transmit Underflow	UNF	5
Receive Buffer Unavailable	RU	7
Receive Process Stopped	RPS	8
Receive Watchdog Timeout	RWT	9

Table 435. AIS field bit assignments (continued)

Field		Bit
Early Transmit Interrupt	ETI	10
Fatal Bus Error	FBI	13

Note: This bit must be cleared (writing a 1'b1) each time a corresponding bit that causes AIS to be set is cleared.

- ERI

  If set it indicates that the DMA had filled the first data buffer of the packet. The RI bit in this register automatically clears the ERI bit.
- FBI

  If set it indicates that a Bus Error occurred (refer to EB field in this register), and DMA disables all its bus accesses.
- ETI

  If set it indicates that the frame to be transmitted was fully transferred to MAC.
- RWT

  This bit is set when a frame with a length greater than 2048 bytes is received.
- RPS

  This bit is set when the Receive Process enters in the Stopped state (refer to RS field in this register).
- RU

  If set it indicates that the Next Descriptor in the Receive list is owned by the host and it can't be acquired by DMA (receive buffer unavailable), resulting in Receive Process

suspended. This bit is set only when the previous descriptor in Receive list is owned by DMA.

- RI

  If set it indicates the completion of frame reception. Note that Receive Process remains in running state.

- UNF

  If set it indicates that the Transmit Buffer had an underflow during frame transmission. Transmission is then suspended and an underflow error is set in TDES0.

- OVF

  If set it 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.

- TJT

  If set it indicates that the transmit jabber timeout expired, meaning that the transmitter had been excessively active. Transmission is then aborted and placed in Stopped state, causing the bit [14] in TDES0 to be set.

- TU

  If set it indicates that the Next Descriptor in the Transmit list is owned by the host and it can't be acquired by DMA (transmit buffer unavailable), resulting in Transmit Process suspended.

- TPS

  This bit is set when the Transmit Process enters in the Stopped state (refer to TS field in this register).

- TI

  If set it indicates the completion of frame transmission, and bit [31] in TDES1 is set for the first descriptor.

24.7.9 Operation mode register (Register 6, DMA)

The Operation Mode is a register which establishes the transmit and receive operating modes and commands. The Operation Mode bit assignments are given in *Table 436*.

Note: The operation mode register should be the last CSR to be written as part of DMA initialization.

Table 436. Operation mode register bit assignments

Bit	Name	Reset Value	Type	Description
[31:22]	Reserved	-	RO	Read: undefined
[21]	*SF*	1'h0	RW	Store and Forward.
[20]	*FTF*	1'h0	RW	Flush Transmit FIFO.
[19:17]	Reserved	-	RW	Read: undefined.
[16:14]	*TTC*	3'h0	RW	Transmit Threshold Control.
[13]	*ST*	1'h0	RW	Start/Stop Transmission Command.
[12:11]	*RFD*	2'h0	RW	Threshold for De-activating Flow Control.
[10:09]	*RFA*	2'h0	RW	Threshold for Activating Flow Control.
[08]	*EFC*	1'h0	RW	Enable HW Flow Control.
[07]	*FEF*	1'h0	RW	Forward Error Frames.
[06]	*FUF*	1'h0	RW	Forward Undersized Good Frames.
[05]	Reserved	-	RO	Read: undefined.
[04:03]	*RTC*	2'h0	RW	Receive Threshold Control.
[02]	*OSF*	1'h0	RW	Operate on Second Frame.
[01]	*SR*	1'h0	RW	Start/Stop Receive.
[00]	Reserved	-	RO	Read: undefined.

- SF

  Setting this bit, the transmission starts when a full frame resides in the Transmit FIFO. If set, the TTC field in this register is ignored.

Note: This bit should be changed only when transmission is stopped.

- FTF

  Setting this bit, the Transmit FIFO controller logic is reset and all data in the FIFO is flushed (lost). When the flushing is fully completed, this bit is automatically cleared.

- TTC

  This 3 bit field allows start of transmission when the frame size in the Transmit FIFO is larger than the stated threshold, according to encoding below:

Table 437. TTC field bit assignments

Value	Threshold (Byte)
3'b000	64
3'b001	128
3'b010	192
3'b011	256
3'b100	40
3'b101	32

Table 437. TTC field bit assignments (continued)

Value	Threshold (Byte)
3‘b110	24
3‘b111	16

Note: This field is used only when SF bit in this register is cleared.

- ST

  Setting this bit, the transmission process is placed in the Running state, and the DMA checks the Transmit List for a frame to be transmitted either at the current position (pointed by the Transmit Descriptor List Address register, *Section 24.7.7*) or at position retained in case of transmission was stopped previously.

  Clearing this bit, the transmission process is placed in the Stopped state after completing the transmission of the current frame.

- RFD

  This 2 bit field controls the threshold (that is, fill-level of Receive FIFO) at which the flow-control (in both HD and FD) is de-asserted after activation, according to encoding below:

Table 438. RFD field bit assignments

Value	Threshold
2‘b00	(Full - 1K) bytes
2‘b01	(Full - 2K) bytes
2‘b10	(Full - 3K) bytes
2‘011	(Full - 4K) bytes

- RFA

  This 2 bit field controls the threshold (that is, fill-level of Receive FIFO) at which the flow-control (in both HD and FD) is activated, according to encoding below:

Table 439. RFA field bit assignments

Value	Threshold
2‘b00	(Full - 1K) bytes
2‘b01	(Full - 2K) bytes
2‘b10	(Full - 3K) bytes
2‘011	(Full - 4K) bytes

Note: This threshold is applicable only for Receive FIFO of size of 4Kbytes and above, and when bit EFC in this register is set.

- EFC

  Setting this bit, the flow-control operation based on fill-level (threshold) of Receive FIFO is enabled.

Note: This bit is not used (reserved) when the Receive FIFO size is less than 4Kbytes.

- FEF

  Setting this bit, all frames except runt-error frames will be forwarded to the DMA. Otherwise, the Receive FIFO will drop frames with error status.

- FUF

  Setting this bit, the Receive FIFO will forward undersized frames (frames with no error and length less than 64 bytes, including pad-bytes and CRC). Otherwise, the Receive FIFO will drop all frames of size less than 64 bytes, unless frame is already transferred due to lower value of RTC value (in this register).

- RTC

  This 2 bit field allows start of transfer request to DMA when the frame size in the Receive FIFO is larger than the stated threshold, according to encoding below:

Table 440. RTC field bit assignments

Value	Threshold (Byte)
2‘b00	64
2‘b01	32
2‘b10	96
2‘011	128

- OSF

  Setting this bit, the DMA is instructed to process a second frame of the Transmit List even before to obtain the status of the first frame.

- SR

  Setting this bit, the receive process is placed in the Running state, and the DMA attempts to acquire the descriptor from the Receive List and process incoming frames. Descriptor acquisition is attempted from the current position (pointed by the Receive Descriptor List Address register) or at position retained in case of reception was previously stopped.

  Clearing this bit, the receive process is placed in the Stopped state after completing the transmission of the current frame.

24.7.10 Interrupt enable register (Register7, DMA)

The Interrupt Enable is a register which enables the interrupts reported by register5 (*Section 24.7.8*). The Interrupt Enable Bit assignments are given in *Table 441*.

Note: Setting a bit enables the corresponding interrupt. After reset, all interrupts are disabled.

Table 441. Interrupt enable register bit assignments

Bit	Name	Reset value	Type	Description
[31:17]	Reserved	-	RO	Read: undefined.
[16]	NIE	1’h0	RW	Normal interrupt summary enable.
[15]	AIE	1’h0	RW	Abnormal interrupt summary enable.
[14]	ERE	1’h0	RW	Early receive interrupt enable.

Table 441. Interrupt enable register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[13]	FBE	1'h0	RW	Fatal bus error interrupt enable.
[12:11]	Reserved	-	RO	Read: undefined.
[10]	ETE	1'h0	RW	Early transmit interrupt enable.
[09]	RWE	1'h0	RW	Receive watchdog timeout enable.
[08]	RSE	1'h0	RW	Receive stopped enable.
[07]	RUE	1'h0	RW	Receive buffer unavailable enable.
[06]	RIE	1'h0	RW	Receive interrupt enable.
[05]	UNE	1'h0	RW	Underflow interrupt enable.
[04]	OVE	1'h0	RW	Overflow interrupt enable.
[03]	TJE	1'h0	RW	Transmit jabber timeout enable.
[02]	TUE	1'h0	RW	Transmit buffer unavailable enable.
[01]	TSE	1'h0	RW	Transmit stopped enable.
[00]	TIE	1'h0	RW	Transmit interrupt enable.

24.7.11 Missed frame and buffer overflow counter register (Register8, DMA)

The Missed Frame And Buffer Overflow Counter is a register which reports the current value of the two counters maintained by DMA Controller to track the number of missed frames during reception.

As stated in the bit assignments given in *Table 442*, bits [15:0] indicate the number of missed frames due to the host buffer being unavailable, and bits [27:17] indicate the number of missed frames due to buffer overflow conditions.

Table 442. Missed frame and buffer overflow counter register bit assignments

Bit	Name	Reset Value	Type	Description
[31:29]	Reserved	-	RO	Read: undefined
[28]	-	1'h0	RW	Overflow for FIFO Overflow Counter
[27:17]	-	11'h0	RW	Number of frames missed by the application
[16]	-	1'h0	RW	Overflow for Missed Frame Counter.
[15:00]	-	16'h0	RW	Number of frames missed by the controller.

24.7.12 Current host transmit descriptor register (Register18, DMA)

The Current Host Transmit Descriptor is a RO register which points to the start address of the current transmit descriptor read by the DMA. This pointer is updated by DMA during operation.

24.7.13 Current host receive descriptor register (Register19, DMA)

The Current Host Receive Descriptor is a RO register which points to the start address of the current receive descriptor read by the DMA. This pointer is updated by DMA during operation.

24.7.14 Current host transmit buffer address register (Register20, DMA)

The Current Host Transmit Buffer Address is a RO register which points to the current transmit buffer address being read by the DMA. This pointer is updated by DMA during operation.

24.7.15 Current host receive buffer address register (Register21, DMA)

The Current Host Receive Buffer Address is a RO register which points to the current receive buffer address being read by the DMA. This pointer is updated by DMA during operation.

24.7.16 MAC configuration register (Register0, MAC)

The MAC configuration is a register which establishes receive and transmit operating modes. The MAC configuration bit assignments are given in *Table 443*.

Table 443. MAC configuration register bit assignments

Bit	Name	Reset Value	Type	Description
[31:24]	Reserved	-	RO	Read: undefined
[23]	*WD*	1'h0	RW	Watchdog Disable.
[22]	*JD*	1'h0	RW	Jabber Disable.
[21]	Reserved	-	RO	Read: undefined.
[20]	*JE*	1'h0	RW	Jumbo Frame Enable.
[19:17]	*IFG*	3'h0	RW	Inter Frame Gap.
[16]	*DCRS*	1'b0	RW	Disable Carrier Sense During Transmission
[15]	Reserved	-	RO	Read: undefined.
[14]	Reserved	-	RO	Read: undefined.
[13]	*DO*	1'h0	RW	Disable Receive Own.
[12]	*LM*	1'h0	RW	Loop-back Mode.
[11]	*DM*	1'h0	RW	Duplex Mode.
[10]	*IPC*	1'h0	RW	Checksum Offload.
[09]	*DR*	1'h0	RW	Disable Retry.
[08]	Reserved			Reserved
[07]	*ACS*	1'h0	RW	Automatic Pad/CRC Stripping.
[06:05]	*BL*	2'h0	RW	Back-off Limit.
[04]	*DC*	1'h0	RW	Deferral Check.
[03]	*TE*	1'h0	RW	Transmitter Enable.
[02]	*RE*	1'h0	RW	Receiver Enable.
[01:00]	Reserved	-	RO	Read: undefined.

- WD

  Setting this bit, the MAC disables the watchdog timer on the receiver. Otherwise, MAC allows no more than 2048 bytes (10240 bytes, if JE bit in this register is set) of the receiving frame and cuts off any bytes received after that.

- JD

  Setting this bit, the MAC disables the jabber timer on the transmitter. Otherwise, MAC cuts off the transmitter if the application sends out more than 2048 bytes (10240 bytes, if JE bit in this register is set) of data during transmission.

- JE

  Setting this bit, the MAC allows Jumbo frames of size 9018 bytes (9022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.

Note: JD bit in this register should be set in order to transmit jumbo frames.

- IFG

Note: This 3 bit field controls the minimum inter frame gap between frames during transmission, according to encoding below:

Table 444. IFG field bit assignments

Value	Inter Frame Gap
3‘b000	96 bit times
3‘b001	88 bit times
3‘b010	80 bit times
...	...
3‘b111	40 bit times

Note: In half-duplex mode, the minimum IFG can be configured up to 64 bit times (IFG = 3'b100).

- DCRS

  When set high, this bit makes the MAC transmitter ignore the 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.

- DO

  Setting this bit, the MAC disables the reception of frame when the *mii_txen_o* is asserted in half-duplex mode. Otherwise, the MAC receives all packets that are given by the PHY while transmitting.

Note: This bit is not applicable (RO with default value) if the MAC is operating in full-duplex.

- LM

  Setting this bit, the MAC operates in loop-back mode at MII. In this mode, the MII receive clock input is required for the loop-back to work properly.

- DM

  Setting this bit, the MAC operates in a full-duplex mode where it can transmit and receive simultaneously.

Note: This bit is RO with default value of 1'b1 in full-duplex only configuration.

- IPC

  Setting this bit, the MAC calculates the 16 bit 1's complement of the 1's complement sum of the payload data (16 bit) and sends it to the application at the end of frame.

- DR

  Setting this bit, the MAC will attempt only one transmission. In case of a collision, the MAC will ignore the current frame transmission and report a Frame Abort with excessive collision error in the transmit frame status.

  Clearing this bit, the MAC will attempt retries based on the settings of BL field in this register (bits [06:05]).

Note: This bit is applicable only to half-duplex mode and it is reserved in full-duplex only configuration.

- ACS

  Setting this bit, the MAC will strip the Pad/FCS field on incoming frames only if the length's field value is less than or equal to 1500 bytes. All received frames with length

field greater than or equal to 1501 bytes will be passed to the application without stripping the Pad/FCS field.

Clearing this bit, the MAC will pass unmodified all incoming frames to the application.

- BL

  This 2 bit field represents the back-off limit which determines the random integer number (r) of slot time delays (that is., 512 bit times) the MAC waits before rescheduling a transmission attempt during retries after a collision.

  The random integer r takes value ranging from 0 to 2k (2k not included), being k specified by BL field according to encoding below:

Table 445. BL field bit assignments

Value	K
2‘b00	min (n,10)
2‘b01	min (n,8)
2‘b10	min (n,4)
2‘b11	min(n,1)

Note: 1 where n is the number of retransmission attempts.

2 This bit is applicable only to half-duplex mode and it is reserved (RO) in full-duplex only configuration.

- DC

  Setting this bit, the deferral check function is enabled in the MAC. The MAC will issue a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status when transmit state machine is deferred for more than 24 288 bit times.

Note: This bit is applicable only to half-duplex mode and it is reserved (RO) in full-duplex only configuration.

- TE

  Setting this bit, transmit state machine of the MAC is enabled for transmission on the MII. Otherwise, transmit state machine is disabled after the completion of the transmission of the current frame, and will not transmit any further frames.

- RE

  Setting this bit, receive state machine of the MAC is enabled for receiving frames from the MII. Otherwise, receive state machine is disabled after the completion of the reception of the current frame, and will not receive any further frames.

24.7.17 MAC frame filter register (Register1, MAC)

The MAC frame filter is a register which contains the filter controls for receiving frames. The MAC frame filter bit assignments are given in *Table 446*.

Note: The 1st level of filtering is performed going to the address check block of the MAC (address filtering). The 2nd level of filtering is performed on the incoming frame, based on other controls such as ‘pass bad frames’ or ‘pass control frames’.

Table 446. MAC frame filter register bit assignments

Bit	Name	Reset Value	Type	Description
[31]	*RA*	1'h0	RW	Receive All.
[30:11]	Reserved	-	RO	Read: undefined
[10]	*HPF*	1'b0	RW	Hash or Perfect Filter
[09]	*SAF*	1'h0	RW	Source Address Filter Enable.
[08]	*SAIF*	1'h0	RW	SA Inverse Filtering.
[07:06]	*PCF*	2'h0	RW	Pass Control Frames.
[05]	*DBF*	1'h0	RW	Disable Broadcast Frames.
[04]	*PM*	1'h0	RW	Pass All Multicast.
[03]	*DAIF*	1'h0	RW	DA Inverse Filtering.
[02]	*HMC*	1'h0	RW	Hash MultiCast.
[01]	*HUC*	1'h0	RW	Hash UniCast.
[00]	*PR*	1'h0	RW	Promiscuous Mode.

- RA

  Setting this bit, the MAC Receiver module passes to the application all frames received regardless of whether they pass the address filter or not (but result of SA/DA filtering is updated - pass or fail - in the corresponding bits in the received frame status word (Receive Descriptor 0, RDES0).

  Clearing this bit, only frames that pass the SA/DA address filter are passed to the application.

- HPF

  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. When low and if the HUC/HMC bit is set, the frame is passed only if it matches the Hash filter.

- SAF

  Setting this bit, the MAC drops frame when SA filter fails (that is, when SA field of received frames doesn't match with the values programmed in the enabled SA registers). Clearing this bit, the MAC Core forwards the received frame to the application and with the updated received frame status word (Receive Descriptor 0, RDES0, section 1.2.3.1) depending on the SA address comparison.

- SAIF

  Setting this bit, the frames whose SA matches the SA registers will be marked as failing the SA address filter (inverse filtering mode). Otherwise (bit cleared), frames whose SA doesn't match the SA registers will be marked as failing the SA address filter (nominal filtering mode).

- PCF

  This 2 bit field controls the forwarding of all control frames (including unicast and multicast PAUSE frames), according to encoding below:

Table 447. PCF field bit assignments

Value	Description
2‘b00	MAC filters all control frames from reaching application
2‘b01	
2‘b10	MAC forwards all control frames to application even if they fail the address filter.
2‘b11	MAC forwards all control frames that pass the address filter.

- DBF

  Setting this bit, the MAC address filtering module filters all incoming broadcast frames.
- PM

  Setting this bit, all received frames with a multicast destination address (first bit in the destination address is 1'b1) are passed. If this bit is cleared, filtering of multicast frames depends on HMC field (bit [2]) in this register.
- DAIF

  Setting this bit, the unicast/multicast frames whose DA matches the DA registers will be marked as failing the DA address filter (inverse filtering mode). Otherwise (bit cleared), frames whose DA doesn't match the DA registers will be marked as failing the DA address filter (nominal filtering mode).
- HMC

  Setting this bit, the MAC performs destination address filtering of received multicast frames according to the hash table (as set in Register2, MAC, and Register3, MAC, in section 1.4.2.16 and 1.4.2.17, respectively). Clearing this bit, the MAC performs a perfect destination address filtering for multicast frames (comparing the DA field with the values programmed in DA registers).
- HUC

  Setting this bit, the MAC performs destination address filtering of received unicast frames according to the hash table (as set in Register2, MAC, and Register3, MAC. Clearing this bit, the MAC performs a perfect destination address filtering for unicast frames (comparing the DA field with the values programmed in DA registers).
- PR

  On setting this bit, the MAC address filtering module passes all incoming frames regardless of its destination or source address. In this case, the SA/DA filter fails status bit of the received frame status word will always be cleared.

24.7.18 Hash table high register (Register2, MAC)

The Hash Table High (HTH) is a register which contains the upper 32 bits of the 64 bit hash table used for group address filtering.

24.7.19 Hash table low register (Register3, MAC)

The Hash Table Low (HTL) is a register which contains the lower 32 bits of the 64 bit hash table used for group address filtering.

24.7.20 MII address register (Register4, MAC)

The MII Address is a register which controls the management cycles to the external PHY through the management interface. The MII address bit assignments are given in *Table 448*.

Table 448. MII address register bit assignments

Bit	Name	Reset Value	Type	Description
[31:16]	Reserved	-	RO	Read: undefined.
[15:11]	*PA*	5'h0	RW	Physical Layer Address.
[10:06]	*GR*	5'h0	RW	MII Register.
[05]	Reserved	-	RO	Read: undefined.
[04:02]	*CR*	3'h0	RW	CSR Clock Range.
[01]	*GW*	1'h0	RW	MII Write.
[00]	*GB*	1'h0	RW	MII Busy.

- PA

  This 5 bit field tells which of the 32 possible PHY devices are being accessed.
- GR

  This 5 bit field selects the desired MII register in the selected PHY device.
- CR

  This 3 bit field allows selection of frequency range of CSR clock (provided as input by the application) and it is used to set the frequency of the MDC (MAC DMA Controller) clock, according to encoding below:

Table 449. CR field bit assignments

Value	CSR Frequency Range	MDC Clock
3‘b000	60-100 MHz	CSR clock/42
3‘b001	100-150 MHz	CSR clock/62
3‘b010	20-35 MHz	CSR clock/16
3‘b011	35-60 MHz	CSR clock/26
3‘b100	150-250 MHz	CSR clock/102
3‘b101	250-300 MHz	CSR clock/122
3‘b110	Reserved	-
3‘b111	Reserved	-

- GW

  If this bit is set, the PHY is informed that the current operation will be a Write operation using the MII Data register. Otherwise (bit cleared), this will be a Read operation placing the data in the MII Data register.

- GB

  This bit should read a logic 1'b0 before writing to this register (Register4, MII Address) and Register5 (MII Data, section 1.4.2.19). 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.

  This bit must be set to 1'b0 during a Write to this Register4. This Register4 should not be written to until this bit is cleared.

  Register5 should be kept valid until this bit GB is cleared by the MAC during a PHY Write operation. Besides, the same Register5 is invalid until this bit is cleared by the MAC during a PHY Read operation.

24.7.21 MII data register (Register5, MAC)

The MII data is a register which stores the 16 bit write data to be written to the PHY register located at the address indicated in MII address register (*Section 24.7.20*). It also stores the 16 bit read data from the PHY register located at the same address. The MII data bit assignments are given in *Table 450*.

Table 450. MII data register bit assignments

Bit	Name	Reset value	Type	Description
[31:16]	Reserved	-	RO	Read: undefined.
[15:00]	GD	16’h0	RW	MII data.

24.7.22 Flow control register (Register6, MAC)

The Flow Control is a register which controls the generation and reception of the Control (pause command) frames by the MAC. The flow control bit assignments are given in *Table 451*.

Table 451. Flow control register bit assignments

Bit	Name	Reset Value	Type	Description
[31:16]	*PT*	16'h0	RW	Pause Time.
[15:08]	Reserved	-	RO	Read: undefined.
[7]	*DZPQ*	1'b0	RW	Disable Zero-Quanta Pause
[6]	Reserved	-	RO	Read: undefined
[05:04]	*PLT*	2'h0	RW	Pause Low Threshold.
[03]	*UP*	1'h0	RW	Unicast Pause Frame Detect.
[02]	*RFE*	1'h0	RW	Receive Flow Control Enable.
[01]	*TFE*	1'h0	RW	Transmit Flow Control Enable.
[00]	*FCB/BPA*	1'h0	RW	Flow Control Busy/Back-Pressure Activate.

- PT

  This 16 bit field represents the value (expressed as an integer number of slot times) to be used in the Pause Time field in the transmit control frame.
- DZPQ

  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 flow control signal). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.
- PLT

  This 2 bit field allows configuration of the threshold of the PAUSE timer at which the input flow control is checked for automatic re-transmission of PAUSE frame, according to encoding below:

Table 452. PLT field bit assignments

Value	Threshold
2'b00	Pause Time - 4 slot time
2'b01	Pause Time - 28 slot time
2'b10	Pause Time - 144 slot time
2'b11	Pause Time - 256 slot time

where, Pause Time is configured by the PT field in this register (see above), and slot time is the time taken to transmit 512 bits (64 bytes) on the MII interface.

Note: The threshold value specified by PLT should be always greater than the Pause Time (PT field).

- UP

  Setting this bit, the MAC will detect the Pause frames with the station's unicast address specified in MAC Address0 High register (section 1.4.2.24) and MAC Address0 Low

register (1.4.2.25), in addition to the detecting Pause frame with the unique multicast address.

Clearing this bit, the MAC will detect only a Pause frame with the unique multicast address specified in the 802.3x standard.

- RFE

  Setting this bit, the MAC will decode the received Pause frame and disable its transmitter for a specified time (Pause Time). Otherwise (bit cleared), the decode function of the Pause frame is disabled.

- TFE

  (In Full-Duplex mode) Setting this bit, the MAC enables the flow control operation to transmit Pause frames. Otherwise (bit cleared), no Pause frames will not be transmitted by MAC.

  (In Half-Duplex mode) Setting this bit, the MAC enables the back-pressure operation. Otherwise (bit cleared), back-pressure feature is disabled.

- FCB/BPA

  Setting this bit, a Pause Control frame is initiated in Full-Duplex mode and the back-pressure function is activated in Half-Duplex mode (if TFE bit above is set).

Note: (In Full-Duplex mode) During a transfer of the Control Frame, this bit will continue to be set meaning that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC will clear this bit.

24.7.23 VLAN tag register (Register7, MAC)

The VLAN tag is a register which contains the IEEE 802.1Q VLAN tag to identify the VLAN frames. The MAC compares the 13th and the 14th bytes of the receiving frames (length/type) with 0x8100, and the following 15th and 16th bytes with the VLAN tag: if a match occurs, the MAC sets the VLAN bit in the received frame status word (receive descriptor 0,). The GMII data bit assignments are given in *Table 453*.

Table 453. VLAN tag register bit assignments

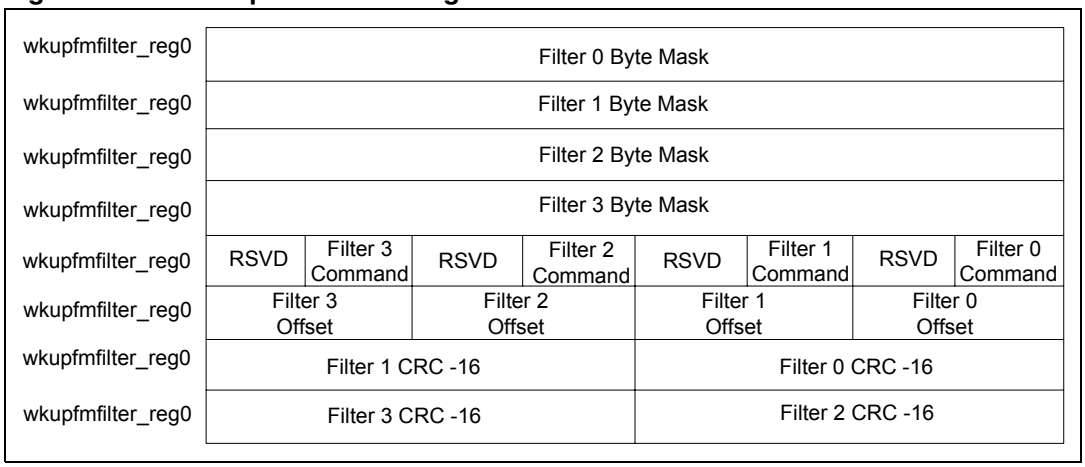
Bit	Name	Reset value	Type	Description
[31:16]	Reserved	-	RO	Read: undefined.
[15:00]	VL	16'h0	RW	VLAN tag identifier.

24.7.24 Wake-up frame filter register (Register10, MAC)

This register is actually a 32 bit pointer used by the application to access (read/write) eight (not transparent) Wake-up Frame Filter registers, reported in *Figure 55*, involved in the power management (PMT, see *Section 24.7.25*).

It means that eight sequential Write operations to this address will write all wake-up frame filter registers, and eight sequential read operations from this address will read all Wake-up Frame Filter registers.

Figure 55. Wake-up frame filter registers



Four programmable filters (filter 0 to filter 3) are available to support four different receive frame patterns. The corresponding 32 bit byte mask registers allow to define which bytes of the frame are checked by the filter to determine whether or not the frame is a wake-up frame. The MSB (bit [31]) must be 1'b0. If a bit j ([30:0]) is set, then the (offset + j) byte of the incoming frame is processed by the CRC block.

As many as 4 bit Command registers control the operation of relevant filter, see *Table 454*.

Table 454. 4 bit command registers

Bit	Description
[03]	Setting this bit, filter applies only to multicast frames, otherwise to unicast frames only.
[02]	Reserved
[01]	Reserved
[00]	Setting this bit, the relevant filter is enabled.

Moreover, the 8 bit Offset registers define the offset (within the frame) which point the first byte of the frames to be examined by the filter. The minimum allowed is 12. At last, the four 16 bit CRC registers contain the 16 bit CRC value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

If the incoming frame passes the address filtering set by the command register, and if the CRC-16 matches the incoming examined pattern, then it means that a wake-up frame is received.

24.7.25 PMT control and status register (Register11, MAC)

The PMT (Power Management) Control And Status Register (CSR) is intended to program the request wake-up events and to monitor the wake-up events as part of the power management mechanism supported by the MAC.

The PMT CSR bit assignments are given in *Table 455*.

Table 455. PMT CSR bit assignments

Bit	Reset value	Type	Description
[31]	1'h0	RW	Wake-up frame filter register pointer reset. If set, it resets the remote wake-up frame filter pointer to 3'b000 (eight remote wake-up registers are present). It is automatically cleared after 1 clock cycle.
[30:10]	-	RO	Reserved. Read: undefined.
[09]	1'h0	RW	Global unicast. If set, it enables any unicast packet filtered by MAC address recognition (DAF) to be a wake-up frame.
[08:07]	-	RO	Reserved. Read: undefined.
[06]	1'h0	RW	Wake-up frame received. If set, it indicates that the power management event was generated due to the reception of a wake-up frame. This bit is cleared by a read into this register.
[05]	1'h0	RW	Magic packet received. If set, it indicates that the power management event was generated due to the reception of a magic packet. This bit is cleared by a Read into this register.
[04:03]	-	RO	Reserved. Read: undefined.
[02]	1'h0	RW	Wake-up frame enable. If set, it enables generation of a power management event due to wake-up frame reception.
[01]	1'h0	RW	Magic packet enable. If set, it enables generation of a power management event due to magic packet reception.
[00]	1'h0	RW	Power down. If set, all received frames will be dropped. This bit is automatically cleared when a wake-up frame or a magic packet is received, and the power-down mode is disabled. Note that this bit should be set only when either wake-up frame enable (bit [2]) or magic packet enable (bit [1]) are set.

24.7.26 Interrupt status register (Register 14, MAC)

The Interrupt Status Register contents identify the events in the MAC-CORE that can generate interrupt.

Table 456. Interrupt status register bit assignments

Bit	Reset Value	Type	Description
[31:16]			Reserved
[15:05]	-	RO	Reserved

Table 456. Interrupt status register bit assignments (continued)

Bit	Reset Value	Type	Description
[04]	1'h0	RO	MMC Interrupt Status This bit is set high whenever an interrupt is generated in the MMC Interrupt register (see section MMC Receive Interrupt Register). This bit is cleared whenever the bit in the interrupt register is cleared.
[03]	1'h0	RO	PMT Interrupt Status This bit is set whenever a Magic packet or Wake-on-Lan frame is received in the Power-down mode (refer to bit 5 and 6 in PMT Control and Status Register (Register11, MAC)
[02:00]	-	RO	Reserved

24.7.27 Interrupt mask register (Register 15, MAC)

The interrupt Mask Register bits enable the user to mask the interrupt signal due to the corresponding event in the Interrupt Status Register. The interrupt signal is sbd_intr_o.

Table 457. Interrupt mask register bit assignments

Bit	Reset value	Type	Description
[31:16]			Reserved
[15:04]	-	RO	Reserved
[03]	1'h0	RW	PMT interrupt mask This bit when set, will disable the assertion of the interrupt signal due to the setting of PMT interrupt status bit in Register 14.
[02:00]	-	RO	Reserved. Read: undefined

24.7.28 MAC address0 high register (Register16, MAC)

The MAC address0 High is a register which contains the upper 16 bits ([47:32]) of the 6-byte first MAC address of the station. The MAC address0 High bit assignments are given in *Table 458*.

Table 458. MAC address0 high register bit assignments

Bit	Name	Reset value	Type	Description
[31]	MO	1'h1	RO	Always set to 1'b1.
[30:16]	Reserved	-	RO	Read: undefined.
[15:00]	A[47:32]	16'hFFFF	RW	MAC address0 [47:32].

24.7.29 MAC address0 low register (Register17, MAC)

The MAC address0 Low is a register which contains the lower 32 bits ([31:00]) of the 6-byte first MAC address of the station. The MAC Address0 Low bit assignments are given in *Table 459*.

Table 459. MAC Address0 low register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	A[31:0]	32'hFFFFFFFF	RW	MAC address0 [31:00].

24.7.30 MAC address1 high register (Register18, MAC)

The MAC Address1 High is a register which contains the upper 16 bits ([47:32]) of the 6-byte 2nd MAC address of the station. The MAC Address1 high bit assignments are given in *Table 460*.

Table 460. MAC Address1 high register bit assignments

Bit	Name	Reset Value	Type	Description
[31]	*AE*	1'h0	RW	Address Enable.
[30]	*SA*	1'h0	RW	Source Address.
[29:24]	*MBC*	6'h0	RW	Mask Byte Control.
[23:16]	Reserved	-	RO	Read: undefined.
[15:00]	A[47:32]	16'hFFFF	RW	MAC Address1 [47:32].

- AE

  Setting this bit, the MAC address filtering module uses the 2nd MAC address for perfect filtering.
- SA

  This bit allows to specify whether the MAC Address1 [47:0] is used to compare with the SA fields (SA is 1'b1) or with the DA fields (SA is 1'b0) of the received frame.
- MBC

  This 6 bit field controls masking of each of the MAC address byte, according to encoding below:

Table 461. MAC address byte

Bit	MAC Address Byte
[29]	Register18[15:8]
[28]	Register18[7:0]
[27]	Register19[31:24]
[26]	Register19[23:16]
[25]	Register19[15:8]
[24]	Register19[7:0]

Setting a bit, the corresponding byte of the received SA/DA is not compared with the contents of MAC Address1 registers.

24.7.31 MAC address1 low register (Register19, MAC)

The MAC Address1 Low is a register which contains the lower 32 bits ([31:0]) of the 6-byte 2nd MAC address of the station. The MAC address1 Low bit assignments are given in *Table 462*.

Table 462. MAC Address1 low register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	A[31:0]	32'hFFFFFFFF	RW	MAC address1 [31:0].

Note: 1 The description for registers20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44 and 46 (MAC address2 High through MAC Address15 High) is the same as for the register18 (MAC address1 High).

2 The description for registers21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45 and 47 (MAC address2 low through MAC address15 low) is the same as for the register19 (MAC address1 low).

24.8 MMC registers

As reported in *Table 423*, the address space of MAC CSRs ranging from 0x0100 to 0x01FC (Register64 to Register127) hosts the MMC (MAC management counters) registers.

The MMC unit of MAC maintains a set of 32 bit registers for gathering statistics on received and transmitted frames (that is., number of bytes transmitted, number of good and bad frames transmitted, number of frames received with CRC error, and so on).

These MMC registers also include a control register (Register64, mmc_cntrl), two registers containing interrupts generated, both receive and transmit (Register65 and Register66, mmc_intr_rx and mmc_intr_tx respectively), and two registers containing masks for these interrupts (Register67 and Register68, mmc_intr_mask_rx and mmc_intr_mask_tx respectively).

A descriptions of the five MMC registers is given in the following Sections.

24.8.1 MMC control register

The MMC control register is responsible of the operating mode of the management counters.

Table 463. MMC control register bit assignments

Bit	Name	Reset value	Type	Description
[31:03]	Reserved	-	RO	-
[02]	ROR	1'h0	RO	Reset on read. 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.

Table 463. MMC control register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[01]	CSR	1'h0	RO	Counter stop rollover. When set, counter after reaching maximum value will not roll over to zero.
[00]	CR	1'h0	RO	Counters reset. When set, all counters will be reset. This bit will be cleared automatically after 1 clock cycle.

24.8.2 MMC receive interrupt register

The MMC receive interrupt register maintains the interrupts generated when receive statistic counters reach half their maximum values. (MSB of the counters is set.) It is a 32 bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte-lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Table 464. MMC receive interrupt register bit assignments

Bit	Name	Reset value	Type	Description
[31:24]	Reserved	-	RO	-
[23]	-	1'h0	-	The bit is set when the rxwatchdog error counter reaches half the maximum value.
[22]	-	1'h0	-	The bit is set when the rxvlanframes_gb counter reaches half the maximum value.
[21]	-	1'h0	-	The bit is set when the rxfifooverflow counter reaches half the maximum value.
[20]	-	1'h0	-	The bit is set when the rxpauseframes counter reaches half the maximum value.
[19]	-	1'h0	-	The bit is set when the rxoutofrangetype counter reaches half the maximum value.
[18]	-	1'h0	-	The bit is set when the rxlengtherror counter reaches half the maximum value.
[17]	-	1'h0	-	The bit is set when the rxunicastframes_gb counter reaches half the maximum value.
[16]	-	1'h0	-	The bit is set when the rx1024tomaxoctects_gb counter reaches half the maximum value.
[15]	-	1'h0	-	The bit is set when the rx512to1023octects_gb counter reaches half the maximum value.
[14]	-	1'h0	-	The bit is set when the rx216to511octects_gb counter reaches half the maximum value.
[13]	-	1'h0	-	The bit is set when the rx128to255octects_gb counter reaches half the maximum value.
[12]	-	1'h0	-	The bit is set when the rx64to127octects_gb counter reaches half the maximum value.
[11]	-	1'h0	-	The bit is set when the rx64octects_gb counter reaches half the maximum value.

Table 464. MMC receive interrupt register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[10]	-	1'h0	-	The bit is set when the rxoversize_g counter reaches half the maximum value.
[09]	-	1'h0	-	The bit is set when the rxundersize_g counter reaches half the maximum value.
[08]	-	1'h0	-	The bit is set when the rxjabbererror counter reaches half the maximum value.
[07]	-	1'h0	-	The bit is set when the rxrunterror counter reaches half the maximum value.
[06]	-	1'h0	-	The bit is set when the rxalignmenterror counter reaches half the maximum value.
[05]	-	1'h0	-	The bit is set when the rxcrcerror counter reaches half the maximum value.
[04]	-	1'h0	-	The bit is set when the rxmulticastframes_g counter reaches half the maximum value.
[03]	-	1'h0	-	The bit is set when the rxbroadcastframes_g counter reaches half the maximum value.
[02]	-	1'h0	-	The bit is set when the rxoctectcount_g counter reaches half the maximum value.
[01]	-	1'h0	-	The bit is set when the rxoctectcount_gb counter reaches half the maximum value.
[00]	-	1'h0	-	The bit is set when the rxframecount_gb counter reaches half the maximum value.

24.8.3 MMC transmit interrupt register

The MMC transmit interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values. (MSB of the counters is set.) It is a 32 bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte-lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Table 465. MMC transmit interrupt register bit assignments

Bit	Name	Reset value	Type	Description
[31:25]	Reserved	-	RO	-
[24]	-	1'h0	-	The bit is set when the txvlanframes_g counter reaches half the maximum value.
[23]	-	1'h0	-	The bit is set when the txpauseframes error counter reaches half the maximum value.
[22]	-	1'h0	-	The bit is set when the txoexcessdef counter reaches half the maximum value.
[21]	-	1'h0	-	The bit is set when the txframecount_g counter reaches half the maximum value.

Table 465. MMC transmit interrupt register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[20]	-	1'h0	-	The bit is set when the txoctectcount_g counter reaches half the maximum value.
[19]	-	1'h0	-	The bit is set when the txcarriererror counter reaches half the maximum value.
[18]	-	1'h0	-	The bit is set when the txexesscol counter reaches half the maximum value.
[17]	-	1'h0	-	The bit is set when the txlatecol counter reaches half the maximum value.
[16]	-	1'h0	-	The bit is set when the txdeferred counter reaches half the maximum value.
[15]	-	1'h0	-	The bit is set when the txmulticol_g counter reaches half the maximum value.
[14]	-	1'h0	-	The bit is set when the txsinglecol_g counter reaches half the maximum value.
[13]	-	1'h0	-	The bit is set when the txunderflowerror counter reaches half the maximum value.
[12]	-	1'h0	-	The bit is set when the txbroadcastframes_gb counter reaches half the maximum value.
[11]	-	1'h0	-	The bit is set when the txmulticastframes_gb counter reaches half the maximum value.
[10]	-	1'h0	-	The bit is set when the txunicastframes_gb counter reaches half the maximum value.
[09]	-	1'h0	-	The bit is set when the tx1024tomaxoctects_gb counter reaches half the maximum value.
[08]	-	1'h0	-	The bit is set when the tx512to1023octects_gb counter reaches half the maximum value.
[07]	-	1'h0	-	The bit is set when the tx256to511octects_gb counter reaches half the maximum value.
[06]	-	1'h0	-	The bit is set when the tx128to255octects_gb counter reaches half the maximum value.
[05]	-	1'h0	-	The bit is set when the tx65to127octects_gb counter reaches half the maximum value.
[04]	-	1'h0	-	The bit is set when the tx64to127octects_gb counter reaches half the maximum value.
[03]	-	1'h0	-	The bit is set when the txmulticastframes_g counter reaches half the maximum value.
[02]	-	1'h0	-	The bit is set when the txbroadcastframes_g counter reaches half the maximum value.
[01]	-	1'h0	-	The bit is set when the txframecount_gb counter reaches half the maximum value.
[00]	-	1'h0	-	The bit is set when the txoctectcount_gb counter reaches half the maximum value.

24.8.4 MMC receive interrupt mask register

The MMC receive interrupt mask register maintains masks for the interrupts generated when receive statistic counters reach half their maximum values. (MSB of the counters is set.) It is a 32 bit wide register.

Table 466. MMC receive interrupt mask register bit assignments

Bit	Name	Reset value	Type	Description
[31:24]	Reserved	-	RO	-
[23]	-	1'h0	RW	Setting this bit masks the interrupt when the rxwatchdog counter reaches half the maximum value.
[22]	-	1'h0	RW	Setting this bit masks the interrupt when the rxvlanframes_gb counter reaches half the maximum value.
[21]	-	1'h0	RW	Setting this bit masks the interrupt when the rxfifooverflow counter reaches half the maximum value.
[20]	-	1'h0	RW	Setting this bit masks the interrupt when the rxpauseframes counter reaches half the maximum value.
[19]	-	1'h0		The bit is set when the rxcarriererror counter reaches half the maximum value.
[18]		1'h0		The bit is set when the rxexesscol counter reaches half the maximum value.
[17]		1'h0		The bit is set when the rxlatecol counter reaches half the maximum value.
[16]		1'h0		The bit is set when the rxdeferred counter reaches half the maximum value.
[15]		1'h0		The bit is set when the rxmulticol_g counter reaches half the maximum value.
[14]		1'h0		The bit is set when the rxsinglecol_g counter reaches half the maximum value.
[13]		1'h0		The bit is set when the rxunderflowerror counter reaches half the maximum value.
[12]		1'h0		The bit is set when the rxbroadcastframes_gb counter reaches half the maximum value.
[11]		1'h0		The bit is set when the rxmulticastframes_gb counter reaches half the maximum value.
[10]		1'h0		The bit is set when the rxunicastframes_gb counter reaches half the maximum value.
[09]		1'h0		The bit is set when the rx1024tomaxoctects_gb counter reaches half the maximum value.
[08]		1'h0		The bit is set when the rx512to1023octects_gb counter reaches half the maximum value.
[07]		1'h0		The bit is set when the rx256to511octects_gb counter reaches half the maximum value.

Table 466. MMC receive interrupt mask register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[06]		1'h0		The bit is set when the rx128to255octects_gb counter reaches half the maximum value.
[05]		1'h0		The bit is set when the rx65to127octects_gb counter reaches half the maximum value.
[04]		1'h0		The bit is set when the rx64to127octects_gb counter reaches half the maximum value.
[03]		1'h0		The bit is set when the rxmulticastframes_g counter reaches half the maximum value.
[02]		1'h0		The bit is set when the rxbroadcastframes_g counter reaches half the maximum value.
[01]	-	1'h0	RW	Setting this bit masks the interrupt when the rxoctectcount_gb counter reaches half the maximum value.
[00]	-	1'h0	RW	Setting this bit masks the interrupt when the rxframecount_gb counter reaches half the maximum value.

24.9 Clocks with MII

The clocking scheme for the MAC-AHB is shown in *Figure 56*. The MAC-AHB uses three clock inputs for normal operation with the MII interface:

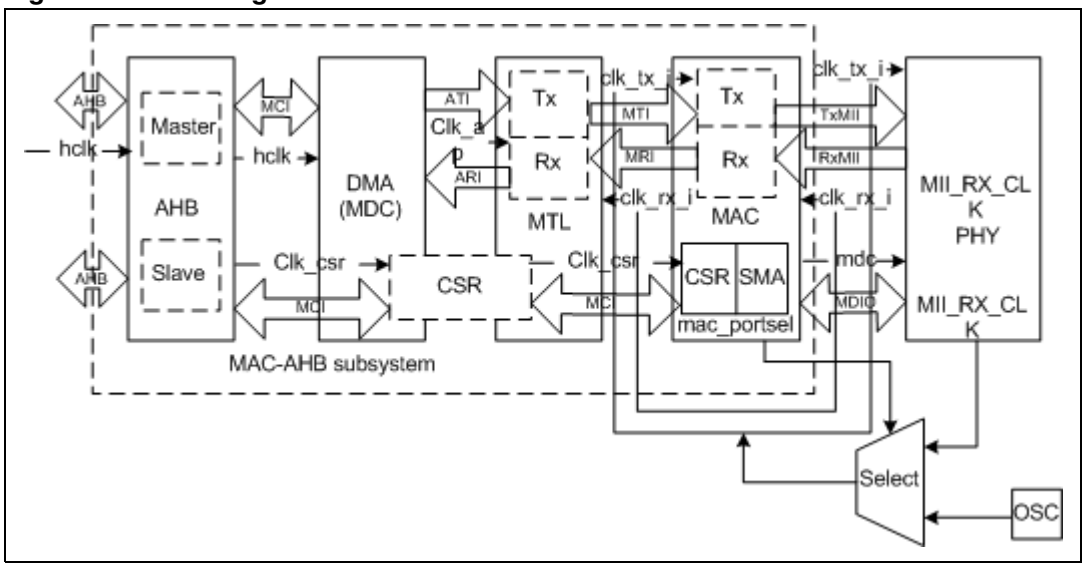
- One clock (clk_tx_i) for transmission functions
- One clock (clk_rx_i) for reception functions
- One application clock (hclk_i)

All clocks except the Application clock are generated from an external PHY. The Application clock is used for the control interface of the MAC-AHB. Transmit clock clk_tx_i is used for the transmission function of the MAC-AHB. Similarly, receive clock clk_rx_i is used for the receive function of the MAC-AHB. The transfer of data from/to the application clock domain to the Transmit and Receive clock domains is performed in the MTL module.

Clock clk_tx_ i gets input from an external PHY (MII mode).

The external PHY outputs a 25 MHz or 2.5 MHz transmit clock on MII_CLK when it operates at 100 Mbps or 10 Mbps, respectively. The frequency of clock clk_rx_i is same as receive clock MII_RX_CLK generated by the external PHY. The PHY outputs a 25 MHz or 2.5 MHz receive clock on MII_RX_CLK when it operates at 100 Mbps, or 10 Mbps, respectively.

Figure 56. Clocking scheme for MAC-AHB



25 LS_JPEG codec

25.1 Overview

Within its Low Speed Connectivity Block, the device provides a **JPEG Codec** with header processing which is built around existing JPEG ECS CODEC and extends its functionality by providing additional support for JPEG Header parsing and generation.

The encoding process compresses 8x8 pixel blocks (data units) into either a complete JPEG encoded output stream or only ECS data depending on whether the header processing functionality of the core is enabled.

The decoding process can either decode a complete JPEG encoded data stream or an input data stream with only ECSs. In either case, the core decodes the ECS data into valid 8 x 8 pixel blocks (data units).

Main features of the JPGC are:

- Compliance with the baseline JPEG standard (ISO/IEC 10918-1).
- Single-clock per pixel encoding/decoding.
- Support for up to four channels of component color.
- 8 bit/channel pixel depths.
- Programmable quantization tables (up to four).
- Programmable Huffman tables (two AC and two DC).
- Programmable Minimum Coded Unit (MCU).
- Configurable JPEG headers processing.
- Support for restart marker insertion.
- Use of two DMA channels (from the Basic Subsystem) and of two 8 x 32 bits FIFO's (local to the JPGC) for efficient transferring and buffering of encoded/decoded data from/to the Codec Core.

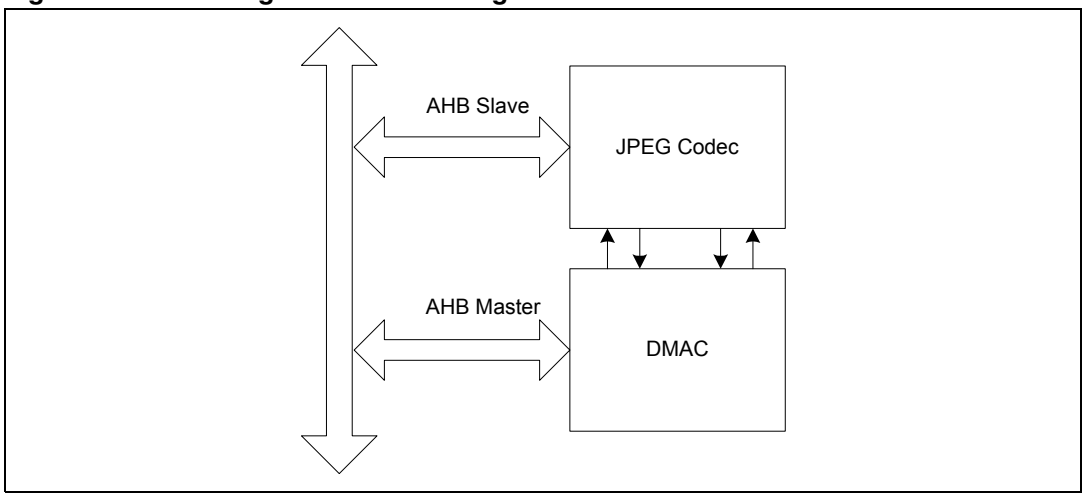
25.2 Signal interfaces

The JPGC directly interfaces with the signals summarized in *Table 467*. The JPGC is connected as a slave on the AHB bus, and has two DMA channels asserted to itself. A functional diagram of these signal interfaces is given in *Figure 57*.

Table 467. GPIO signal interface

Group	Signal name	Direction	Size (bit)	Description
AHB Slave	-	Input/Output	-	See AMBA specification.
DMAC	-	Input/Output	-	See *Chapter 19: BS_DMA controller*

Figure 57. JPGC signal interfaces diagram

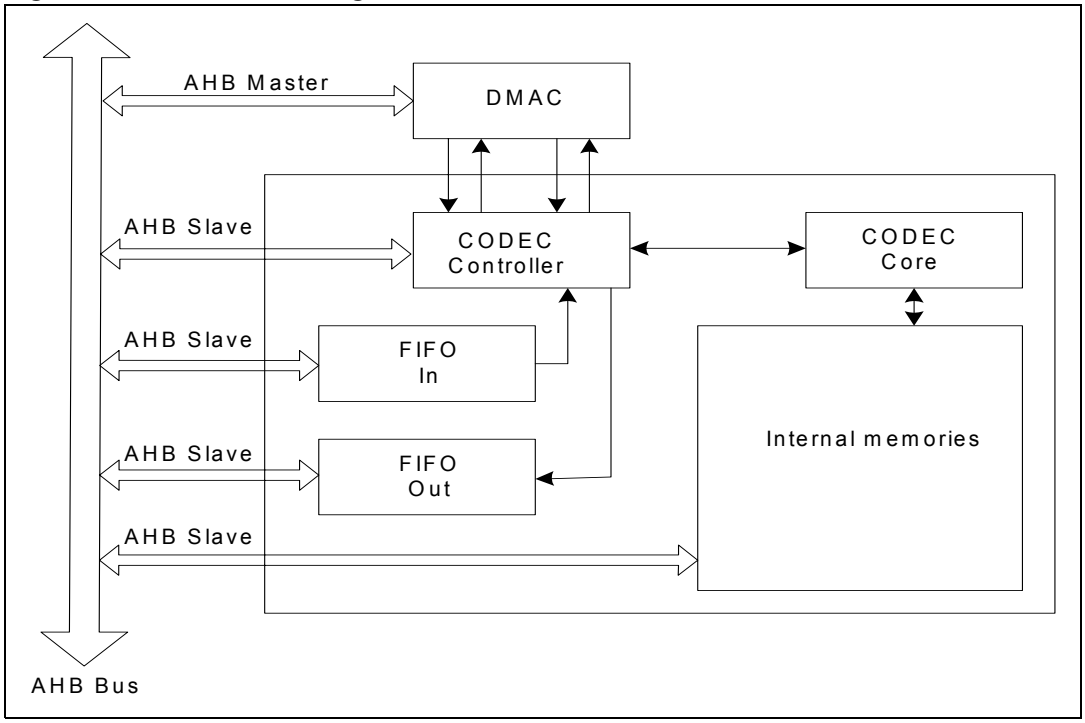


25.3 Functional description

25.3.1 Block diagram

The block diagram of the JPGC is shown in *Figure 58*.

Figure 58. JPGC block diagram



25.3.2 Main functions description

As one can see from the block diagram (*Figure 58*), the main building blocks of the JPGC are five:

- The Codec Core;
- The Codec Controller;
- The DMA controller (DMAC);
- The FIFO buffers (FIFO in and FIFO out);
- The Internal Memories.

A general description follows each of these blocks, while a comprehensive discussion of their internal registers from the programmer's standpoint is the object of *Section 25.4.1*.

25.3.3 Codec core

The *Codec Core* implements all the steps necessary to encode and decode image data according to the JPEG baseline algorithm as specified in *ISO/IEC 10918-1*. It is specifically designed to accelerate entropy-coded segment (ECS) encoding and decoding, because this forms the most computing-intensive part of the baseline JPEG algorithm.

The Codec Core can enable/disable header processing. If disabled, only the ECS data are generated/decoded. Support for restart markers is also provided: the Codec Core recognizes them in the encoded stream when decoding, and can optionally insert them when encoding.

JPEG encoded data streams decoded by the Codec Core must be compliant with the interchange format syntax specified in the ISO/IEC 10918-1. Also JFIF images, the de facto standard used to encoded JPEG images, is supported.

Before any coding process can start, the codec core, together with the *DMAC* and the Internal Memories, must be programmed, by writing to the corresponding registers.

The Codec Core receives from the *FIFO in* buffer its input data, which can be either a sequence of *Minimum Coded Units* (MCU) (if the JPGC is used as an encoder from YUV MCU's to JPEG) or a stream of *Entropy Coded Segments* (ECS) (if the JPGC is used as a decoder from JPEG to YUV MCU's). Conversely, output data from the codec core are sent to the *FIFO Out* buffer as an ECS stream (resp. MCU sequence), whenever the JPGC is working as an encoder (resp. decoder).

25.3.4 Codec controller

The *Codec Controller* manages the data flow between the Codec Core and the FIFO buffers between the FIFO buffers and the external RAM. In order to accomplish the latter task, it uses the *DMAC* to perform fast data transfers.

Due to area optimization of the JPGC block, encoding and decoding operations performed by the Codec Core cannot be simultaneous. Thus the Codec Controller is in charge to assure that only a given data path (JPEG data from RAM -> JPGC -> YUV MCU data to RAM, or the opposite) is active at a certain time.

25.3.5 DMAC

The *DMA controller* is exploited by the codec controller to perform fast data transfers from/to external RAM to/from the internal FIFO buffers. The DMAC has to be programmed with the

correct transfer parameters, before any coding process can start. See also *Chapter 19: BS_DMA controller*, for an in-depth description of the direct memory access block.

25.3.6 FIFO buffers

These two *First-in First-out* buffers have a word width of 32 bits, and a depth of 8 words. FIFO's are used by the Codec Controller to bufferize the flow of data incoming to (FIFO In) and outcoming from (FIFO out) the Codec Core. Each FIFO is accessed by reading/writing always from/to the same address.

25.3.7 Internal memories

These memories have to be programmed, before the encoding process can start, with the tables needed by the baseline JPEG algorithm (see *ISO/IEC 10918-1*).

Up to four *quantization tables* are used for both encoding and decoding. DHTMem and *HuffEnc* memories are used for encoding. *HuffMin, HuffBase* and *HuffSymb* memories are used for decoding.

25.4 Programming model

25.4.1 Register map

The JPGC can be fully configured by programming its 32 bits wide registers, which can be accessed through the AHB slave interface at the base address 0xD080_0000.

An overview of the JPGC memory map is shown in *Table 468*.

JPGC registers can be logically arranged in five groups, each one referring to the corresponding main block of the JPGC (see also *Section 25.3.2* Main Functions Description):

- Codec core registers (listed in *Table 469*).
- Codec controller registers (listed in *Table 470*).
- DMAC registers (refer to *Section 22.6.2: UHC interrupts* in *Chapter 22: HS_USB2.0 host*)
- FIFO registers (listed in *Table 471*).
- Internal memories (listed in *Table 472*).

A detailed description of all the JPGC registers is given in *Section 25.4.2: Register description*.

Table 468. JPGC memory map

Name	Base address
Codec Core	0x0000
Codec Controller	0x0200
FIFO In	0x0400
FIFO Out	0x0600
Quantization Memory	0x0800
HuffMin Memory	0x0C00

Table 468. JPGC memory map (continued)

Name	Base address
HuffBase Memory	0x1000
HuffSymb Memory	0x1400
DHTMem Memory	0x1800
HuffEnc Memory	0x1C00

Table 469. JPGC codec core registers

Name	Offset	Type	Reset value	Description
JPGCReg0	0x00	WO	32'h0	Codec Core Register 0
JPGCReg1	0x04	RW	32'h0	Codec Core Register 1
JPGCReg2	0x08	RW	32'h0	Codec Core Register 2.
JPGCReg3	0x0C	RW	32'h0	Codec Core Register 3.
JPGCReg4	0x10	RW	32'h0	Codec Core Register 4.
JPGCReg5	0x14	RW	32'h0	Codec Core Register 5.
JPGCReg6	0x18	RW	32'h0	Codec Core Register 6.
JPGCReg7	0x1C	RW	32'h0	Codec Core Register 7.

Table 470. JPGC codec controller registers

Name	Offset	Type	Reset value	Description
JPGC Control Status	0x00	RW	32'h0	Codec controller status.
JPGC Bytes From Fifo to Core	0x04	RO	32'h0	Number of bytes from FIFO in to Codec Core.
JPGC Bytes From Core to Fifo	0x08	RO	32'h0	Number of bytes from Codec Core to FIFO out.
JPGC Burst Count Before Init	0x0C	RW	32'h0	Number of burst transfer send by TX FIFO before Interrupt.

Table 471. JPGC FIFO registers

Name	Offset	Type	Reset value	Description
JPGC FifoIn	0x0400	RW	32'h0	FIFO in register.
JPGC FifoOut	0x0600	RW	32'h0	FIFO out register.

Table 472. JPGC internal memories

Name	Offset	Type	Reset value	Description
JPGCQMem	0x0800	RW	-	Quantization Table Memory.
JPGCHuffMin	0x0C00	RW	-	Huffmin Table Memory.
JPGCHuffBase	0x1000	RW	-	Huffbase Table Memory.
JPGCHuffSymb	0x1400	RW	-	Huffsymb Table Memory.
JPGCDHTMem	0x1800	RW	-	Dht Marker Segment Memory.
JPGCHuffEnc	0x1C00	RW	-	Huffenc Table Memory.

25.4.2 Register description

25.4.3 JPGCreg0 register

This register is used to start and stop the coding process. It is intended to be a write-only register. reading it always return 0. All other registers must be programmed before the JPGC is started using this register.

Table 473. JPGCreg0 register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read as zero.
[00]	*StartStop*	1'h0	Write: coding process start/stop. Read as zero.

StartStop

When this bit is set, the coding process starts. Clearing this bit during the coding process has the effect of stopping the process itself.

25.4.4 JPGCReg1 register

This register defines several parameters for the image format and the coding process.

Table 474. JPGCreg1 register bit assignments

Bit	Name	Reset Value	Description
[31:16]	*Ysiz*	-	Number of lines.
[15:09]	Reserved	-	
[08]	*Hdr*	-	Header processing enable.
[07:06]	*Ns*	-	Number of components for scan header marker segment minus 1.
[05:04]	*colspctype*	-	Number of quantization tables in the output stream.
[03]	*De*	-	Decoding/encoding.
[02]	*Re*	-	Restart marker processing enable.
[01:00]	*Nf*	-	Number of color components minus 1.

- Ysiz

  Number of lines in source image; values can range from 0 to 65,535.
- Hdr

  When set, this bit enables the JPEG headers processing (generation or parsing, depending if the encoding or decoding behavior is selected).
- Ns

  Number of components for scan header market segment minus 1; there can be from 1 to 4 components.
- colspctype

  The number of quantization tables to insert in the output stream minus one. 0 = Grayscale, 1 = YUV, 2 = RGB, 3= CMYK.
- De

  Selects encoding or decoding behavior. When the bit is set, the CODEC acts as a decoder, otherwise, it works as an encoder.
- Re

  When set, this bit enables restart marker processing. The ECS encoder inserts restart markers every (NRST + 1) minimum coded units.
- Nf

  Number of color components in the source image minus 1; there can be from 1 to 4 components. For example, in a grayscale image Nf = 0; for a RGB or YUV image Nf = 2.

25.4.5 JPGCreg2 register

This register defines the number of minimum coded units which are to be encoded by the codec core.

Table 475. JPGCreg2 register bit assignments

Bit	Name	Reset Value	Description
[31:26]	Reserved	-	
[25:00]	*NMCU*	-	Number of MCU's minus 1.

- NMCU

  This value defines the number of minimum coded units to be coded, minus 1; there can be from 0 to 67,108,863 MCU's.

25.4.6 JPGCreg3 register

This register defines a couple of parameters for the image format and the coding process.

Table 476. JPGCReg3 register bit assignments

Bit	Name	Reset Value	Description
[31:16]	*Xsiz*	-	Number of pixels per line.
[15:00]	*NRST*	-	Number of MCU's between two restart markers minus 1.

- Xsiz

  Number of pixels per line of the image; a single line's length can range from 0 to 65,535 pixels.

- NRST

  Number of minimum coded units between two consecutive restart markers, minus 1. This value is ignored if the Re bit in JPGC Reg1 is not set.

25.4.7 JPGCreg4-7 register

These registers describe the composition of a Minimum Coded Unit (MCU).

As specified in the ISO document for the baseline algorithm (see ISO/IEC 10918-1), up to four color components can be encoded in a single ECS. Accordingly, these registers contain four sections, one for each color component I = 0, 1, 2, 3

Table 477. JPGCreg4-7 register bit assignments

Bit	Name	Reset Value	Description
[31:16]	Reserved	-	
[15:12]	*Vi*	-	Vertical sampling factor for component *i*.
[11:08]	*Hi*	-	Horizontal sampling factor for component *i*.
[07:04]	*Nblocki*	-	Number of data units of the component *I* contained in a MCU, minus 1.
[03:02]	*QTi*	-	Quantization table used for component *i*.
[01]	*HAi*	-	AC Huffman table used for component *i*.
[00]	*HDi*	-	DC Huffman table used for component *i*.

- **Vi**

  This value defines the vertical sampling factor for the color component I; value can range from 1 to 4.
- Hi

  This value defines the horizontal sampling factor for the color component I; value can range from 1 to 4.
- Nblocki

  Number of data units (i.e. 8 x 8 blocks of data) of the color component I contained in a MCU, minus 1. The range of possible values for Nblocki is 0-15, because 4 bits are set aside for this field. However, it is important to note that according to the ISO specification, in the case of the baseline algorithm, the following relation must hold:
  $^{Nf}\Sigma_{i=0}(Nblocki+1) \leq 10$

- **QTi**

  This value defines the quantization table to be used for the color component i. Since four quantization tables are possible, 2 bits are sufficient for this field.
- HAi

  This value defines the Huffman table to be used for the encoding of the AC coefficients in the data units belonging to the color component i. Since only two AC tables are allowed in the baseline algorithm, 1 bit is sufficient for this field.
- HDi

  This value defines the Huffman table to be used for the encoding of the DC coefficients in the data units belonging to the color component i. Since only two DC tables are allowed in the baseline algorithm, 1 bit is sufficient for this field.

25.4.8 JPGC control status register

This register contains the status of the codec controller. The bit 0 (interrupt bit) is automatically set when a coding process has finished.

Table 478. JPGC control status register bit assignments

Bit	Name	Reset value	Description
[31]	*EOC*	-	End of Conversion (Active High)
[30]	*SCR*	-	Synchronous Core Reset (Active High). Write only field. Writing 1 on this bit will reset & disable both CODEC and controller. Clear this bit to enable CODEC.
[29:18]	Reserved	-	
[17:03]	*LLI*	-	Number of LLI (DMA parameter). This field is only writable and not readable
[02:01]	*BNV*	-	Number of bytes not valid in last word.
[00]	*INT*	-	Interrupt bit. It is possible to write only 0 in this bit to clear the interrupt bit. It is wrong to write 1 in this field.

- EOC

  End of Conversion (Active High)
- SCR

  Synchronous Core Reset (Active High.
- LLI

  Number of LLI (DMA parameter).for input data has to be programmed with.
- BNV

  Number of bytes not valid in last word: if the total byte number is not an exact multiple of 4, it will happen that 1, 2 or 3 bytes in the last 4-byte word will be meaningless.
- INT

  Interrupt bit. Only a 0 can be written to this bit, having the effect of clearing the interrupt bit. Trying to write a 1 to this bit will result in an unpredictable behavior.

25.4.9 JPGC bytes from Fifo to core register

This register contains the number of bytes that have been sent, at a given time, from the FIFO In buffer to the codec core. The content of this register is cleared automatically when a new coding process starts.

Table 479. JPGC bytes from Fifo to core register bit assignments

Bit	Name	Reset value	Description
[31:00]	*NRX*	-	Number of bytes from FIFO in to Codec Core.

- NRX

  Number of bytes sent from FIFO In to the Codec Core. This register is cleared when a new encoding process starts.

25.4.10 JPGC bytes from core to Fifo register

This register contains the number of bytes that have been sent, at a given time, from the codec core to the FIFO Out buffer. The content of this register is cleared automatically when a new coding process starts.

Table 480. JPGC bytes from core to Fifo register bit assignments

Bit	Name	Reset Value	Description
[31:00]	*NTX*	-	Number of bytes from codec core to FIFO out.

- NTX

  Number of bytes sent from the Codec Core to FIFO Out. This register is cleared when a new encoding process starts.

25.4.11 JPGC bust count beforeInit

This register contains the number of burst transfer sent by TX FIFO before controller will set interrupt. It's ignored if burst count ENABLE bit is 0.

Table 481. JPGCbust Count before Init register bit assignments

Bit	Name	Reset Value	Description
[31]	*EN*	-	Burst Count ENABLE, active high
[30:00]	*BTF*	-	Number of burst transfer send by TX FIFO before interrupt.

- EN

  Burst Count Enable, Active High.
- BTF

  Number of burst transfer sent by TX FIFO before controller will set interrupt.

25.4.12 DMAC registers

See *Chapter 19: BS_DMA controller*, for a detailed description of the DMAC registers.

25.4.13 JPGCFifoIn register

This register is used to read data from, or write data to, the FIFO In, which is used to bufferize the transfers from the external RAM to the codec core, under the control of the codec controller.

Table 482. JPGC Fifo in register bit assignments

Bit	Name	Reset Value	Description
[31:00]	*DATA*	-	FIFO data.

- DATA
  Data read from, or written to, the FIFO In buffer.

25.4.14 JPGCFifoOut register

This register is used to read data from, or write data to, the FIFO out, which is used to bufferize the transfers from the codec core to the external RAM, under the control of the codec controller.

Table 483. JPGC Fifo out register bit assignments

Bit	Name	Reset value	Description
[31:00]	*DATA*	-	FIFO data.

- DATA
  Data read from, or written to, the FIFO In buffer.

25.4.15 JPGCqmem memory

This memory is used to store the *quantization tables* used by the codec core.

As specified in the ISO documentation, in the case of the baseline algorithm, up to four tables can be used. Each table requires 64 x 8 bit words. The tables occupy contiguous memory locations. The memory map of the quantization memory is shown in *Table 484.*

Table 484. JPGCqmem memory map

First Address	Last Address	Table
0	63	Table 0
64	127	Table 1
128	191	Table 2
192	255	Table 3

For decoding with header parsing, no quantization table programming is required, because the codec core extracts the dequantization coefficients from the JPEG encoded data, and write them to the JPGCQMem memory.

For encoding and decoding ECS data, quantization value can be simply loaded into the tables.

Note: The quantization coefficients must be specified in the table in zigzag order.

25.4.16 JPGChuffmin memory

Together with the HuffBase table and the HuffSymb table, this is one of the three Huffman tables required by the Codec Core when it acts as a decoder.

The HuffMin table can be up to 4 x 100 bit words. its memory map is shown in *Table 485.*

Table 485. JPGCHuffMin memory map

Address	Value
0	MIN AC 0 value
1	MIN DC 0 value
2	MIN AC 1 value
3	MIN DC 1 value

When decoding with header processing, this table is automatically programmed by the codec core, while in the case of ECS only decoding, the HuffMin table must be programmed before starting the codec core.

25.4.17 JPGC huffbase memory

Together with the HuffMin table and the HuffSymb table, this is one of the three Huffman tables required by the Codec Core when it acts as a decoder.

The HuffBase table can be up to 64 x 9 bit words. its memory map is shown in *Table 486.*

Table 486. JPGC huffbase memory map

First address	Last address	Table
0	15	BASE AC 0 value
16	31	BASE DC 0 value
32	47	BASE AC 1 value
48	63	BASE DC 1 value

When decoding with header processing, this table is automatically programmed by the codec core, while in the case of ECS only decoding, the HuffBase table must be programmed before starting the codec core.

25.4.18 JPGChuffsymb memory

Together with the HuffMin table and the HuffBase table, this is one of the three Huffman tables required by the codec core when it acts as a decoder.

The HuffSymb table can be up to 336 x 8 bit words. its memory map is shown in *Table 487.*

Table 487. JPGC huffsymb memory map

First address	Last address	Table
0	161	SYMB AC 0 value

Table 487. JPGC huffsymb memory map (continued)

First address	Last address	Table
162	173	SYMB DC 0 and 1values
174	335	SYMB AC 1 value

When decoding with header processing, this table is automatically programmed by the codec core, while in the case of ECS only decoding, the HuffSymb table must be programmed before starting the codec core.

25.4.19 JPGCDHTmem memory

Together with the HuffEnc table, this is one of the two Huffman tables required by the codec core when it acts as an encoder.

As specified in the ISO documentation, in the case of the baseline algorithm, up to two tables for encoding DC coefficients and two tables for encoding AC coefficients can be used. the memory map is shown in *Table 488.*

Table 488. JPGCDHTmem memory map

First address	Last address	Table
0	27	DC Huffman table 0
28	205	AC Huffman table 0
206	233	DC Huffman table 1
234	411	AC huffman table 1

The standard specifies that the Huffman table values be 8 bit words and in the following format:

DC tables and AC tables:

Li: number of huffman codes of length *i*: this specifies the number of huffman codes for each of the 16 possible lengths that the specification allows. This represents the first 16 bytes of each DC table and AC table address block in the JPGCDHTMem memory.

Vi: value associated with each Huffman code: this specifies the value associated with each Huffman code of length *i*. This $mt = L1 + L2 + \ldots + L16$ bytes following the 16 length values.

25.4.20 JPGChuffenc memory

Together with the HuffDHTMem table, this is one of the two Huffman tables required by the codec core when it acts as an encoder.

As specified in the ISO documentation, in the case of the baseline algorithm, up to two tables for encoding DC coefficients and two tables for encoding AC coefficients can be used. the memory map is shown in *Table 489.*

Table 489. JPGCHuffEnc memory map

First address	Last address	Table
0	175	AC Huffman table 0
176	351	AC Huffman table 1
352	367	DC Huffman table 0
368	383	DC Huffman table 1

Each AC table requires 176 x 12 bit words. Each DC table requires 16 x 12 bit words. All the AC and DC tables occupy contiguous locations in the JPGCHuffEnc memory.

Each Huffman code is stored as record containing the actual code HCODE (bits [7:0] of each 12 bit word) and its length HLEN (bits [11:8] of each 12 bit word).

HLEN are the 4 most significant bits of the huffman code. It is the number of bits in the Huffman code minus 1.

HCODE are the 8 least significant bits of the huffman code. If the huffman code is less than 8 bits long, the bits that are not used must be 0.

Although Huffman codes used in the JPEG algorithm can be up to 16 bits long, when the code is more than 8 bits long, the most significant bits are always 1. Therefore, it is unnecessary to specify more than 8 bits for any code, as the most significant bits are generated internally.

162 Huffman codes are required for the encoding the AC run-length codes and 12 for the DC coefficients.

The location of the Huffman codes for the 162 run-length codes in an AC table is shown in *Table 490.*

Table 490. Location of AC huffman codes in JPGCHuffEnc memory

Address	Value
0-9	Huffman code of run lengths 0/1 to 0/A
10-19	Huffman code of run lengths 1/1 to 1/A
20-29	Huffman code of run lengths 2/1 to 2/A
30-39	Huffman code of run lengths 3/1 to 3/A
40-49	Huffman code of run lengths 4/1 to 4/A
50-59	Huffman code of run lengths 5/1 to 5/A
60-69	Huffman code of run lengths 6/1 to 6/A
70-79	Huffman code of run lengths 7/1 to 7/A
80-89	Huffman code of run lengths 8/1 to 8/A
90-99	Huffman code of run lengths 9/1 to 9/A
100-109	Huffman code of run lengths A/1 to A/A
110-119	Huffman code of run lengths B/1 to B/A
120-129	Huffman code of run lengths C/1 to C/A
130-139	Huffman code of run lengths D/1 to D/A

Table 490. Location of AC huffman codes in JPGCHuffEnc memory (continued)

Address	Value
140-149	Huffman code of run lengths E/1 to E/A
150-159	Huffman code of run lengths F/1 to F/A
160	Huffman code of EOB
161	Huffman code of ZRL
162-167	$FFF
168-175	$FD0-$FD7

Locations 162-175 of each AC table contain information used internally by the Codec Core. The location of the huffman codes for the 12 codes in an DC table is shown in *Table 491.*

Table 491. Location of DC huffman codes in JPGCHuffEnc memory

Address	Value
0-11	Huffman code of DC codes 0 to A
12-15	Not used

26 LS_Fast IrDA controller

26.1 Overview

Within its low speed connectivity subsystem, the device provides a fast IrDA Controller modeled according to the standard of the infrared data association (IrDA).

It is a programmable infrared controller that acts as an interface between the on-chip AHB bus and infrared transceiver. This controller is able to perform the modulation and the demodulation of the infrared signals and the wrapping of the IrDA link access protocol (IrLAP) frames.

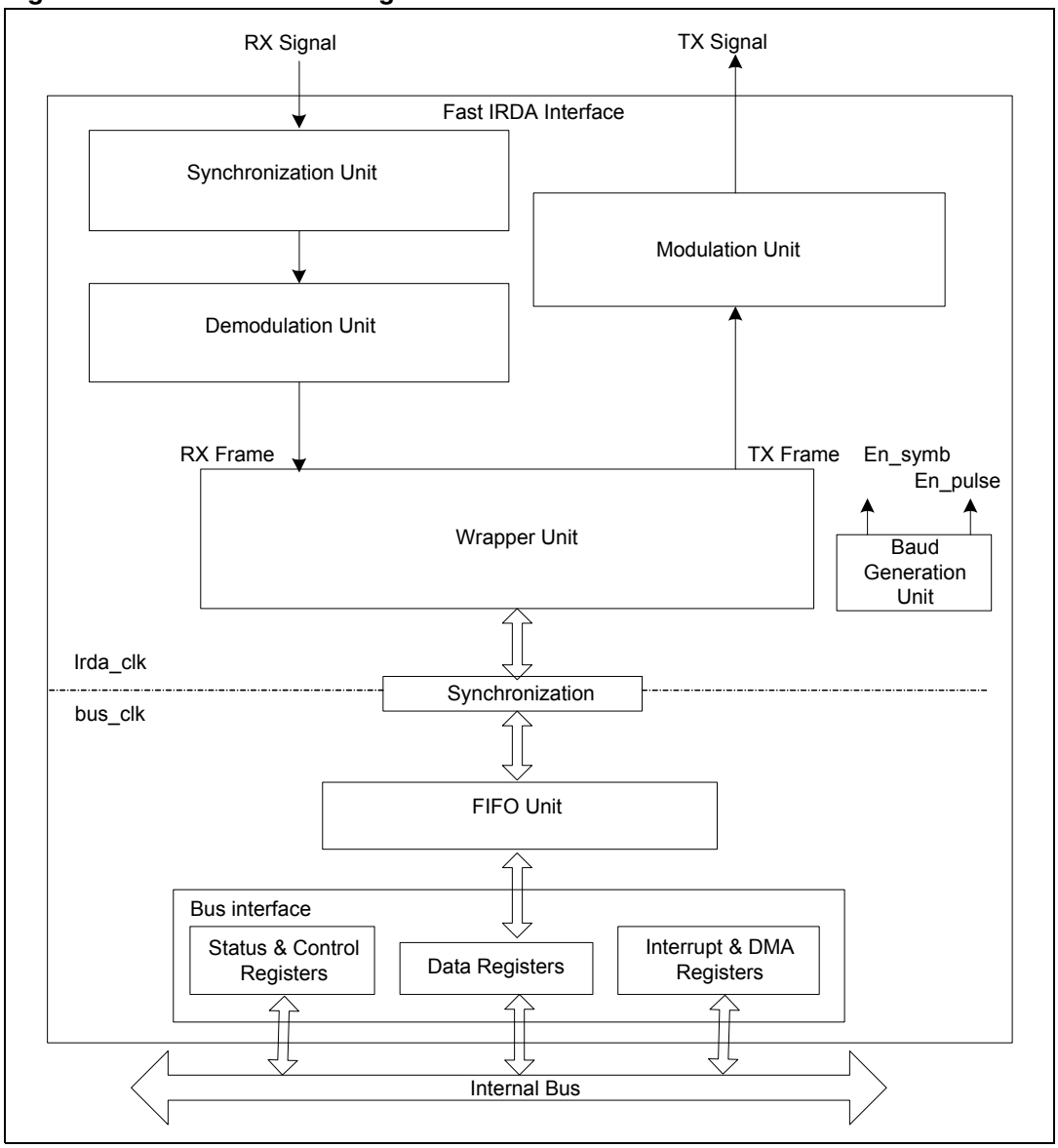
Main features of the fast IrDA (FIrDA) Controller are:

- supports different standards:
  - IrDA serial infrared physical layer specification (IrPHY), version 1.3
  - IrDA link access protocol (IrLAP), version 1.1
- supports different infrared modes and baud rates:
  - Serial infrared (SIR), with rates 9.6 Kbps, 19.2 Kbps, 38.4 Kbps, 57.6 Kbps and 115.2 Kbps
  - Medium infrared (MIR), with rates 576 Kbps and 1.152 Mbps
  - Fast infrared (FIR), with rate 4 Mbps.
- Provides a transceiver interface compliant to all IrDA transceivers with configurable polarity of TX and RX signals.
- Integrates half-duplex infrared frame transmission and reception.
- Integrates 16 bit CRC algorithm for SIR and MIR, and 32 bit CRC algorithm for FIR.
- Generates preamble, start and stop flag.
- Uses the RZI (return-to-zero inverted) modulation/demodulation scheme for SIR and MIR, and the 4PPM (4 pulse position modulation) modulation/demodulation scheme for FIR.
- Provides synchronization by means of a DPLL in FIR mode.
- Implements a payload data transfer controllable by either CPU or DMA controller.
- Presents two clock domains:
  - a dedicated clock (irda_clk signal) for an accurate signal generation (for example. 48 MHz),), (see for more details the Miscellaneous register PRPH_CLK_CFG[bit 6:5] in *Table 165: PRPH_CLK_CFG register bit assignments* and PLL programming sequence.
  - an independent and variable clock for the bus interface (for example. 13 MHz).

26.2 Block diagram

Figure 59 shows the dataflow block diagram of the FIrDA controller.

Figure 59. Dataflow block diagram of the FIrDA controller



26.3 Main functions description

26.3.1 Synchronization unit

The *synchronization unit* block allows to synchronize the RX signal of the off-chip IrDA transceiver. The RX signal is sampled by the rising edge of the `irda_clk` clock signal for synchronization.

If the *synchronization unit* detects an activity of the RX signal in the listening state, the FIrDA controller switches to the reception state and sets the RXS bit of the IrDA_STAT register (*Section 26.5.8*), then a signal detected interrupt (SD_INT, *Section 26.4*) is generated.

Besides, if the *synchronization unit* detects no activity of the RX signal for more than 10 ms when it is in the reception state, the FIrDA controller switches back to the listening state and

reset the bit RXS of the IrDA_STAT register. At last, a frame invalid interrupt (FI_INT, *Section 26.4*) is generated. This behaviour allows to handle the case of a frame abort.

Note: The used reception abort timer has to be programmed via the field RATV of the configuration register IrDA_CONF (Section 26.5.5).

26.3.2 Demodulation unit

The demodulation unit is active in the reception state only, and it is responsible for the demodulation of the synchronized active high RX signal from the synchronization unit in order to obtain the RX frame. The actual demodulation performed by this unit depends on the infrared mode (SIR, MIR or FIR).

Note: The POLRX bit in the IrDA_CONF register (Section 26.5.5) must be set according to the polarity of the RX signal.

SIR and MIR (RIZ demodulation)

At first, the demodulation unit extends the pulse of the synchronized active high RX signal in order to avoid jitter influences. Then the obtained signal is then sampled by the `en_symb` signal (from the baud rate generation unit, *Section 26.3.5: Baud rate generation unit on page 574*) to get the RX frame for wrapper unit.

The `en_symb` signal has a phase determinated by the first rising edge of the incoming RX signal and is checked (re-adjusted, if needed) every following rising edge.

FIR (4PPM demodulation)

The preamble field PA of the synchronized RX signal is used by the receiver to establish phase lock by means of a DPLL (digital phase locked loop). Then the incoming signal is sampled with the recovered enable signal `en_pulse`. To establish symbol synchronization the receiver, during PA, begins to search for the start flag STA. When it is received correctly the received starts to demodulate the RX frame and generates a frame detected interrupt (FD_INT, *Section 26.4*) until the stop flag STOP, which indicates the end of the frame, is recognized.

26.3.3 Wrapper unit

The wrapper unit is active in transmission and reception states only.

Reception state

The wrapper unit retrieves the IrLAP frame and the CRC bytes out of the RX frame from demodulation unit. The decoding mode depends on the used infrared mode, which is determined by the 2 bit field MODE of the parameter register IrDA_PARA (*Section 26.5.6*).

After decoding, the IrLAP and CRC bytes are shifted into the FIFO unit. If an error is detected either in the demodulation unit or in FIFO Unit, the decoding process is aborted.

Transmission state

The *wrapper unit* builds the TX frame out of IrLAP frame which should be transmitted. The TX frame is then sent (LSB first) to the modulation unit. The encoding mode depends on the used infrared mode, which is determined by the 2 bit field MODE of the parameter register IrDA_PARA (*Section 26.5.6*).

26.3.4 Modulation unit

The modulation unit is active in the transmission state only, and it is responsible for the modulation of the TX frames from the wrapper unit in order to generate the TX signal for the off-chip IrDA transceiver. The actual modulation performed by this unit depends on the infrared mode (SIR, MIR or FIR).

Note: The POLTX bit in the IrDA_CONF register (Section 26.5.5) determines the polarity of the TX signal.

The TX signal is generated by means of the `en_symb` and `en_pulse` signals from the baud rate generation unit (*Section 26.3.5: Baud rate generation unit on page 574*). If a frame is completely transmitted, a frame transmitted interrupt (FT_INT, *Section 26.4*) is generated and the FIrDA controller changes back to the listening state.

In case of FIR mode a 4PPM modulation is used. Additional preamble (PA), start flag (STA) and stop flag (STO) are added. With a bit rate of 4 Mbps the resulting data symbol duration is 500 ns and the chip duration is then 125 ns.

26.3.5 Baud rate generation unit

The baud rate generation unit creates the two enable signals which are used throughout the FIrDA controller, namely:

- en_symb, which determines the symbol rate at which the synchronized inverted RX signal from synchronization unit (*Section 26.3.1: Synchronization unit on page 572*) is sampled by the demodulation unit (*Section 26.3.2: Demodulation unit on page 573*) in the reception state of SIR and MIR modes.
- en_pulse, which creates the pulses of the TX signal during transmission.

The two signals are obtained from the same irda_clk clock signal by using cascaded clock dividers, so the resulting frequencies are:

$f_{en\_pulse} = f_{irda\_clk} \cdot K/L$

$f_{en\_symb} = f_{en\_pulse} / (N+1)$

where the values of K, L and N parameters are determined by software setting the 8 bit field INC, the 11 bit field DEC, and the 8 bit field N, respectively, of the divider register IrDA_DV (*Section 26.5.7*).

Note: The fractional divider causes jitter with a maximum of $1/(2 \cdot fi_{rda\_clk})$, that is 10.417 ns at SIR and MIR (being f_{irda_clk} = 48 MHz), which meets the IrPHY specification.

In case of SIR, for each SIR symbol one bit is transmitted, then the bit rate and the symbol rate are equal. It follows that the *baud rate generation unit* has to create the following symbol rates: fen_symb = 9.6 kHz, 19.2 kHz, 38.4 kHz, 57.6 kHz and 115.2 kHz. Besides, since a pulse duration of 1.736 µs is used in SIR transmission, the *baud rate generation unit* has to create a pulse rate of f_{en_pulse} = 576 kHz.

Like SIR, for each MIR symbol one bit is transmitted only, then the bit rate and the symbol rate are equal. The *baud rate generation unit* has to create the following symbol rates, f_{en_symb} = 576 kHz and 1.152 MHz. Moreover, since a pulse duration of a quarter of the symbol duration is used for MIR transmission, the *baud rate generation unit* has to create a pulse rate of fen_pulse = 4* f_{en_symb}.

At last, for each FIR symbol two bits are transmitted: the symbol rate is then one half of the bit rate, and the *baud rate generation unit* has to create a unique symbol rate of f_{en_symb} = 2 MHz (being a bit rate of 4 Mbps). Since the pulse duration is a quarter of the symbol

duration for FIR transmission (like MIR), the *baud rate generation unit* has to create a pulse rate of fen_pulse = 4* f_{en_symb}.

Table 492 provides a list of the settings of K, L and (N+1) parameters in case of SIR, MIR and FIR (and with f_{irda_clk} = 48 MHz).

Table 492. Settings of K,L and (N+1) parameters for SIR,MIR and FIR in baud rate generation unit

	Bit rate [Kbps]	f_{en_pulse} [kHz]	f_{en_symb} [kHz]	K	L	N+1
SIR	9.6	576	9.6	3	250	60
	19.2	576	19.2	3	250	30
	38.4	576	38.4	3	250	15
	57.6	576	57.6	3	250	10
	115.2	576	115.2	3	250	5
MIR	576	2304	576	6	125	4
	1152	4608	1152	12	125	4
FIR	4000	8000	2000	1	6	4

26.3.6 FIFO unit

The *FIFO unit* allows to control the data transfer between peripheral and system memory, and to buffer the reception and the transmission data.

In particular, data can be transmitted by writing to the transmission buffer register (IrDA_TXB, *Section 26.5.11*), which represent the head of the FIFO, and can be read out from the reception buffer register (IrDA_RXB, *Section 26.5.12*), which is the tail of the FIFO.

An 8-stage 32 bit shift register is used as buffer. This allows that data can be transferred at full speed using DMA burst transfers.

Note: Since IrDA supports only half-duplex communication, one buffer is used for both transmission and reception.

The FIrDA controller generates the following request signals to control the data transfer to and from memory.

Table 493. Request signals

Signal	Description
BREQ_INT	Burst request signal. Request a transfer of a programmed burst of words.
LBREQ_INT	Last burst request signal. Request a last burst transfer.
SREQ_INT	Single request signal. Request a transfer of a single word.
LSREQ_INT	Last single request signal. Request a last single transfer.

These signals can either be used as either interrupt requests (*Section 26.4*) or DMA requests, and they are reset on the occurrence of the request clear signal (`REQCLR`) which is either set by software via IrDA_ICR register (*Section 26.5.16*) or generated by a DMA

controller, respectively. The burst size is programmable by the field BS of IrDA_CONF register (*Section 26.5.5*).

Transmission state

In order to start to transmit data, the software writes the frame size to the 12 bit field TFS of transmission frame size register (IrDA_TFS, *Section 26.5.9*). Then, the FIrDA controller changes to the transmission state and the *FIFO unit* asserts burst requests (BREQ_INT) until the amount of data to be transferred is equal or less than BS.

If the remaining data is equal to BS, a last burst request is issued using LBREQ_INT, otherwise single requests are issued using SREQ_INT until the last data item is ready, when LSREQ_INT is used.

Note: The size of the frame to be transmitted is not necessarily a multiple of 4, so the last word could be filled up with dummy bytes. The hardware transmits only the valid bytes of the last word by means of the bit field TFS of IrDA_TFS register.

The data is buffered in an 8-stage 32 bit shift register before it is processed by the FIrDA controller. If a FIFO underflow occurs before all bytes of a frame has been shifted into the FIFO, a frame invalid interrupt (FI_INT, *Section 26.3.6*) is generated, the transmission is aborted and all pending bytes in the peripheral are discarded.

The next frame to be transmitted can be copied into the buffer only when all bytes of the current frame are completely transferred to the wrapper unit. The software can write the size of the next frame into TFS immediately after the last word of the current frame has been written to the IrDA_TXB register (*Section 26.5.11*).

Reception state

The received bytes of the frame are shifted from the wrapper unit to the FIFO where the data is buffered. The received bytes are counted by a 12 bit counter and that value can be read by software in the received frame size register (IrDA_RFS, *Section 26.5.10*). If this number is greater than the maximum number of received bytes (MNRB field in the IrDA_PARA register, *Section 26.5.6*), the currently received frame becomes invalid, a buffer overflow occurs and a frame invalid interrupt (FI_INT, *Section 26.4*) is generated.

The signal frame complete indicates that the whole data of the current received frame has been moved to the buffer. The bytes of this frame have to be moved out of the buffer by software before the next received frame will be shifted into the buffer, if not the next received frame will be completely discarded.

The buffered data is moved out at full speed bus using DMA burst transfers: the FIFO unit asserts burst requests (BREQ_INT) until the amount of data to be transferred is equal or less than BS. If the remaining data is equal to BS, a last burst request is issued using LBREQ_INT, otherwise single requests are issued using SREQ_INT until the last data item is ready, when a LSREQ_INT is used.

Note: 1 The size of the frame to be received is not necessarily a multiple of 4, so the upper bytes of the last word could be invalid. The SW has to check for invalid bytes of the last word by means of the bit field RFS of the IrDA_RFS register (Section 26.5.10).

2 Along with the IrLAP bytes, the CRC bytes are also transferred to the memory. The CRC bytes can be double-checked by the software for the purpose of testing.

The occurrence of a frame invalid interrupt (FI_INT, *Section 26.4*) due to any reason during the reception indicates that the received data has become invalid and then the buffer content is cleared without sending further requests.

When the received frame has been completely read out of the buffer, the FIrDA controller changes back to the listening stage.

26.4 Interrupt sources

Table 494 shows a summary of the interrupts of the FIrDA controller. A brief description of each interrupt follows after the table.

Table 494. FIrDA controller interrupt summary

Name	Description
FD_INT	Frame detected. This type of interrupt indicates that a frame has been detected during the Reception State (see *Section 26.3.3: Wrapper unit*).
FI_INT	Frame invalid. When it occurs in reception state, it means that the currently received frame is invalid. This can be due to a CRC error, the reception of an invalid flag (see *Section 26.3.3: Wrapper unit*), a frame abort (see *Section 26.3.1: Synchronization unit*), the reception of an invalid symbol (see *Section 26.3.2: Demodulation unit*), the reception of a too long frame or a FIFO overflow (see *Section 26.3.6*), or an abort by software. In the interrupt service routine the software should discard the bytes of the current frame, which have already been transferred to the memory. When it occurs in transmission state, it indicates that the current frame has not been sent completely. This can be due to either a FIFO underflow (see *Section 26.3.6*), or an abort by software. Note: FD_INT must have a lower priority than FI_INT.
SD_INT	Signal detected. This kind of interrupt indicates that a signal has been detected by the Synchronization Unit during the Listening State (*Section 26.3.1: Synchronization unit*).
FT_INT	Frame transmitted. This kind of interrupt occurs when a frame has been completely transmitted by the Modulation Unit (see *Section 26.3.4: Modulation unit*).
BREQ_INT	Burst request. This interrupt occurs when a transfer of a programmed burst number of words from/to the memory is requested. This request can either be used as interrupt or DMA requests (see *Section 26.3.6*).
LBREQ_INT	Last burst request. This interrupt indicates that a last burst transfer from/to the memory is requested. This request can either be used as interrupt or DMA requests (see *Section 26.3.6*).
SREQ_INT	Single request. This interrupt indicates that a transfer of a single word from/to the memory is requested. This request can either be used as interrupt or DMA requests (see *Section 26.3.6*).
LSREQ_INT	Last single request. This interrupt occurs when a last single transfer from/to the memory is requested. This request can either be used as interrupt or DMA requests (see *Section 26.3.6*).

26.5 Programming model

26.5.1 External pin connection

Table 495. External pin connection

Signal	Ball
IrDA_RXD	E3
IrDA_TXD	F3

26.5.2 Register map

The FIrDA controller can be fully configured by programming its 32 bit wide registers which can be accessed at the base address 0xD100_0000.

As depicted in *Figure 59*, FIrDA controller registers can be logically arranged in three main groups:

- Control and status registers (listed in *Table 496*), for IrDA configuration,
- Data registers (listed in *Table 497*), containing the data bytes,
- Interrupt and DMA registers (listed in *Table 498*), for managing interrupts and DMA requests.

Table 496. FIrDA controller control and status registers summary

Name	Offset	Type	Reset value	Description
IrDA_CON	0x10	RW	32'h0	IrDA control.
IrDA_CONF	0x14	RW	32'h00020EA6	IrDA configuration.
IrDA_PARA	0x18	RW	32'h00460000	IrDA parameter.
IrDA_DV	0x1C	RW	32'h0	IrDA divider.
IrDA_STAT	0x20	RO	32'h0	IrDA status.
IrDA_TFS	0x24	WO	32'h0	Transmission frame size.
IrDA_RFS	0x28	RO	32'h0	Reception frame size.

Table 497. FIrDA controller data registers summary

Name	Offset	Type	Reset value	Description
IrDA_TXB	0x2C	WO	32'h0	Transmission buffer.
IrDA_RXB	0x30	RO	32'h0	Reception buffer.

Table 498. FIrDA controller interrupt and DMA registers summary

Name	Offset	Type	Reset value	Description
IrDA_IMSC	0xE8	RW	32'h0	Interrupt mask control.
IrDA_RIS	0xEC	RO	32'h0	Raw interrupt status.
IrDA_MIS	0xF0	RO	32'h0	Masked interrupt status.

Table 498. FIrDA controller interrupt and DMA registers summary (continued)

Name	Offset	Type	Reset value	Description
IrDA_ICR	0xF4	WO	32'h0	Interrupt clear.
IrDA_ISR	0xF8	WO	32'h0	Interrupt set.
IrDA_DMA	0xFC	RW	32'h0	DMA control.

26.5.3 Register description

26.5.4 IrDA_CON register

The IrDA_CON (control) is a RW register which allows to control the FIrDA controller. The IrDA_CON bit assignments are given in *Table 499*.

Table 499. IrDA_CON register bit assignments

Bit	Name	Reset value	Description
[31:01]	Reserved	-	Read: undefined. Write: should be zero.
[00]	RUN	1'h0	Enable FIrDA controller. Enable the FIrDA controller according to the encoding: 1'b0 = FIrDA controller switches to the inactive state. 1'b1 = FIrDA controller switches to the listening state.

26.5.5 IrDA_CONF register

The IrDA_CONF (configuration) is a RW register which is able to configure the FIrDA controller. This register should only be modified when the FIrDA controller is disabled by clearing the bit RUN of the IrDA_CON register. The IrDA_CONF bit assignments are given in *Table 500*.

Table 500. IrDA_CONF register bit assignments

Bit	Name	Reset value	Description
[31:21]	Reserved	-	Read: undefined. Write: should be zero.
[20]	POLTX	1'h0	Polarity of TX pulses. This bit indicates the polarity of the TX pulses generated by the modulation unit (*Section 26.3.4: Modulation unit on page 574*), according to the encoding: – 1'b0 = Active high (default). – 1'b1 = Active low.
[19]	POLRX	1'h0	Polarity of RX pulses. This bit indicates the polarity of the RX pulses generated by the demodulation unit (*Section 26.3.2: Demodulation unit on page 573*), according to the encoding: – 1'b0 = Active low (default). – 1'b1 = Active high.

Table 500. IrDA_CONF register bit assignments (continued)

Bit	Name	Reset value	Description
[18:16]	BS	3'h2	Burst size. This 3 bit field indicates the value of the burst size, according to the encoding: – 3'b000 = 1 word – 3'b001 = 2 words – 3'b010 = 4 words (default) Any other value = Reserved Note: DMA controller does not support a burst size of 2 words.
[15:13]	Reserved	-	Read: undefined. Write: should be zero.
[12:00]	RATV	13'h0EA6	Reception abort timer value. This 13 bit field indicates the reception abort timer value, according to the encoding: **Value f_{irda_clk} [MHz] Time gap** 13'b0110000110101 4010 ms 13'b0111010100110 48 (default)10 ms 13'b1000100010111 5610 ms 13'b1111110111101 10410 ms Note: A frame with a single pair of characters with a time gap greater than 10 ms is considered as an invalid frame. The reception abort timer T_{abort} of the synchronization unit (*Section 26.3.1: Synchronization unit on page 572*) has to be programmed with RATV accordingly to the following equation: RATV = ($T_{abort} \cdot f_{irda\_clk}$) / 128.

26.5.6 IrDA_PARA register

The IrDA_PARA (parameter) is a RW register which states the transmission parameters. This register should only be modified when the FIrDA controller is disabled by clearing the bit RUN of the IrDA_CON register. The IrDA_PARA bit assignments are given in *Table 501*.

Table 501. IrDA_PARA register bit assignments

Bit	Name	Reset value	Description
[31:28]	Reserved	-	Read: undefined. Write: should be zero.
[27:16]	MNRB	12'h046	Maximum number of received bytes. This 12 bit field indicates the maximum number of received bytes, according to the encoding: – 12'b000001000110 = 70 bytes (default) – 12'b000010000110 = 134 bytes – 12'b000100000110 = 262 bytes – 12'b001000000110 = 518 bytes – 12'b010000000110 = 1030 bytes – 12'b100000000110 = 2054 bytes Note: In FIR mode, the effective maximum number of received bytes is data size + 6 bytes, including the information byte, the address and control byte, and the 4 CRC bytes. Note: This field should be programmed according to the negotiated data size (see IrLAP specification).
[15:08]	Reserved	-	Read: undefined. Write: should be zero.
[07:02]	ABF	6'h0	Number of additional beginning flags. This 6 bit field indicates the number of additional beginning flags, according to the encoding: – 6'b000000 = No additional beginning flags. – 6'b000001 = 1. … = … – 6'b110000 = 48. Any other value = Reserved.
[01:00]	MODE	2'h0	Infrared mode. This 2 bit field allows to select the used infrared mode, according to the encoding: – 2'b00 = SIR – 2'b01 = MIR – 2'b10 = FIR – 2'b11 = Reserved

26.5.7 IrDA_DV register

The IrDA_DV (Divider) is a RW register which allows to set the clock divider within the baud generation unit (*Section 26.3.5: Baud rate generation unit on page 574*) to get the `en_symb` and `en_pulse` signals. This register should only be modified when the FIrDA controller is disabled by clearing the bit RUN of the IrDA_CON register. The IrDA_DV bit assignments are given in *Table 502*.

Table 502. IrDA_DV register bit assignments

Bit	Name	Reset value	Description
[31:27]	Reserved	-	Read: undefined. Write: should be zero.
[26:16]	DEC	11'h0	Decrement value of fractional divider. This 11 bit field represents the decrement value of the fractional divider, following the formula DEC = L – K, where L and K values are listed in *Table 492*.
[15:08]	INC	8'h0	Increment value of fractional divider. This 8 bit field indicates the increment value of the fractional divider, following the formula INC = K, where K values are listed in *Table 492*. Note: It always has to be K < L. K = L is not allowed, apart from K = L = 0, resulting in `en_pulse` equal to `irda_clk`.
[07:00]	N	8'h0	Denominator of the integer divider. This 8 bit field allows to set the denominator of the integer divider, which is (N+1). The N value can range from 0 (8'h00) to 255 (8'hFF).

26.5.8 IrDA_STAT register

The IrDA_STAT (status) is a RO register which reflects the status of the FIrDA controller. The IrDA_STAT bit assignments are given in *Table 503*.

Table 503. IrDA_STAT register bit assignments

Bit	Name	Reset value	Description
[31:02]	Reserved	-	Read: undefined.
[01]	TXS	1'h0	If set, the FIrDA controller is in the transmission state.
[00]	RXS	1'h0	If set, the FIrDA controller is in the reception state.

26.5.9 IrDA_TFS register

The IrDA_TFS (transmission frame size) is a WO register which indicates the size of the frame to be transmitted by the FIrDA controller. The IrDA_TFS bit assignments are given in *Table 504*.

Table 504. IrDA_TFS register bit assignments

Bit	Name	Reset value	Description
[31:12]	Reserved	-	Write: should be zero.
[11:00]	TFS	12'h0	Transmission frame size. This 12 bit field indicates the size of the transmitted frame, according to the encoding: – 12'b000000000000 = Reset value. – 12'b000000000001 = 2 data bytes. – 12'b000000000010 = 3 data bytes. … = … – 12'b100000000001 = 2050 data bytes. Any other value = Reserved. Note: The number of transmitted bytes is data size + the information byte + the address and control byte.

26.5.10 IrDA_RFS register

The IrDA_RFS (reception frame size) is a RO register which states the size of the received frame. The IrDA_RFS bit assignments are given in *Table 505*.

Table 505. IrDA_RFS register bit assignments

Bit	Name	Reset value	Description
[31:12]	Reserved	-	Read: undefined.
[11:00]	RFS	12'h0	Reception frame size. This 12 bit field indicates the size of the received frame, according to the encoding: – 12'b000000000000 = Reset value. – 12'b000000000100 = 4 data bytes. … = … – 12'b100000000110 = 2054 data bytes. Any other value = Reserved Note: In SIR and MIR modes, the number of received bytes is data size + 4 bytes, including the information byte, the address and control byte, and the 2 CRC bytes. Note: In FIR mode, the number of received bytes is data size + 6 bytes, including the information byte, the address and control byte, and the 4 CRC bytes.

26.5.11 IrDA_TXB register

The IrDA_TXB (transmission buffer) is a WO register which contains the transmit data bytes in transmission mode. The IrDA_TXB bit assignments are given in *Table 506*.

Table 506. IrDA_TXB register bit assignments

Bit	Name	Reset value	Description
[31:00]	TXD	32'h0	Transmission data.

Note: Between two write accesses there must be a pause of one clock cycle.

26.5.12 IrDA_RXB register

The IrDA_RXB (reception buffer) is a RO register which contains the receive data bytes in reception mode. The IrDA_RXB bit assignments are given in *Table 507*.

Table 507. IrDA_RXB register bit assignments

Bit	Name	Reset value	Description
[31:00]	RXD	32'h0	Reception data.

26.5.13 IrDA_IMSC register

The IrDA_IMSC (interrupt mask control) is a RW register which allows to enable the FIrDA controller interrupts (*Section 26.4*). The IrDA_IMSC bit assignments are given in *Table 508*.

Reading this register gives the current value of the interrupts mask (1'b0 means interrupt disabled, 1'b1 interrupt enabled).

Writing a 1'b1 to a particular bit ([0:7]) sets the corresponding mask of that interrupt, whereas writing a 1'b0 clears the relevant interrupt.

Table 508. IrDA_IMSC register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined. Write: should be zero.
[07]	FD	1'h0	Frame detected interrupt mask.
[06]	FI	1'h0	Frame invalid interrupt mask.
[05]	SD	1'h0	Signal detected interrupt mask.
[04]	FT	1'h0	Frame transmitted interrupt mask.
[03]	BREQ	1'h0	BREQ interrupt mask.
[02]	LBREQ	1'h0	LBREQ interrupt mask.
[01]	SREQ	1'h0	SREQ interrupt mask.
[00]	LSREQ	1'h0	LSREQ interrupt mask.

26.5.14 IrDA_RIS register

The IrDA_RIS (Raw Interrupt Status) is a RO register which reflects the current raw status value of the corresponding interrupt (before masking by IrDA_IMSC). The IrDA_RIS bit assignments are given in *Table 509*.

Table 509. IrDA_RIS register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[07]	FD	1’h0	Frame detected raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[06]	FI	1’h0	Frame invalid raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[05]	SD	1’h0	Signal detected raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[04]	FT	1’h0	Frame transmitted raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[03]	BREQ	1’h0	BREQ raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[02]	LBREQ	1’h0	LBREQ raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[01]	SREQ	1’h0	SREQ raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[00]	LSREQ	1’h0	LSREQ raw interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.

26.5.15 IrDA_MIS register

The IrDA_MIS (masked interrupt status) is a RO register which gives the current masked status value of the corresponding interrupt (after masking by IrDA_IMSC). The IrDA_MIS bit assignments are given in *Table 510*.

Table 510. IrDA_MIS register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined. Write: should be zero. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[07]	FD	1’h0	Frame detected masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[06]	FI	1’h0	Frame invalid masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[05]	SD	1’h0	Signal detected masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[04]	FT	1’h0	Frame transmitted masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[03]	BREQ	1’h0	BREQ masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[02]	LBREQ	1’h0	LBREQ masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[01]	SREQ	1’h0	SREQ masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.
[00]	LSREQ	1’h0	LSREQ masked interrupt status. 1‘b0 = No interrupt. 1‘b1 = Interrupt pending.

26.5.16 IrDA_ICR register

The IrDA_ICR (interrupt clear) is a WO register which allows to clear interrupts. The IrDA_ICR bit assignments are given in *Table 511*.

Writing a 1‘b1 to a bit clears the corresponding interrupt. Writing 1‘b0 has no effect.

Table 511. IrDA_ICR register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Write: should be zero.
[07]	FD	1’h0	Frame detected interrupt clear.
[06]	FI	1’h0	Frame invalid interrupt clear.
[05]	SD	1’h0	Signal detected interrupt clear.

Table 511. IrDA_ICR register bit assignments (continued)

Bit	Name	Reset value	Description
[04]	FT	1'h0	Frame transmitted interrupt clear.
[03]	BREQ	1'h0	BREQ interrupt clear.
[02]	LBREQ	1'h0	LBREQ interrupt clear.
[01]	SREQ	1'h0	SREQ interrupt clear.
[00]	LSREQ	1'h0	LSREQ interrupt clear.

26.5.17 IrDA_ISR register

The IrDA_ISR (interrupt set) is a WO register which allows to set interrupt. The OR of the eight less significant bits is the interrupt line (IRQ 17) that goes to the VIC module (see the *Section 8.4: Interrupt connection table*). The IrDA_ISR bit assignments are given in *Table 512*.

Writing a 1'b1 to a bit sets the corresponding interrupt. Writing 1'b0 has no effect.

Table 512. IrDA_ISR register bit assignments

Bit	Name	Reset value	Description
[31:08]	Reserved	-	Read: undefined. Write: should be zero.
[07]	FD	1'h0	Frame detected interrupt set.
[06]	FI	1'h0	Frame invalid interrupt set.
[05]	SD	1'h0	Signal detected interrupt set.
[04]	FT	1'h0	Frame transmitted interrupt set.
[03]	BREQ	1'h0	BREQ interrupt set.
[02]	LBREQ	1'h0	LBREQ interrupt set.
[01]	SREQ	1'h0	SREQ interrupt set.
[00]	LSREQ	1'h0	LSREQ interrupt set.

26.5.18 IrDA_DMA register

The IrDA_DMA is a RW register which manages the DMA requests. The IrDA_DMA bit assignments are given in *Table 513*.

Reading this register gives the current status of the mask on the relevant DMA request.

Writing a 1'b1 to a particular bit ([0:3]) enables the corresponding DMA request, whereas writing a 1'b0 clears a pending request and disables further requests.

Table 513. IrDA_DMA register bit assignments

Bit	Name	Reset value	Description
[31:04]	Reserved	-	Read: undefined. Write: should be zero.
[03]	BREQEN	1'h0	Burst request DMA enable.
[02]	LBREQEN	1'h0	Last burst request DMA enable.

Table 513. IrDA_DMA register bit assignments (continued)

Bit	Name	Reset value	Description
[01]	SREQEN	1'h0	Single request DMA enable.
[00]	LSREQEN	1'h0	Last single request DMA enable.

27 LS_Universal asynchronous receiver/transmitter (UART)

27.1 Overview

SPEAr300 has single UART:

- UART has a maximum baud rate of 3 Mbps and minimum baud rate of 45 bps. It supports modem control and hardware flow control signals.

Each UART is intended to perform:

- Serial-to-parallel conversion on data received from a peripheral device;
- Parallel-to-serial conversion on data transmitted to the peripheral device.

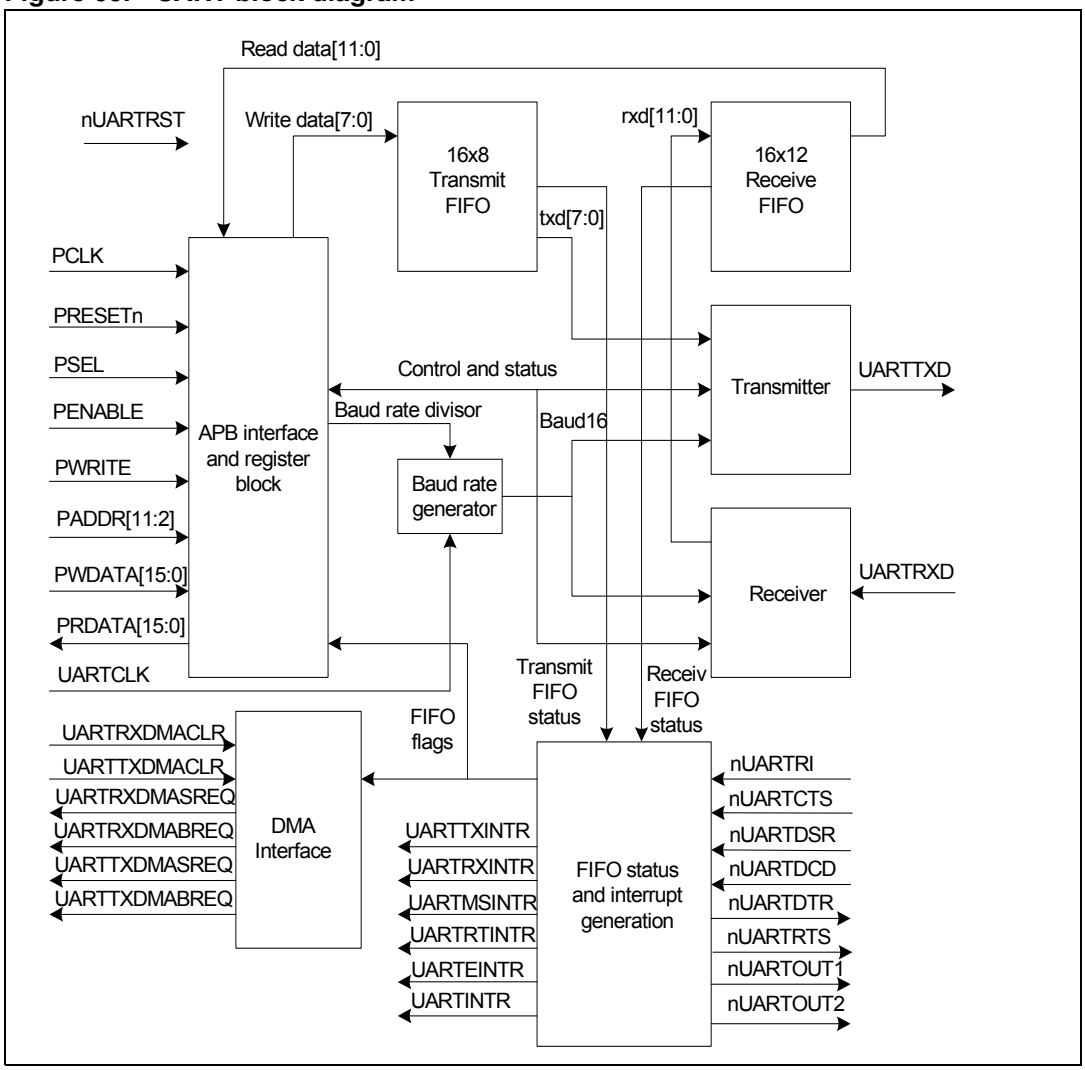
Main features of each UART are listed below:

- Separate 16 x 8 (16 location deep x 8 bit wide) transmit and 16 x 12 receive FIFOs to reduce CPU interrupts;
- Provides programmable FIFO disabling for 1-byte depth;
- Programmable baud rate generator that enables division of the reference clock by (1 x 16) to (65535 x16) and generates an internal x16 clock. The divisor can be a fractional number.
- Provides standard asynchronous communication bits (start, stop and parity) which are added prior transmission and removed on reception;
- Supplies independent masking of transmit or receive FIFO, receive timeout, and error condition interrupts. UART also supports modem status.
- Supports DMA;
- Detects false start bit;
- Generates and detects line break;
- UART supports the modem control functions CTS,DCD,DSR,RTS, DTS and RI.
- UART provides programmable hardware flow control.
- Provides fully-programmable serial interface with the subsequent characteristics:
  - data can be 5, 6, 7 or 8 bits;
  - even, odd, stick or no-parity bit generation and detection;
  - 1 or 2 stop bit generation;
  - UART baud rate generation dc up to UARTCLK_max_freq/16 (*Table 165: PRPH_CLK_CFG register bit assignments* (Bit 04) in Miscellaneous chapter).
  - UART1 - UART5 baud rate generation up to 5 Mbps
- Provides some programmable parameters, such as:
  - communication baud rate, integer and fractional parts;
  - number of data bits;
  - parity mode;
  - FIFO enable (16 deep) or disable (1 deep);
  - FIFO trigger levels selectable between 1/8, 1/4, 1/2, 3/4 and 7/8;

27.2 Functional description

27.2.1 Block diagram

Figure 60. UART block diagram



27.2.2 Main functions description

APB Interface

The *APB Interface* block generates read and write decodes for accesses to control and status registers (CSRs) as well as to transmit/receive FIFO memories.

Register Block

The *Register Block* allows to store data written, or to be read across the APB Interface.

Baud Rate Generator

The Baud Rate Generator contains free-running counters that generate the internal x16 clocks, and **Baud16** signal.

Baud16 provides timing information for UART transmit and receive control. It consists of a stream of pulses with a width of one **UARTCLK** clock period and a frequency of 16 times the baud rate.

Transmit FIFO

The Transmit FIFO consists of an 8 bit wide, 16 location deep, and FIFO memory buffer. CPU data written across the APB interface is stored in this FIFO until read out by the Transmit LogicThe Receive FIFO block can be disabled to act like a one-byte holding register.).

Note: The Transmit FIFO block can be disabled to act like a one-byte holding register.

Receive FIFO

The Receive FIFO consists of a 12 bit wide, 16 location deep, and FIFO memory buffer. Received data and corresponding error bits are stored in the *Receive FIFO* by the Receive Logic (section Receive logic) until read our by the CPU across the APB interface. This block can be disabled to act like a one-byte holding register.

Note: The Receive FIFO block can be disabled to act like a one-byte holding register.

Transmit Logic

The Transmit Logic performs **parallel-to-serial conversion** on the data read from the Transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit followed by data bits, with the LSB first and ended by parity bit and stop bit according to the programmed configuration in control registers (section Programming model).

Receive logic

The Receive Logic performs **serial-to-parallel conversion** on the received serial bit stream after a valid pulse has been detected. The *Receive Logic* also performs detection of overrun, parity, frame error checking and line break, and their status accompanies the data that is written to the Receive FIFO.

Interrupt generation logic

UART generates individual maskable active HIGH interrupts. A combined interrupt output is also generated as an OR function of the individual interrupt requests.

- A **single combined interrupt** can be used in a system interrupt controller that provides another level of masking on a per-peripheral basis. This enables to use modular device drivers that always know where to find the interrupt source control register bits.

DMA interface

The UART provides a *DMA Interface* to connect to a DMA controller. The DMA operation of the UART is controlled through the UART DMA control register.

The burst transfer and single transfer request signals are not mutually exclusive, so they can both be asserted at the same time. When the UART is in the FIFO disabled mode (where

both FIFO's act like a one-byte holding register), only the DMA single transfer mode can operate, since only one character can be transferred to or from the FIFO at any time.

Synchronization registers and logic

Since the UART supports both asynchronous and synchronous operation of the clocks **PCLK** and **UARTCLK**, *Synchronization Registers and Handshaking Logic* have been implemented and are active at all times. Synchronization of control signal is performed on both directions of data flow.

27.2.3 Interrupt sources

Table 514. shows a summary of the 11 maskable interrupts generated within the UART. These interrupts are combined to form four individual interrupt outputs and one which is the logical OR of the individual outputs.

The interrupts from UART1 to UART5 are further combined to generate a single interrupt at the RAS interface. The individual interrupt source can be determined from RAS interrupt status register.

Any individual interrupt can be enabled or disabled by changing the corresponding mask bit in the UARTIMSC register The status of the individual interrupt sources can be read either from the UARTRIS register for raw status or from the UARTMIS register for the masked status.

Table 514. UART interrupt summary together with combined outputs

<table>
<tr><th>Name</th><th>Source</th><th colspan="2">Combined Outputs</th></tr>
<tr><td>: UARTRXINTR</td><td>Receive FIFO.</td><td>UARTRXINTR</td><td rowspan="11">UARTINTR</td></tr>
<tr><td>: UARTTXINTR</td><td>Transmit FIFO.</td><td>UARTTXINTR</td></tr>
<tr><td>: UARTRTINTR</td><td>Receive time-out in Receive FIFO.</td><td>UARTRTINTR</td></tr>
<tr><td>: UARTMSINTR</td><td rowspan="4">NA</td><td rowspan="4">NA</td></tr>
<tr><td>UARTCTSINTR</td></tr>
<tr><td>UARTDCDINTR</td></tr>
<tr><td>UARTDCSRINTR</td></tr>
<tr><td>UARTOEINTR</td><td>Overrun error.</td><td rowspan="4">UARTEINTR</td></tr>
<tr><td>UARTBEINTR</td><td>Break error (in reception)</td></tr>
<tr><td>UARTPEINTR</td><td>Parity error in the received character.</td></tr>
<tr><td>UARTFEINTR</td><td>Framing error in the received character.</td></tr>
</table>

UARTRXINTR

- This interrupt is asserted when one of the following events occurs:
- If the FIFOs are enabled and the Receive FIFO reaches the programmed trigger level.
- The interrupt is then cleared by reading data from the Receive FIFO until it becomes less than the trigger level, or by clearing the interrupt;
- If the FIFOs are disabled and data is received thereby filling the location. The interrupt is then cleared by performing a single read of the Receive FIFO, or by clearing the interrupt (writing a 1‘b1 to the corresponding bit of the UARTICR register).

UARTTXINTR

- This interrupt is asserted when one of the following events occurs:
- If the FIFOs are enabled (FEN bit set to 1‘b1 in UARTLCR_H register) and the Transmit FIFO reaches the programmed trigger level (TXIFLSEL in UARTIFLS register). The interrupt is then cleared by writing data to the Transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt (writing a 1‘b1 to the corresponding bit of the UARTICR register);
- If the FIFOs are disabled and there is no data in the transmitter single location. The interrupt is then cleared by performing a single write to the Transmit FIFO, or by clearing the interrupt (writing a 1‘b1 to the corresponding bit of the UARTICR register).

UARTRTINTR

- This interrupt is asserted when the Receive FIFO is not empty, and no further data is received over a 32 bit period. The interrupt is then cleared either when the Receive FIFO becomes empty through reading all the data (or by reading the holding register), or by clearing the interrupt (writing a 1‘b1 to the corresponding bit of the UARTICR register).

UARTMSINTR

- This interrupt feature is available in UART only. It represents the modem status interrupt, that is a combined interrupt of the four individual modem status lines (`nUARTRI`, `nUARTCTS`, `nUARTDCS` and `nUARTDSR`). This interrupt is then asserted if any of the modem status lines change.

UARTEINTR

- This error interrupt is triggered when there is an error in the reception of the data. The interrupt can be caused by a number of different error conditions, such as overrun, break, parity and framing.

UARTINTR

- It is the OR logical function of all the individual masked interrupt sources. That is, this interrupt is asserted if any of the individual interrupts are asserted and enabled.

27.3 Programming model

27.3.1 Register map

The UART can be fully configured by programming its registers which can be accessed at the base address shown in *Table 515: UART base address*.

Table 515. UART base address

UART	Base Address
UART	0x D000.0000

UART registers can be logically arranged in five main groups:

- **Data register**, (listed in *Table 516*), contains data to be transmitted or received.
- **Error Status/Clear**, (listed in *Table 517*), contains UART receiver error status and clearing UART receive error.
- **control and status registers**, CSRs (listed in *Table 518*), for UART configuration and control.
- **interrupts and DMA registers** (listed in *Table 519*), for interrupts generation and DMA control.
- **identification registers** (listed in *Table 520*), namely eight 8 bit RO registers reporting UART-specific information.

Note: In addition to reserved locations within the CSRs address space (Table 518), offset addresses from 0x080 to 0xFDC are reserved for test purposes as well as for future extensions. All these locations must not be used during normal operation.

Note: UART must be disabled before any of the CSrs is programmed. When UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Table 516. UART data registers summary

Name	Offset	Width(bit)	Type	Reset value	Description
UARTDR	0x000	16	RW	16‘h0	UART data.

Table 517. UART error status/clear registers summary

Name	Offset	Width(bit)	Type	Reset value	Description
UARTRSR/UARTECR	0x004	8	RW	8‘h0	Receive status/error clear.

Table 518. UART control and status register summary

Name	Offset	Width(bit)	Type	Reset value	Description
-	0x008 to 0x014	-	-	-	Reserved.
UARTFR	0x018	16	RO	16’h0090	UART flag.
-	0x01C	-	-	-	Reserved.
UARTILPR	0x020	8	RW	8’h0	NA
UARTIBRD	0x024	16	RW	16‘h0	Integer baud rate.

Table 518. UART control and status register summary (continued)

Name	Offset	Width(bit)	Type	Reset value	Description
UARTFBRD	0x028	6	RW	6'h0	Fractional baud rate.
UARTLCR_H	0x02C	16	RW	16'h0	Line control.
UARTCR	0x030	16	RW	16'h0300	UART control.

Note: UARTLCR_H, UARTIBRD and UARTFBRD form a single 30 bit wide register named UARTLCR, which is updated on a single write strobe generated by a UARTLCR_H write. So, in order to internally update the contents of the UARTIBRD or UARTFBRD registers, a write to UARTLCR_H must always be performed at the end.

UART must be disabled (UARTEN(bit 1) of UARTCR register reset) before modifying any of the above control registers.

Table 519. UART interrupts and DMA registers summary

Name	Offset	Width (bit)	Type	Reset Value	Description
UARTIFLS	0x034	16	RW	16'h0012	Interrupt FIFO level select.
UARTIMSC	0x038	16	RW	16'h0	Interrupt Mask Select/clear.
UARTRIS	0x03C	16	RO	16'h0	Raw Interrupt Status.
UARTMIS	0x040	16	RO	16'h0	Masked Interrupt Status.
UARTICR	0x044	16	WO	-	Interrupt Clear.
UARTDMACR	0x048	16	RW	16'h0	DMA control.
-	0x04C to 0x07C	-	-	-	Reserved.

Table 520. UART identification register summary

Name	Offset	Width (bit)	Type	Reset value	Description
UARTPeriphID0	0xFE0	8	RO	8'h11	Peripheral identification.
UARTPeriphID1	0xFE4	8	RO	8'h10	
UARTPeriphID2	0xFE8	8	RO	8'h24	
UARTPeriphID3	0xFEC	8	RO	8'h00	
UARTPCellID0	0xFF0	8	RO	8'h0D	Prime cell identification.
UARTPCellID1	0xFF4	8	RO	8'hF0	
UARTPCellID2	0xFF8	8	RO	8'h05	
UARTPCellID3	0xFFC	8	RO	8'hB1	

27.4 Register description

27.4.1 UARTDR register

The UARTDR (UART Data) is a 16 bit RW register which contains data.

For words to be transmitted, if FIFOs are enabled, data written to this location is pushed onto transmit FIFO. If FIFOs are not enabled, data is stored in the transmitter holding

register (bottom word of the transmit FIFO). The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the parity bit (if enabled) and a stop bit. The resultant word is then transmitted.

For received words, if FIFOs are enabled, the data byte and the 4 bit status (break, frame, parity and overrun) is pushed into the 12 bit receive FIFO. If FIFOs are not enabled, data byte and status are stored in the receiving holding register (bottom word of the receive FIFO).

The UARTDR bit assignments are given in *Table 521*.

Table 521. UART data register summary

Name	Offset	Width(bit)	Type	Reset value	Description
UARTDR	0x000	16	RW	16'h0	UART data.

27.4.2 UARTRSR/UARTECR register

The UARTRSR/UARTECR (receive status/error clear) is a unique 8 bit RW register which allows to manage the function of both receive status and error clear register.

UARTRSR is intended for reading only to give the status information for break, framing and priority corresponds to the data character read from UARTDR (*Section 27.4.1*) prior to reading UARTRSR. The status information for overrun is set immediately when an overrun condition occurs. The UARTRSR bit assignments are given in *Table 522*.

In contrast, a write to UARTECR clears the framing, parity, break and overrun errors. The data value is not important. The UARTECR bit assignments are given in *Table 523*.

Table 522. UARTRSR register bit assignments

Bit	Name	Reset value	Description	
[07:04]	Reserved	-	Read: undefined. Write: should be zero.	
[03]	OE	1'h0	Overrun error.	See UARTDR register (*Section 27.4.1*).
[02]	BE	1'h0	Break error.	
[01]	PE	1'h0	Parity error.	
[00]	FE	1'h0	Framing error.	

Table 523. UARTECR register bit assignments

Bit	Name	Reset value	Description
[07:00]	-	8'h0	Clear errors.

Note: The received data character must be read first from UARTDR before reading the error status associated with the data character from UARTRSR. This read sequence cannot be reversed because UARTRSR is updated only when a read occurs from UARTDR. However, the status informations can also be obtained by reading the UARTDR register.

27.4.3 UARTFR register

The UARTFR (flag) is a RO register which indicates the flag status. The UARTFR bit assignments are given in *Table 524*.

Table 524. UARTFR register bit assignments

Bit	Name	Reset value	Description
[15:09]	Reserved	-	Read: as zero.
[08]	RI	1'h0	Ring indicator. This bit is set when the modem status input is 1'b0 (*Section 27.5*). Specifically, it is the complement of the UART data carrier detect nUARTRI modem status input.
[07]	TXFE	1'h1	Transmit FIFO empty. This bit depends on the state of the FEN bit in the UARTLCR_H register (*Section 27.4.6*). If FIFOs are disabled (FEN set to 1'b0), the TXFE bit is set when the transmit holding register is empty. if FIFOs are enabled (FEN set to 1'b1), it is set when the Transmit FIFO is empty. This bit does not indicate if there is data in the transmit shift register.
[06]	RXFF	1'h0	Receive FIFO full. This bit depends on the state of the FEN bit in the UARTLCR_H register. If FIFOs are disabled, RXFF is set when the receive holding register is full, whereas (FIFOs enabled) it is set when the receive FIFO is full.
[05]	TXFF	1'h0	Transmit FIFO full. This bit depends on the state of the FEN bit in the UARTLCR_H register. If FIFOs are disabled, TXFF is set when the transmit holding register is full, whereas (FIFOs enabled) it is set when the transmit FIFO is full.
[04]	RXFE	1'h1	Receive FIFO empty. This bit depends on the state of the FEN bit in the UARTLCR_H register. If FIFOs are disabled, RXFE is set when the receive holding register is empty, whereas (FIFOs enabled) it is set when the receive FIFO is empty.
[03]	BUSY	1'h0	UART busy. If this bit is set to 1'b1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether the UART is enabled or not).
[02]	DCD	1'h0	Data carrier detect. This bit is set to 1'b1 when the modem status input is 1'b0 (*Section 27.5*). Specifically, it is the complement of the UART data carrier detect nUARTDCD modem status input.

Table 524. UARTFR register bit assignments (continued)

Bit	Name	Reset value	Description
[01]	DSR	1'h0	Data set ready. This bit is set to 1'b1 when the modem status input is 1'b0 (*Section 27.5*). Specifically, it is the complement of the UART data carrier detect nUARTDSR modem status input.
[00]	CTS	1'h0	Clear to send. This bit is set to 1'b1 when the modem status input is 1'b0 (*Section 27.5*). Specifically, it is the complement of the UART clear to send nUARTCTS modem status input.

27.4.4 UARTIBRD register

The UARTIBRD (integer baud rate) is a 16 bit RW register which indicates the integer part of the baud rate divisor value. The UARTIBRD bit assignments are given in *Table 525*.

Table 525. UARTIBRD register bit assignments

Bit	Name	Reset value	Description
[15:00]	BAUD DIVINT	16'h0	Integer baud rate divisor.

27.4.5 UARTFBRD register

The UARTFBRD (fractional baud rate) is a 6 bit RW register which indicates the fractional part of the baud rate divisor value. The UARTFBRD bit assignments are given in *Table 526*.

Table 526. UARTFBRD register bit assignments

Bit	Name	Reset value	Description
[05:00]	BAUD DIVFRAC	6'h0	Fractional baud rate divisor.

The baud rate divisor is calculates as follows:

$BAUDDIV = f_{UARTCLK} / (16 \cdot \text{baud rate})$

where $f_{UARTCLK}$ is the UART reference clock frequency. The BAUDDIV is comprised of the integer value BAUD DIVINT and the fractional value BAUD DIVFRAC.

Note: The contents of UARTIBRD and UARTFBRD registers are not update until transmission or reception of the current character is complete.

Note: The minimum divide ratio is 1 and the maximum is 65535 (that is, 216-1). When UARTIBRD = 65535 (16'hFFFF), UARTFBRD must not be greater than zero.

Some typical bit rates and their corresponding integer divisors (BAUD DIVINT in UARTIBRD) are given in *Table 527*.

Table 527. Typical baud rate and divisors

Programmed integer divisor (UARTIBRD)	Bit rate [Bps]	Error
16‘h000D	230400	0.16%
16‘h001A	115200	0.16%
16‘h0027	76800	0.16%
16‘h0034	57600	0.16%
16‘h004E	38400	0.16%
16‘h009C	19200	0.16%
16‘h00D0	14400	0.16%
16‘h0138	9600	0.16%
16’h01A1	7200	-0.08%
16’h0271	4800	0%
16‘h04E2	2400	0%
16’h09C4	1200	0%
16’h1388	600	0%
16’h2710	300	0%
16’h3A98	200	0%
16‘h4E20	150	0%
16‘h6A88	110	0%

Note: For UART2-5, due to the faster 83 MHz UARTCLK, other divider values must be used to generate the selected baudrate

27.4.6 UARTLCR_H register

The UARTLCR_H (line control) is a 16 bit RW register which accesses bit 29 to 22 of the UART bit rate and line control register UARTLCR. The UARTLCR_H bit assignments are given in *Table 528*.

Table 528. UARTLCR_H register bit assignments

Bit	Name	Reset value	Description
[15:08]	Reserved	-	Read: as zero. Write: should be zero.
[07]	SPS	1’h0	Stick parity select. When bits 1 (PEN), 2 (EPS) and 7 (SPS, this one) of this register are set, the parity bit is transmitted and checked as 1‘b0. When bits 1 and 7 of this register are set, and bit 2 is cleared, the parity bit is transmitted and checked as 1‘b1. When bit SPS is cleared, stick parity is disabled.

Table 528. UARTLCR_H register bit assignments (continued)

Bit	Name	Reset value	Description
[06:05]	WLEN	2'h0	Word length. This 2 bit field indicates the number of data bits transmitted or received in a frame, according to encoding: – 2'b00 = 5 – 2'b01 = 6 – 2'b10 = 7 – 2'b11 = 8
[04]	FEN	1'h0	Enable FIFOs. Setting this bit, transmit and receive FIFO buffers are enabled. In contrast (FEN cleared), the FIFOs are disabled becoming 1-byte-deep holding registers.
[03]	STP2	1'h0	Two stop bit select. Setting this bit, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
[02]	EPS	1'h0	Even parity select. This bit allows to select either an even or an odd parity generation and checking during transmission and reception, which checks for an even or an odd number of 1s in data and parity bits, according to encoding: 1'b0 = Odd 1'b1 = Even Note: This bit has no effect when parity is disabled by clearing Parity Enable bit (PEN in this register).
[01]	PEN	1'h0	Parity enable. Setting this bit, parity checking and generation is enabled, otherwise (PEN set to 1'b0) parity is disabled and no parity bit is added to the data frame.
[00]	BRK	1'h0	Send break. Setting this bit, a low-level is continually output on the UARTTXD output after completing transmission of the current character. For proper execution of the break command, the software must set this bit for at least two complete frames.

Note: UARTLCR_H, UARTIBRD and UARTFBRD form a single 30 bit wide register named UARTLCR, which is updated on a single write strobe generated by a UARTLCR_H write. So, in order to internally update the contents of the UARTIBRD or UARTFBRD registers, a write to UARTLCR_H must always be performed at the end.

Table 529. Truth table for SPS, EPS and PEN bits

Pen	Eps	Sps	Parity bit
1'b0	X	X	Not transmitted or checked.
1'b1	1'b1	1'b0	Even parity.
1'b1	1'b0	1'b0	Odd parity.

Table 529. Truth table for SPS, EPS and PEN bits (continued)

Pen	Eps	Sps	Parity bit
1‘b1	1‘b0	1‘b1	1
1‘b1	1‘b1	1‘b1	0

Note: Baud rate and line control registers (UARTIBRD, UARTFBRD and UARTLCR_H) must not be changed:
- when UART is enabled,
- when completing a transmission or a reception when it has programmed to become disabled.
Moreover, the FIFOs integrity is not guaranteed under the following conditions:
- after the BRK bit (in UARTLCR_H register) has been initiated,
- if the software disables the UART in the middle of a transmission with data in the FIFO and then re-enables it.

27.4.7 UARTCR register

The UARTCR (control) is a 16 bit RW register which allows to control the UART. The UARTCR bit assignments are given in *Table 530*.

Table 530. UARTCR register bit assignments

Bit	Name	Reset value	Description
[15]	CTSEn	1’h0	CTS hardware flow control enable. Setting this bit, the CTS hardware flow control is enabled and data is only transmitted when `nUARTCTS` signal is asserted.
[14]	RTSEn	1’h0	RTS hardware flow control enable. Setting this bit, the RTS hardware flow control is enabled and data is only requested when there is space in the Receive FIFO.
[13]	Out2	1’h0	Output. This bit is the complement of UART Out2 (`nUARTOut2`) modem status output (*Section 27.5*). Setting this bit, this output is 1‘b0. For DTE this can be used as Ring Indicator (RI).
[12]	Out1	1’h0	Out1. This bit is the complement of UART Out1 (`nUARTOut1`) modem status output (*Section 27.5*). Setting this bit, this output is 1‘b0. For DTE this can be used as Data Carrier Detect (DCD).
[11]	RTS	1’h0	Request to send. This bit is the complement of UART RTS (`nUARTRTS`) modem status output (*Section 27.5*). Setting this bit, this output is 1‘b0.
[10]	DTR	1’h0	Data transmit ready. This bit is the complement of UART DTR (nUARTDTR) modem status output (*Section 27.5*). Setting this bit, this output is 1‘b0.
[09]	RXE	1’h1	Receive enable. Setting this bit the receive section of UART is enabled. Data reception occurs for UART signals. When the UART is disabled in the middle of reception, it completes the current character before stopping.

Table 530. UARTCR register bit assignments (continued)

Bit	Name	Reset value	Description
[08]	TXE	1'h1	Transmit enable. Setting this bit the transmit section of UART is enabled. Data transmission occurs for UART signals. When the UART is disabled in the middle of transmission, it completes the current character before stopping.
[07]	LBE	1'h0	Loop back enable. Used together with test registers only.
[06:01]	Reserved	-	Read: as zero. Write: should be zero.
[00]	UARTEN	1'h0	UART enable. Setting this bit, the UART is enabled. Data transmission and reception occurs for UART signals. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Note: To enable transmission, both TXE (bit 8) and UARTEN (bit 0) must be set. Similarly, to enable reception, both RXE (bit 9) and UARTEN must be set.

Note: The UART CSRs should be programmed as follows:
- disable the UART (clearing the UARTEN bit in UARTCR register),
- wait for the end of transmission or reception of the current character,
- flush the Transmit FIFO by disabling the FEN bit in the UARTLCR_H register,
- program the UART CSRs (if required)
- enable the UART (setting the UARTEN bit in UARTCR register).

27.4.8 UARTIFLS register

The UARTIFLS (Interrupt FIFO level select) is a 16 bit RW register which defines the FIFO level at which the UARTTXINTR and UARTRXINTR interrupts are triggered (*Section 27.4*).

The interrupts are generated based on a transition through a level rather than being based on the level, that is, when the fill level progresses through the trigger level. The UARTIFLS bit assignments are given in *Table 531*.

Table 531. UARTIFLS register bit assignments

Bit	Name	Reset value	Description
[15:06]	Reserved	-	Read: as zero. Write: should be zero.

Table 531. UARTIFLS register bit assignments (continued)

Bit	Name	Reset value	Description
[05:03]	RXIFLSEL	3'h12	Receive interrupt FIFO level select. This 3 bit field allows to set the trigger points for the receive interrupt, according to encoding: – 3'b000 = 1/8 full – 3'b001 = 1/4 full – 3'b010 = 1/2 full (default) – 3'b011 = 3/4 full – 3'b100 = 7/8 full Any other value = Reserved
[02:00]	TXIFLSEL	3'h12	Transmit interrupt FIFO level select. This 3 bit field allows to set the trigger points for the transmit interrupt, according to encoding: – 3'b000 = 1/8 full – 3'b001 = 1/4 full – 3'b010 = 1/2 full (default) – 3'b011 = 3/4 full – 3'b100 = 7/8 full Any other value = Reserved

27.4.9 UARTIMSC register

The UARTIMSC (interrupt mask set/clear) is a 16 bit RW register which allows masking and clearing of each UART interrupt source (*Section 27.4*).

Reading from this register gives the current value of the mask on relevant interrupt. Writing a 1'b1 to a particular bit sets the corresponding mask of that interrupt, whereas writing a 1'b0 clears the corresponding mask.

The UARTIMSC bit assignments are given in *Table 532*.

Table 532. UARTIMSC register bit assignments

Bit	Name	Reset value	Description
[15:11]	Reserved	-	Read: as zero. Write: should be zero.
[10]	OEIM	1'h0	Overrun error interrupt mask.
[09]	BEIM	1'h0	Break error interrupt mask.
[08]	PEIM	1'h0	Parity error interrupt mask.
[07]	FEIM	1'h0	Framing error interrupt mask.
[06]	RTIM	1'h0	Receive timeout interrupt mask.
[05]	TXIM	1'h0	Transmit interrupt mask.
[04]	RXIM	1'h0	Receive interrupt mask.
[03]	DSRMIM	1'h0	nUARTDSR modem interrupt mask (see *Section 27.5*).
[02]	DCDMIM	1'h0	nUARTDCD modem interrupt mask (see *Section 27.5*).

Table 532. UARTIMSC register bit assignments (continued)

Bit	Name	Reset value	Description
[01]	CTSMIM	1'h0	nUARTCTS modem interrupt mask (see *Section 27.5*).
[00]	RIMIM	1'h0	nUARTRI modem interrupt mask (see *Section 27.5*).

27.4.10 UARTRIS register

The UARTRIS (raw interrupt status) is a 16 bit RO register which gives the current raw status value (prior to masking by UARTIMSC) of the corresponding interrupt. A write has no effect.

The UARTRIS bit assignments are given in *Table 533*.

Table 533. UARTRIS register bit assignments

Bit	Name	Reset value	Description
[15:11]	Reserved	-	Read: as zero.
[10]	OERIS	1'h0	Overrun error raw interrupt status.
[09]	BERIS	1'h0	Break error raw interrupt status.
[08]	PERIS	1'h0	Parity error raw interrupt status.
[07]	FERIS	1'h0	Framing error raw interrupt status.
[06]	RTRIS (1)	1'h0	Receive timeout raw interrupt status.
[05]	TXRIS	1'h0	Transmit raw interrupt status.
[04]	RXRIS	1'h0	Receive raw interrupt status.
[03]	DSRRMIS	1'h0	nUARTDSR modem raw interrupt status (see *Section 27.5*).
[02]	DCDRMIS	1'h0	nUARTDCD modem raw interrupt status (see *Section 27.5*).
[01]	CTSRMIS	1'h0	nUARTCTS modem raw interrupt status (see *Section 27.5*).
[00]	RIRMIS	1'h0	nUARTRI modem raw interrupt status (see *Section 27.5*).

1. The raw interrupt cannot be set unless the mask is set, because the mask acts as an enable for power saving.

Note: All the bits, except for the modem interrupt status (bit [3:0]), are cleared when reset. The modem interrupt status bits are undefined after reset.

27.4.11 UARTMIS Register

The UARTMIS (Masked Interrupt Status) is a 16 bit RO register which gives the current masked status value (after masking by UARTIMSC) of the corresponding interrupt. A write has no effect. The UARTMIS bit assignments are given in *Table 534*.

Table 534. UARTMIS register bit assignments

Bit	Name	Reset value	Description
[15:11]	Reserved	-	Read: as zero.
[10]	OEMIS	1'h0	Overrun error masked interrupt status.

Table 534. UARTMIS register bit assignments (continued)

Bit	Name	Reset value	Description
[09]	BEMIS	1'h0	Break error masked interrupt status.
[08]	PEMIS	1'h0	Parity error masked interrupt status.
[07]	FEMIS	1'h0	Framing error masked interrupt status.
[06]	RTMIS	1'h0	Receive timeout masked interrupt status.
[05]	TXMIS	1'h0	Transmit masked interrupt status.
[04]	RXMIS	1'h0	Receive masked interrupt status.
[03]	DSRMMIS	1'h0	nUARTDSR modem masked interrupt status (see *Section 27.5*).
[02]	DCDMMIS	1'h0	nUARTDCD modem masked interrupt status (see *Section 27.5*).
[01]	CTSMMIS	1'h0	nUARTCTS modem masked interrupt status (see *Section 27.5*).
[00]	RIMMIS	1'h0	nUARTRI modem masked interrupt status (see *Section 27.5*).

Note: All the bits, except for the modem interrupt status (bit [3:0]), are cleared when reset. The modem interrupt status bits are undefined after reset.

27.4.12 UARTICR register

The UARTICR (interrupt clear) is a 16 bit WO register which is able to clear the corresponding interrupt writing a 1'b1 to the appropriate field. A write of 1'b0 has no effect. The UARTICR bit assignments are given in *Table 535*.

Table 535. UARTICR register bit assignments

Bit	Name	Reset value	Description
[15:11]	Reserved	-	Write: should be zero.
[10]	OEIC	-	Overrun error interrupt clear.
[09]	BEIC	-	Break error interrupt clear.
[08]	PEIC	-	Parity error interrupt clear.
[07]	FEIC	-	Framing error interrupt clear.
[06]	RTIC	-	Receive timeout interrupt clear.
[05]	TXIC	-	Transmit interrupt clear.
[04]	RXIC	-	Receive interrupt clear.
[03]	DSRMIC	-	nUARTDSR modem interrupt clear (see *Section 27.5*).
[02]	DCDMIC	-	nUARTDCD modem interrupt clear (see *Section 27.5*).
[01]	CTSMIC	-	nUARTCTS modem interrupt clear (see *Section 27.5*).
[00]	RIMIC	-	nUARTRI modem interrupt clear (see *Section 27.5*).

27.4.13 UARTDMACR register

The UARTDMACR (DMA control) is the 16 bit RW DMA control register. The UARTDMACR bit assignments are given in *Table 536*.

Table 536. UARTDMACR register bit assignments

Bit	Name	Reset value	Description
[15:03]	Reserved	-	Read: as zero. Write: should be zero.
[02]	DMAONERR	1'h0	DMA on error. Setting this bit, the DMA receive request outputs (UARTRXDMASREQ or UARTRXDMABREQ) are disabled when UART error interrupt is asserted.
[01]	TXDMAE	1'h0	Transmit DMA enable. Setting this bit, DMA for the transmit FIFO is enabled.
[00]	RXDMAE	1'h0	Receive DMA enable. Setting this bit, DMA for the receive FIFO is enabled.

27.5 UART modem operation

UART is allowed to support both data terminal equipment (DTE) and data communication equipment (DCE) modes of operation. *Table 537* shows the meaning of the signals.

Table 537. Meaning of UART modem input/output in DTE and DCE modes

Signal	Meaning	
	DTE	DCE
nUARTCTS	Clear to send	Request to send
nUARTDSR	Data set ready	Data terminal ready
nUARTDCD	Data carrier detect	-
nUARTRI	Ring indicator	-
nUARTRTS	Request to send	Clear to send
nUARTDTR	Data terminal ready	Data set ready
nUARTOUT1	-	Data carrier detect
nUARTOUT2	-	Ring indicator

28 LS_I2C controller

This chapter describes the I2C Controller which is part of the low speed connectivity subsystem of the SPEAr^{TM} device. SPEAr300 has two I2C interfaces - one in fixed logic and another in the RAS subsystem.

28.1 Overview

The I2C Controller provides an APB interface to the processor to access the two-wire serial I2C bus. It can be used to connect any device which conforms to the I2C-Bus Specification from Philips.'

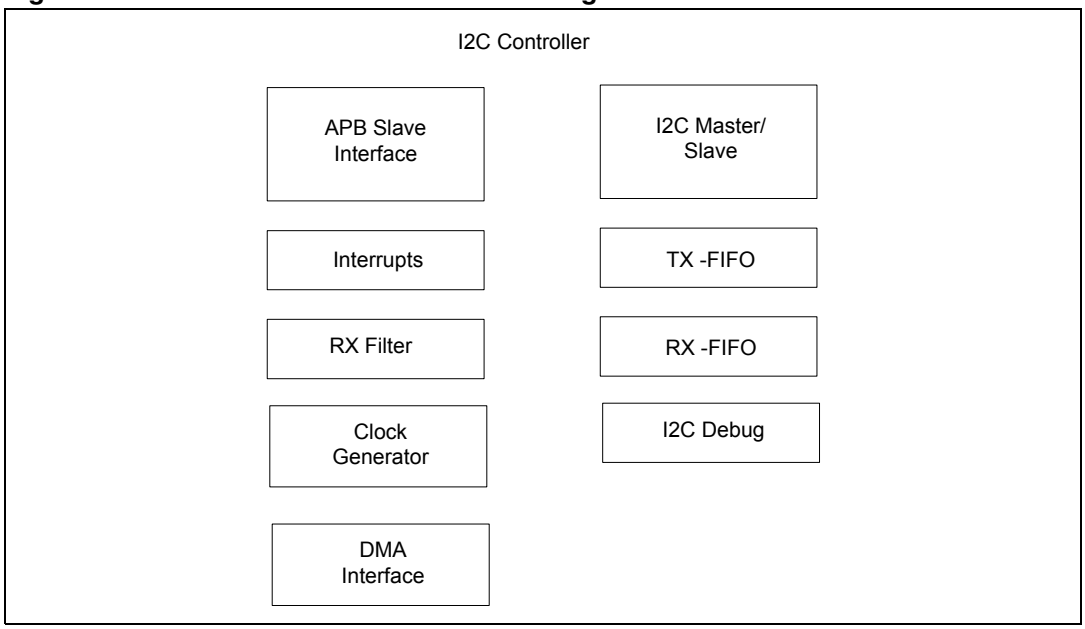
Main features of the I^2C controller are:

- Compliance to the I^2C-bus specification from Phillips.
- Operates in three different modes:
  - Standard-speed mode (data rates up to 100 Kb/s)
  - Fast-speed mode (data rates up to 400 Kb/s)
  - High-speed mode (data rates up to 3.4 Mb/s)
- Provides clock synchronization.
- Supports either master (only in a single master environment) or slave I^2C operation mode.
- Supports only slave operation in multi-master environment;
- Provides 7 bit or 10 bit addressing both in master and slave mode;
- Supports 7 bit or 10 bit combined format transfers.
- Provides slave bulk transfer mode.
- Ignores CBUS addresses (an older ancestor of I^2C that used to share the I^2C bus).
- Transmits and receives buffers.
- Provides interrupt or polled-mode operation.
- Handles bit and byte waiting at all bus speeds.
- Provides digital filter for the received SDA and SCL lines.
- Provides a DMA handshaking interface compatible with SPEAr^{TM} Basic's DMA Controller handshaking interface (see the Chapter on DMA Controller);
- Provides 8 or 16 bit wide transaction on the APB bus.

28.2 Block diagram

Figure 61 shows the functional block diagram of the I^2C controller.

Figure 61. I²C controller functional block diagram



28.3 Main functions description

This chapter describes the functional behavior of I2C in more detail.

28.3.1 APB interface

The host processor accesses data, control and status information on the I²C controller through the APB Slave Interface. The I²C controller has 16 bit APB data bus width.

28.3.2 I²C protocols

According to the I²C-bus specification, the I²C controller implements the following protocols:

- START and STOP Condition Protocol,
- Addressing Slave Protocol,
- Transmitting and Receiving Protocol,
- START Byte Transfer Protocol.

START and STOP condition protocol

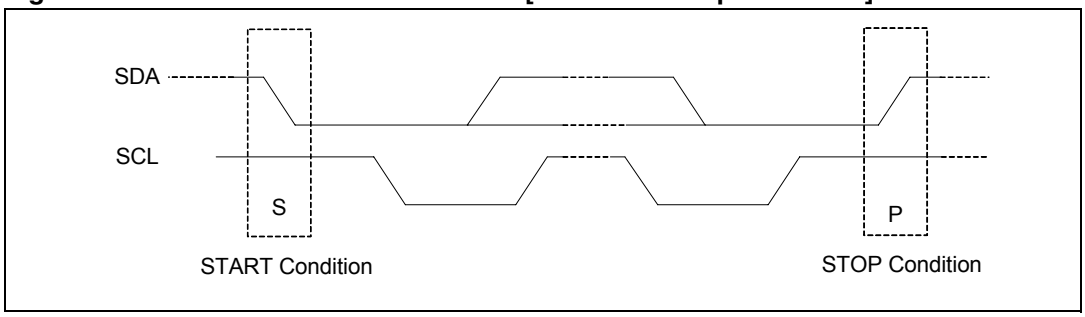
When the bus is IDLE, both the SCL (serial clock) and SDA (serial data) signals are pulled high through external pull-up resistors on the bus.

When the i2c master wants to start a transmission on the bus, it issues a START condition which is defined as a high-to-low transition of the SDA signal while SCL is high (see *Figure 62*).

When the i2c master wants to terminate the transmission, it issues a STOP condition which is defined as a low-to-high transition of the SDA signal while SCL is high.

When data is being transmitted on the bus, the SDA line must be stable when SCL is high.

Figure 62. START and STOP conditions [from I²C-bus specification]



Addressing slave protocol

Two address formats are supported: the 7 bit address format and the 10 bit address format.

In case of the 7 bit address format, the first seven bits (bits 7 to 1) of the first byte sent on the bus after the START condition set the slave address, while the LSB is the data direction bit. In particular, if LSB is set to ‘b0, the master writes to the slave (WRITE operation), otherwise (LSB set to ‘b1) the master reads from the slave (READ operation). Data is transmitted from the MSB.

In case of 10 bit addressing, two bytes are transferred following a START condition to set the 10 bit address. The first five bits (7 to 3) notify the slaves that this is a 10 bit transfer, followed by the next two bits (2 to 1) which set the bit 9 and 8 of the 10 bit slave address. The LSB of the first byte is the RW bit. *Table 538* lists the special purpose and reserved first byte addresses. The second byte transferred sets bits 7 to 0 of the 10 bit slave address.

Table 538. First byte assignment in addressing slave protocol

First byte sent		Description
BIT [7:1]	**RW BIT [0]**	
7’b0000 000	1‘h0	General call address. The I²C controller places the data in the receive buffer and issues a general call interrupt (*Section 28.5*).
7’b0000 000	1‘h1	START byte **(see Section: START byte transfer protocol)**.
7’h0000 001	1’hX	CBUS address. The I²C controller ignores these accesses.
7’h0000 010	1’hX	Reserved.
7’h0000 011	1’hX	Reserved.
7’h0000 1XX	1’hX	High-speed master code.
7’h1111 1XX	1’hX	Reserved.
7’h1111 0XX	1’hX	10 bit slave addressing.

Transmitting and receiving protocol

All data is transmitted in byte format, with no limits on the number of bytes transferred per data transfer. After the master sends the slave address and the data direction bit, or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal after every byte of data is received. When a slave-receiver does not respond with an acknowledge pulse, the master aborts the transfer by issuing a STOP condition. The slave shall leave the SDA line high so the master can abort the transfer.

If the master is receiving data, then the master-receiver responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the no acknowledge so that the master-receiver can issue a STOP condition.

When the master does not want to relinquish the bus with a STOP condition, the master can issue a repeated start condition. This is identical to a START condition except it occurs after the acknowledge pulse. The master can then communicate with the same slave or with a different slave.

START byte transfer protocol

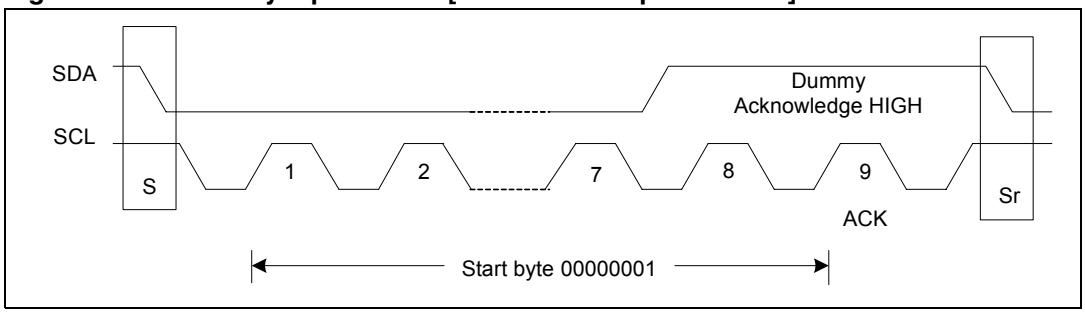
The “START byte’ transfer protocol is set up for systems that do not have an on-board dedicated I^2C hardware module. In this case, the systems can’t be only interrupted by requests from the I^2C bus, but it must constantly monitor the bus (software polling).

When the I^2C controller is addressed as a slave it always samples the I^2C bus at the highest speed supported, so that it never requires START byte transfer. However, when the I^2C controller is a master, it supports the generation of START byte transfer at the beginning of every transfer in case a slave device requires it.

As depicted in, the start procedure is as follows:

- Master generates a start condition (as explained above),
- Master transmits the start byte (constant 8’b0000 0001),
- Master transmits an acknowledge clock pulse,
- No slave sets the acknowledge signal to 1‘b0,
- Master generates a repeated START (Sr) condition.

Figure 63. START byte procedure [from I^2C-bus specification]



The START byte protocol consist of seven zeros being transmitted followed by a ‘b1 (the start byte). This allows the system processor that is polling the bus to under-sample the address phase until ‘b0 (low level on SDA) is detected. Once the system processor detects a low level on SDA, it switches to a higher sampling rate to find the Sr condition of the master (which is the used for synchronization).

A hardware receiver does not respond to the START byte because it is a reserved address and it resets after the Sr (restart condition) is generated.

28.3.3 DMA controller interface

The I2C Controller has a handshaking interface to request and control transfers from the DMA Controller of the SPEAr™ device. The APB bus is used to perform the data transfer to or from the DMA.

Note: When DMA Controller of the SPEAr™ device is used for data transfers to/from the I2C Controller, the DMA Controller must always be programmed as the flow controller; that is, the DMA Controller controls the block size.

Enabling the DMA controller interface

To enable the DMA Controller interface on the I2C, the DMA Control Register (IC_DMA_CR, *Section 28.6.25*) has to be set. Writing a 'b1 into the TDMAE bit field of IC_DMA_CR register, it enables the I2C Controller transmit handshaking interface. Besides, writing a 'b1 into the RDMAE bit field of the IC_DMA_CR register, the I2C Controller receive handshaking interface is enabled.

Transmit watermark level and transmit FIFO underflow

During I2C Controller serial transfers, transmit FIFO requests are made to the DMA Controller whenever the number of entries in the transmit FIFO is less than or equal to the DMA Transmit Data Level Register (IC_DMA_TDLR, *Section 28.6.26*) value, this is known as the "watermark level".

The DMA Controller responds by writing a burst of data to the transmit FIFO buffer. Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty another DMA request should be triggered. Otherwise, the FIFO will run out of data (underflow). To prevent this condition, the user must set the watermark level correctly."

Choosing the Transmit watermark level

The goal in choosing a watermark level is to minimize the number of transactions per block, keeping the probability of an underflow condition to an acceptable level.

In practice, this is a function of the ratio of the rate at which the I2C transmits data to the rate at which the DMA can respond to destination burst requests. For example, promoting the channel to the highest priority channel in the DMA, and promoting the DMA master interface to the highest priority master in the AMBA layer, increases the rate at which the DMA controller can respond to burst transaction requests. This in turn allows the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

Receive watermark level and receive FIFO overflow

During I2C Controller serial transfers, receive FIFO requests are made to the DMA Controller whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register (IC_DMA_RDLR, *Section 28.6.27*), that is DMARDLR + 1, which is known as the "watermark level".

The DMA Controller responds by writing a burst of data to the transmit FIFO buffer. Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers

continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO will fill with data (overflow). To prevent this condition, the user must correctly set the watermark level."

Choosing the receive watermark level

Similar to choosing the transmit watermark level described earlier, the receive watermark level should be set to minimize the probability of overflow. It is a trade-off between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

28.4 Operation modes

The I^2C interface protocol is setup with a master and a 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 the master or the slave. The protocol also allows multiple masters to reside on the I^2C bus, which requires the masters to arbitrate for ownership.

According to this specification, the I^2C controller provided by the device supports three distinct operation modes, specifically:

- Slave mode (*Section 28.4.1: Slave mode on page 612*).
- Master mode (*Section 28.4.2*).
- Multi-master mode (*Section 28.4.3*).

28.4.1 Slave mode

This section provides information on the slave mode of operation.

Initial configuration

In order to use the I^2C controller as a slave, the following steps have to be performed:

- Disable the I^2C controller by writing a ‘b0 to the IC_ENABLE register (*Section 28.6.21*),
- Write to the IC_SAR register (*Section 28.6.5*) to set the slave address (only required if the slave address to be programmed is other than 0x0055). This is the address to which the I^2C controller responds,
- Write to the IC_CON register (*Section 28.6.3*) to specify whether 10 bit addressing is supported (through IC_10BITADDR_SLAVE bit) and whether the I^2C controller is in slave-only or master-slave mode. The master-only mode is not valid in slave mode,
- Then, enable the I^2C controller setting the IC_ENABLE register (*Section 28.6.21*).

Slave-transmitter operation

When another master addresses the I^{2}C controller to requests data, the I^{2}C controller acts as a “slave-transmitter,” and the following steps occur:

- The other master initiates an I^{2}C transfer with an address that matches the slave address in the IC_SAR register (*Section 28.6.5*) of the slave I^{2}C controller (programmed during initial configuration),
- The I^{2}C controller acknowledges the sent address and recognizes the direction of transfer to indicate that it is acting as a slave-transmitter,
- The I^{2}C controller asserts the RD_REQ interrupt, (and relevant bit in IC_RAW_INTR_STAT register, *Section 28.6.16*, is set) and holds the SCL line low, placing in a wait state until software responds,
- If there is any data remaining in the Transmit FIFO before receiving the read request, then the I^{2}C controller asserts a TX_ABRT interrupt, section, (and relevant bit in IC_RAW_INTR_STAT register, *Section 28.6.16*, is set) to flush the old data from the Transmit FIFO,
- Software then writes the IC_DATA_CMD register (*Section 28.6.7*) with the data to be written, setting to 1‘b0 the CMD bit (meaning a write operation),
- Software should clear the RD_REQ and the TX_ABRT interrupts before proceeding,
- The I^{2}C controller releases the SCL and transmits the byte,
- Finally, the master may hold the I^{2}C bus by issuing a restart condition or release the bus by issuing a stop condition.

Slave-receiver operation

When another master addresses the I^{2}C controller to send its data, the I^{2}C controller acts as a slave-receiver and the following steps occur:

- The other master initiates an I^{2}C transfer with an address that matches the I^{2}C controller slave address in the IC_SAR register (*Section 28.6.5*), programmed during initial configuration,
- The I^{2}C controller acknowledges the sent address and recognizes the direction of transfer to indicate that it is acting as a slave-receiver,
- The I^{2}C controller receives the transmitted byte from the master and place it in the receive buffer, assuming there is room for this incoming data,
- The status and interrupt bits corresponding to the receive buffer are updated,
- Software may read the received byte from the IC_DATA_CMD register (*Section 28.6.7*), setting to 1‘b1 the CMD bit (meaning a read operation),
- Finally, the other master may hold the I^{2}C bus by issuing a restart condition or release the bus by issuing a stop condition.

Slave bulk transfer mode

In the standard I^{2}C protocol, all transaction are single byte transactions and a remote master read request is replied by writing one byte into the transmit FIFO.

In the mode named Slave Bulk Transfer, if the remote master acknowledged the sent byte to request more data, then the slave must hold the I^{2}C SCL line low and request another byte from the processor side. If it is known in advance that the remote master is requesting a packet of n bytes, then when another master addresses the I^{2}C controller and request data, the Transmit FIFO could be written with 16 number of bytes and the remote master will receive it as a continuous stream of data.

If the remote master is to receive n bytes from the I²C controller but a number of bytes larger than n is written to the Transmit FIFO then, when the slave finishes sending the requested n bytes, it will clear the Transmit FIFO and ignore any excess bytes.

28.4.2 Master mode

Initial configuration

In order to use the I²C controller as a master, the following steps have to be performed:

- Disable the I²C controller by writing a 1‘b0 to the IC_ENABLE register (*Section 28.6.21*),
- Write to the IC_SAR register (*Section 28.6.5*) to set the slave address at which I²C controller responds (only required if the slave address is other than 0x0055),
- (for master mode) write to the IC_CON register (*Section 28.6.3*) to set the maximum speed mode supported for slave operation and the desired speed of the I²C controller master-initiated transfers, 7 or 10 bit addressing,
- (for slave mode) write to the IC_CON register (*Section 28.6.5*)to set the maximum speed mode supported for slave operation and whether the I²C controller starts transfers in 7- or 10 bit addressing mode when a slave,
- Write to the IC_TAR register (*Section 28.6.4*) the address of the I²C device to be addressed by the I²C controller as a master (10 bit field IC_TAR). It also indicates whether adding a START BYTE or issuing a general call is going to occur (through the GC_OR_START bit on the same register), The desired speed of the I2C Controller master-initiated transfer is controlled by the IC_10BITADDR_MASTER bit in the IC_TAR register,
- For high-speed mode transfer only: the desired master code for I²C controller must be written to the IC_HS_MADDR register (3 bit IC_HS_MAR field, *Section 28.6.6*),
- Enable again the I²C controller setting the IC_ENABLE register (*Section 28.6.21*),
- Command and data to be sent may be written now to the IC_DATA_CMD register (*Section 28.6.7*). If this register is written before I²C controller is enabled, the data and commands are lost as the buffers are kept cleared when I²C controller is not enabled.

Dynamic IC_TAR or IC_10BITADDR_MASTER update

The I²C controller supports a dynamic IC_TAR or IC_10BITADDR_MASTER update.

- Even if the slave part of the DW_apb_i2c is involved in an I²C transfer, both of the following must occurs:
  - MST_ACTIVITY must be IDLE, that is, IC_STATUS[5] = ‘b0 (see *Section 28.6.22*),
  - Transmit FIFO completely empty must occurs, that is, IC_STATUS[2] = ‘b0 (see *Section 28.6.22*).

Note: If a bulk read is performed on the slave part of the DW_apb_i2c over the I²C bus, then only MST_ACTIVITY must be IDLE. that is, the Transmit FIFO does not need to be completely empty. This is a very specific case and should be monitored in software.

- Dynamically write the IC_TAR and IC_10BITADDR_MASTER using the following requirements for writing to the IC_TAR register (see *Section 28.6.4*):
  - IC_TAR[12] = IC_10BITADDR_MASTER. Master uses 7 or 10 bit addressing and is writable when the conditions in step 1 are met and if I2C_DYNAMIC_TAR_UPDATE is set to ‘b1,
  - IC_TAR[11:10] = Only writable when DW_apb_i2c interface is disabled, which corresponds to the IC_ENABLE register (see *Section 28.6.21*) being set to ‘b0. otherwise writes have no effects,
  - IC_TAR[9:0] = IC_TAR. Master’s 10 bit target address is writable at any time when the conditions in step 1 are met and if I2C_DYNAMIC_TAR_UPDATE is set to ‘b1.

Master transmit and master receive

The I²C controller supports switching back and forward between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the IC_DATA_CMD register (*Section 28.6.7*). The CMD bit in the same register should be written to ‘b0 meaning a write operation. Subsequently, a read command may be issued by writing “don’t cares” to the lower byte of the IC_DATA_CMD register, and a ‘b1 should be written to the CMD bit.

As data is transmitted and received, transmit and receive buffer status bits and interrupts change accordingly.

28.4.3 Multi-master mode

In a multiple master I2C bus system, the I2C Controller should not be programmed as a master device. For multiple master systems, the I2C Controller can only be operated as a slave (set IC_CON.MASTER_MODE to 0 (master disabled).

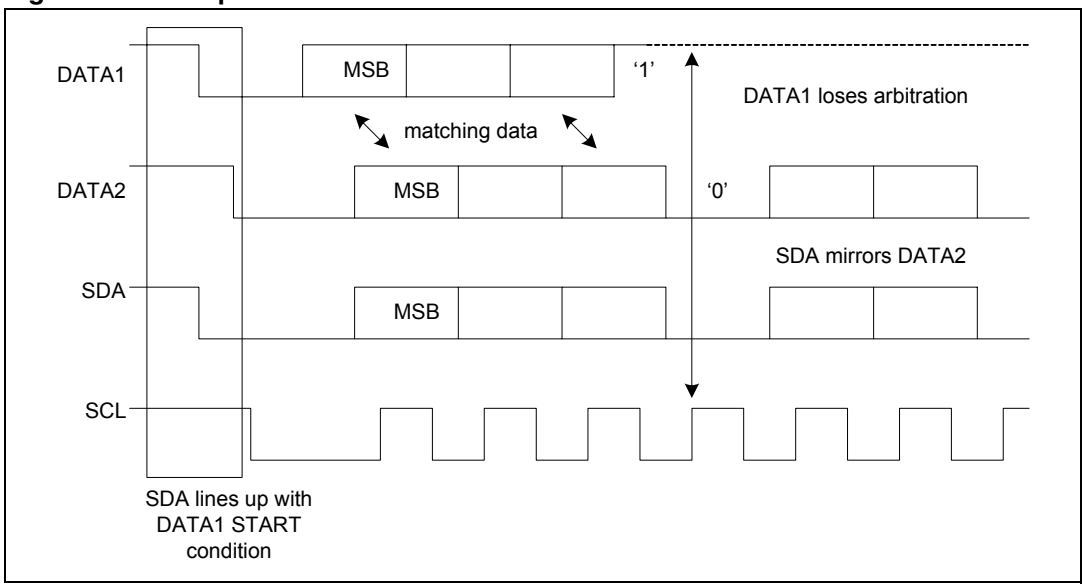
The I²C controller bus protocol allows multiple masters to reside on the same bus. When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock.

Master arbitration

Arbitration takes place on the SDA line, while the SCL line is ‘b1. The master, which transmits a ‘b1 while the other master transmits ‘b0, loses arbitration and turns off its data output. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

For high-speed mode, the arbitration can not go into the data phase, because each master is programmed with a different high-speed master code. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code.

Figure 64. Multiple master arbitration



Clock synchronization

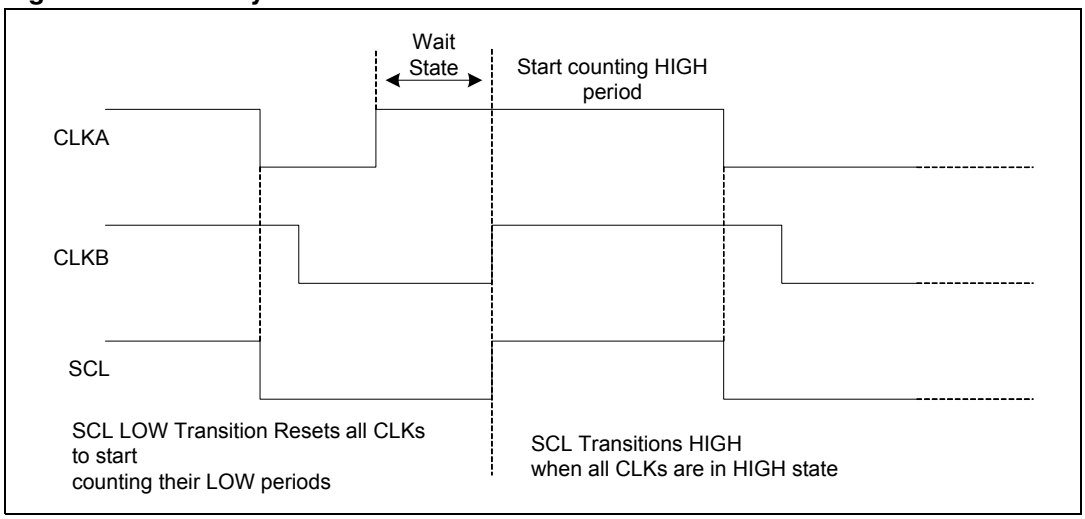
All masters generate their own clock to transfer messages, and data is valid only during the high period of SCL clock.

Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 'b0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 'b1 at the beginning of the next clock period. However, if another master is holding the SCL line to 'b0, then the master goes into a high wait state until the SCL clock line transitions to 'b1.

All masters then count off their high time, and the master with the shortest high time transitions the SCL line to 'b0. The masters counts out their low time and the one with the longest low time forces the other master into a high wait state. Therefore, a synchronized SCL clock is generated.

Note: Optionally, slaves may hold the SCL line low to slow down the timing on the I^2C bus.

Figure 65. Clock synchronization



28.5 Interrupt sources

The following *Table 539* lists the interrupt generated within the I²C controller. These interrupt sources could be masked using the IC_INTR_MASK register (*Section 28.6.15*).

Interrupts status (after masking) and raw interrupts status (before masking) are available through the IC_INTR_STAT register (*Section 28.6.14*) and the IC_RAW_INTR_STAT register (*Section 28.6.16*), respectively.

Table 539. I²C controller interrupt sources

Name	Source
GEN_CALL	General call request received. Indicates that a general call request was received (refer to *Section 28.3.2: I2C protocols on page 608*). The I²C controller stores the received data in the Receive buffer.
START_DET	START condition occurred. Indicates that a START condition has occurred on the I²C interface (refer to *Section 28.3.2: I2C protocols on page 608*).
STOP_DET	STOP condition occurred. Indicates that a STOP condition has occurred on the I²C interface (refer to *Section 28.3.2: I2C protocols on page 608*).
ACTIVITY	Capture system activity. This bit captures I²C controller activity and it remains set until it is cleared, regardless of the I²C controller going idle.
RX_DONE	Indicates transmission done. This bit is set to 1'b1 if the master does not acknowledge a transmitted byte, while I²C controller is acting as a slave-transmitter. This occurs on the last byte of the transmission, indicating that the transmission is done.

Table 539. I²C controller interrupt sources (continued)

Name	Source
TX_ABRT	Indicates transmission abort. This bit is set to ‘b1 when the I²C controller, acting as a master, is unable to complete a command that the processor has sent. Several conditions could cause this interrupt to be issued: – no slave acknowledge after the address is sent, – the address slave does not acknowledge a byte of data, – arbitration is lost, – attempting to send a master command when configured only to be slave, – IC_RESTART_EN bit in the IC_CON register (*Section 28.6.3*) is set to ‘b0 (restart condition disabled), and the processor attempts to issue an I²C function that is impossible to perform without using restart conditions, – high-speed master code is acknowledged, – start byte is acknowledged, – general call address is not acknowledged, – when a read request interrupt occurs and the processor has previously placed the data in transmit buffer that has not been transmitted yet. This data could have been intended to service a multi-byte RD_REQ that ended up having fewer numbers of byte requested. Or, if IC_RESTART_EN is disabled and the I²C loses control of the bus between transfers and is then accessed as a slave-transmitter, – if a read command is issued after a general call command has been issued. Disabling the I²C reverts it back to normal operation, – if the processor attempts to issue read command before a RD_REQ is serviced. Anytime this bit is set, the contents of both transmit and receive buffers are flushed.
RD_REQ	Read request. This bit is set to ‘b1 when the I²C controller is acting as a slave and another I²C master is attempting to read data from our module. The I²C controller holds the I²C bus in waiting state (SCL tied to low) until this interrupt is serviced. The processor must acknowledge this interrupt and then write the request data to the IC_DATA_CMD register.
TX_EMPTY	Transmit buffer at threshold value. This bit is set to ‘b1 when the transmit buffer is at or below the threshold value set in the IC_TX_TL register (*Section 28.6.18*). It is automatically cleared by hardware when buffer level goes above the threshold.
TX_OVER	Transmit buffer filled to IC_TX_BUFFER_DEPTH. This bit is 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.
RX_FULL	Transmit buffer reach RX_TL threshold. This bit is set when the transmit buffer reaches or goes above the threshold set in the IC_RX_TL register (*Section 28.6.17*). It is automatically cleared by hardware when buffer level goes below the threshold.

Table 539. I²C controller interrupt sources (continued)

Name	Source
RX_OVER	Receive buffer filled to IC_RX_BUFFER_DEPTH. This bit is set when the receive buffer was completely filled to IC_RX_BUFFER_DEPTH and more data arrived. The data is lost.
RX_UNDER	Receive buffer empty. This bit is set when the processor attempts to read the receive buffer when it is empty by reading from the IC_DATA_CMD register.

28.6 Programming model

The i2c can be programmed via software registers. For information about programming the software registers in terms of i2c operation. refer to Slave mode operation on *Section 28.4.1: Slave mode on page 612* and master mode operation on *Section 28.4.2: Master mode on page 614*. The software registers are described in more detail in *Section 28.6.2: Register map*.

28.6.1 External pin connections

At the chip boundary, the external I2C peripherals can be connected on the following pins:

Table 540. External pin connections(1)

Signal	Ball
SCL	C1
SDA	D2

1. See PL_GPIO Sharing Schemes to verify the availability of the external signals.

28.6.2 Register map

The I²C controller can be fully configured by programming its 16 bit registers which can be accessed at the base address 0xD018_0000.

Table 541. I2C registers

Name	Offset	Width (bit) (1)1	Type	Reset value	Description
IC_CON (*Section 28.6.3*)	0x000	7	RW	7'h2F	I²C control.
IC_TAR (*Section 28.6.4*)	0x004	13	RW	13'h0055	I²C target address.
IC_SAR (*Section 28.6.5*)	0x008	10	RW	10'h0055	I²C slave address.
IC_HS_MADDR (*Section 28.6.6*)	0x00C	3	RW	3'b001	I²C HS master mode core address.
IC_DATA_CMD (*Section 28.6.7*)	0x010	9	RW	9'h0	I²C RX/TX data buffer and command.
IC_SS_SCL_HCNT (*Section 28.6.8*)	0x014	16	RW	16'h029b	Standard-Speed I2C Clock SCL High Count.

Table 541. I2C registers (continued)

Name	Offset	Width (bit) (1)1	Type	Reset value	Description
IC_SS_SCL_LCNT (*Section 28.6.9*)	0x018	16	RW	16'h0310	Standard-Speed I2C Clock SCL Low Count.
IC_FS_SCL_HCNT (*Section 28.6.10*)	0x01C	16	RW	16'h0064	Fast-Speed I2C Clock SCL High Count.
IC_FS_SCL_LCNT (*Section 28.6.11*)	0x020	16	RW	16'h00d9	Fast-Speed I2C Clock SCL Low Count.
IC_HS_SCL_HCNT (*Section 28.6.12*)	0x024	16	RW	16'h000a	High-Speed I2C Clock SCL High Count.
IC_HS_SCL_LCNT (*Section 28.6.12*)	0x028	16	RW	16'h001b	High-Speed I2C Clock SCL Low Count.
IC_INTR_STAT (*Section 28.6.14*)	0x02C	12	RO	12'h0	I^2C interrupt status.
IC_INTR_MASK (*Section 28.6.15*)	0x030	12	RW	12'h8ff	I^2C interrupt mask.
IC_RAW_INTR_STAT (*Section 28.6.16*)	0x034	12	RO	12'h0	I^2C raw interrupt status.
IC_RX_TL (*Section 28.6.17*)	0x038	8	RW	8'h00	I^2C receive FIFO threshold.
IC_TX_TL (*Section 28.6.18*)	0x03C	8	RW	8'h00	I^2C transmit FIFO threshold.
IC_CLR_INTR (*Section 28.6.19*)	0x040	1	RO	1'b0	Clear combined and Individual Interrupts.
IC_CLR_RX_UNDER	0x044	1	RO	1'b0	Clear RX_UNDER interrupt.
IC_CLR_RX_OVER	0x048	1	RO	1'b0	Clear RX_OVER interrupt.
IC_CLR_TX_OVER	0x04C	1	RO	1'b0	Clear TX_OVER interrupt.
IC_CLR_RD_REQ	0x050	1	RO	1'b0	Clear RD_REQ interrupt.
IC_CLR_TX_ABRT	0x054	1	RO	1'b0	Clear TX_ABRT interrupt.
IC_CLR_RX_DONE	0x058	1	RO	1'b0	Clear RX_DONE interrupt.
IC_CLR_ACTIVITY	0x05C	1	RO	1'b0	Clear ACTIVITY interrupt.
IC_CLR_STOP_DET	0x060	1	RO	1'b0	Clear STOP_DET interrupt.
IC_CLR_START_DET	0x064	1	RO	1'b0	Clear START_DET interrupt.
IC_CLR_GEN_CALL	0x068	1	RO	1'b0	Clear GEN_CALL Interrupt
IC_ENABLE (*Section 28.6.21*)	0x06C	1	RW	1'b0	I2C Enable.

Table 541. I2C registers (continued)

Name	Offset	Width (bit) (1)1	Type	Reset value	Description
IC_STATUS(*Section 28.6.22*)	0x070	7	RO	7'h06	I2C Status
IC_TXFLR (*Section 28.6.23*)	0x074	4	RO	4'h0	Transmit FIFO Level.
IC_RXFLR (*Section 28.6.23*)	0x078	4	RO	4'h0	Receive FIFO Level.
-	0x07C	-	-	-	Reserved
IC_TX_ABRT_SOURCE (*Section 28.6.24*)	0x080	16	RW	16'h0	I2C Transmit Abort Status.
-	0x084	-	-	-	Reserved
IC_DMA_CR (*Section 28.6.25*)	0x088	2	RW	2'h0	DMA Control.
IC_DMA_TDLR (*Section 28.6.26*)	0x08C	3	RW	3'b000	DMA Transmit Data Level.
IC_DMA_RDLR(*Section 28.6.27*)	0x090	3	RW	3'b000	DMA Receive Data Level.
-	0x094 to 0x0F0	-	-	-	Reserved
IC_COMP_PARAM_1(*Section 28.6.28*)	0x0F4	32	RO	32'h000707ED	Component Parameter.
IC_COMP_VERSION	0x0F8	32	RO	32'h3130352A	Component Version ID.
IC_COMP_TYPE	0x0FC	32	RO	32'h44570140	DW Component Type.

1. This value represents the actual number of used bits, being reserved the others to 16.

28.6.3 IC_CON register(0x000)

The IC_CON is a RW register which allows to control the I^2C controller. The IC_CON bit assignments are given in *Table 542*.

Note: This register can be written only when the I^2C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to 1'b0. Write at other times has no effect.

Table 542. IC_CON register bit assignments

Bit	Name	Type	Reset value	Description
[15:07]	Reserved		-	Read: undefined. Write: should be zero.
[06]	IC_SLAVE_DISABLE	RW	1'h0	Slave disabled after reset. This bit controls whether the I^2C controller has its slave disabled after reset, according to the encoding: 1'b0 = Enabled (default). 1'b1 = Disabled.
[05]	IC_RESTART_EN	RW	1'h1	Enable restart conditions (when acting as master). This bit determines whether restart conditions may be sent (if set to 'b1) when acting as a master or not (if set to 'b0). Indeed, some older slaves do not support handling restart conditions. Note: Disabling a restart does not allow the master to perform the following function: ● send multiple bytes per transfer (split), ● change direction within a transfer (split), ● send a start byte, ● perform any high-speed mode operation, ● perform combined format transfers in 7- or 10 bit addressing mode (split for 7 bit), ● perform a read operation with a 10 bit address. ● Split operations are broken down into multiple I^2C transfers with a stop and start condition in between. The other operations are not performed at all and result in setting TX_ABRT.
[04]	IC_10BITADDR_MASTER	RO	1'h0	10 bit addressing mode (when acting as master). The function of this bit is handled by bit 12 of IC_TAR. This bit is a read-only field called IC_10BITADDR_MASTER_rd_only.
[03]	IC_10BITADDR_SLAVE	RW	1'h1	Responds to 7- or 10 bit addresses (when acting as slave). This bit controls if I^2C controller responds to either 7- or 10 bit addresses when acting as a slave, according to the encoding: 1'b0 = 7. The I^2C controller ignores transactions which involve 10 bit addressing. for 7 bit addressing, only the lower 7 bits of the IC_SAR register (*Section 28.6.5*) are compared. 1'b1 = 10. The I^2C controller responds to only 10 bit addressing transfers that match the full 10 bits of the IC_SAR register.

Table 542. IC_CON register bit assignments (continued)

Bit	Name	Type	Reset value	Description
[02:01]	SPEED	RW	2'h11	Controls operation speed. This 2 bit field controls at which speed the I^2C controller operates, according to the encoding: 'b00, Illegal = - 'b01, Standard = 100 kbit/s 'b10, Fast = 400 kbit/s 'b11, High = 3.4 Mbit/s (default) If the device is configured for fast or standard mode and value 3 is written, then the IC_MAX_SPEED_MODE is stored. If an APB write is performed to these bits such that the data is decimal 2 or 3, then these would change the maximum speed mode. Hardware prevents this fact and writes in the value of IC_MAX_SPEED_MODE instead. The value of IC_MAX_SPEED_MODE is configured to be 'b11.
[00]	MASTER_MODE	RW	1'h1	Enable master. This bit controls if the I^2C controller is enabled to act as master, according to the encoding: 1'b0 = Disabled. 1'b1 = Enabled (default)

Note: The I^2C controller slave is always enabled.

28.6.4 IC_TAR register(0x004)

The IC_TAR (I^2C target address) is a RW register. The IC_TAR bit assignments are given in *Table 543*.

Note: 1 Bit 12 and bits 9 through 0 can be dynamically updated as long as the following are true:

2 MST_ACTIVITY must be IDLE. that is, IC_STATUS[5] = 1'b0 (see Section 28.6.22)

Transmit FIFO completely empty must occur. that is, IC_STATUS = 'b0.

Bits 10 and 11 are writable only when IC_ENABLE[0] = 1'b0 (see Section 28.6.21).

Table 543. IC_TAR register bit assignments

Bit	Name	Type	Reset value	Description
[15:13]	Reserved		-	Read: undefined. Write: should be zero.
[12]	IC_10BITADDR_MASTER	RW	1'h0	10 bit addressing mode (when acting as master). this bit controls whether DW_apb_i2c starts its transfer in 10 bit addressing mode when acting as a master according to the encoding below: 1'b0 = 7. 1'b1 = 10.
[11]	SPECIAL	RW	1'h0	Perform a general call or start byte I^2C command. This bit indicates whether software would like to either perform a general call or start byte I^2C command, according to the encoding: 1'b0 = Ignore bit[10], GC_OR_START, in this register and use IC_TAR normally. 1'b1 = Perform special I^2C command as specified in GC_OR_START bit.
[10]	GC_OR_START	RW	1'h0	Indicates when a general call or start byte I^2C command is to be performed. If bit[11], SPECIAL, in this register is set to 'b1, the GC_OR_START bit indicates whether a general call or start byte command is to be performed by the I^2C controller, according to the encoding below: 1'b0 = General Call Address: after issuing a general call, only writes may be performed. Attempting to issue a read command result in setting TX_ABRT. The I^2C controller remains in general call mode until the SPECIAL bit value is cleared. 1'b1 = Start byte.
[09:00]	IC_TAR	RW	10'h055	Target address. This 10 bit field is the target address for any master transactions. Its reset value indicates loopback mode.

28.6.5 IC_SAR register(0x008)

The IC_SAR is the 10 bit RW register which holds the slave address. The I^2C controller responds to this address when it is operating as a slave. In case of 7 bit addressing (IC_10BITADDR_SLAVE bit set to 'b0 in IC_CON register, *Section 28.6.3*), only bits [6:0] are used. The IC_SAR bit assignments are given in *Table 544*.

Note: This register can be written only when the I^2C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to 'b0. Write at other times has no effect.

Table 544. IC_SAR register bit assignments

Bit	Name	Type	Reset value	Description
[15:10]	Reserved		-	Read: undefined. Write: should be zero.
[09:00]	IC_SAR	RW	10'h55	Slave address.

28.6.6 IC_HS_MADDR register(0x00C)

The IC_HS_MADDR is the RW register which holds the 3 bit value of the I^2C master code in HS (high-speed) mode. The IC_HS_MADDR bit assignments are given in *Table 545*.

Note: This register can be written only when the I^2C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to 'b0. Write at other times has no effect.

Table 545. IC_HS_MADDR register bit assignments

Bit	Name	Type	Reset value	Description
[15:03]	Reserved		-	Read: undefined. Write: should be zero.
[02:00]	IC_HS_MAR	RW	3'b001	I^2C HS mode master code.

28.6.7 IC_DATA_CMD register(0x010)

The IC_DATA_CMD is a RW register which contains the I^2C Rx/Tx data buffer and related read/write command. The IC_DATA_CMD bit assignments are given in *Table 546*.

Table 546. IC_DATA_CMD register bit assignments

Bit	Name	Type	Reset value	Description
[15:09]	Reserved		-	Read: undefined. Write: should be zero.

Table 546. IC_DATA_CMD register bit assignments (continued)

Bit	Name	Type	Reset value	Description
[08]	CMD	RW	1‘h0	Control read or write. This bit controls whether a read or write is performed, according to the encoding: 1‘b0 = Write. 1‘b1 = Read. Note: In case of reading, the lower bits from 7 to 0 (DAT field) are ignored by the I^2C controller. Reading this bit returns 'b0. Attempting to perform a read operation after a general call command has been sent results in TX_ABRT unless the SPECIAL bit in IC_TAR register (see *Section 28.6.4*) has been cleared. If this bit is written to ‘b1 after receiving RD_REQ, then a TX_ABRT occurs.
[07:00]	DAT	RW	7‘h0	Contains data. This 8 bit field contains the data to be transmitted or received on the I^2C bus. Read these bits means reading out the data received on the I^2C interface. Write this field means sending data out on the I^2C interface.

28.6.8 IC_SS_SCL_HCNT register (0x014)

The IC_SS_SCL_HCNT is a 16 bit RW register which allows setting the high period of the SCL clock for standard-speed mode. The IC_SS_SCL_HCNT bit assignments are given in *Table 547*.

Note: 1 This register can be written only when the I^2C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to ‘b0. Write at other times has no effect.

2 This register must be set before any I^2C bus transaction can take place in order to ensure proper I/O timing.

Table 547. IC_SS_SCL_HCNT register bit assignments

Bit	Name	Type	Reset value	Description
[15:00]	IC_SS_SCL_HCNT	RW	16'h29B	SCL clock high period count for standard speed. This 16 bit field states the SCL clock high period count for standard speed. The minimum valid value is 6, and hardware prevents that a value less than this minimum will be written (setting 6 if attempted)

Table 548. IC_SS_SCL_HCNT sample calculations

I²C data rate - SS (Kbps)	SCL clock frequency (MHz)	SCL high time required min (µs)	IC_SS_SCL_HCNT (hex/decimal)	SCL high time actual (µs)
100	2	4	16‘h0008/’d8	4.00
100	6.6	4	16‘h001B/’d27	4.09
100	10	4	16‘h0028/’d40	4.00
100	75	4	16‘h012C/’d300	4.00
100	100	4	16‘h0190/’d400	4.00
100	125	4	16‘h01F4/’d500	4.00
100	1000	4	16’h0FA0/’d4000	4.00

28.6.9 IC_SS_SCL_LCNT register(0x018)

The IC_SS_SCL_LCNT is a 16 bit RW register which allows to set the low period of the SCL clock for standard-speed mode. The IC_SS_SCL_LCNT bit assignments are given in *Table 549*.

Note: 1 This register can be written only when the I²C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to ‘b0. Write at other times has no effect.

2 This register must be set before any I²C bus transaction can take place in order to ensure proper I/O timing.

Table 549. IC_SS_SCL_LCNT register bit assignments

Bit	Name	Type	Reset value	Description
[15:00]	IC_SS_SCL_LCNT	RW	16'h310	SCL clock low period count for standard speed. This 16 bit field states the SCL clock low period count for standard speed. The minimum valid value is 8, and hardware prevents that a value less than this minimum will be written (setting 8 if attempted).

Table 550. IC_SS_SCL_LCNT sample calculations

I²C data rate - SS (Kbps)	SCL clock frequency (MHz)	SCL low time required min (µs)	IC_SS_SCL_LCNT (hex/decimal)	SCL low timeactual (µs)
100	2	4.7	16‘h000A/’d10	5.00
100	6.6	4.7	16‘h0020/’d32	4.85
100	10	4.7	16‘h002F/’d47	4.70
100	75	4.7	16’h0161/’d353	4.71
100	100	4.7	16‘h01D6/’d470	4.70

Table 550. IC_SS_SCL_LCNT sample calculations (continued)

I²C data rate - SS (Kbps)	SCL clock frequency (MHz)	SCL low time required min (µs)	IC_SS_SCL_LCNT (hex/decimal)	SCL low timeactual (µs)
100	125	4.7	16'h024C/'d588	4.70
100	1000	4.7	16'h125C/'d4700	4.70

28.6.10 IC_FS_SCL_HCNT register(0x01C)

The IC_FS_SCL_HCNT is a 16 bit RW register which allows to set the high period of the SCL clock for fast-speed mode. The IC_FS_SCL_HCNT bit assignments are given in *Table 551*.

Note: 1 This register can be written only when the I²C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to 'b0. Write at other times has no effect.

2 This register must be set before any I²C bus transaction can take place in order to ensure proper I/O timing.

Table 551. IC_FS_SCL_HCNT register bit assignments

Bit	Name	Type	Reset value	Description
[15:00]	IC_FS_SCL_HCNT	RW	16'h64	SCL clock high period count for fast speed. This 16 bit field states the SCL clock high period count for fast speed. The minimum valid value is 6, and hardware prevents that a value less than this minimum will be written (setting 6 if attempted). It is used in high speed mode to send the master code and START BYTE or general CALL.

Table 552. IC_FS_SCL_HCNT sample calculations

I²C data rate - FS (Kbps)	SCL clock frequency (MHz)	SCL high time required min (µs)	IC_FS_SCL_HCNT (hex/decimal)	SCL high time actual (µs)
400	10	0.6	16'h0006/'d6	0.60
400	25	0.6	16'h000F/'d15	0.60
400	50	0.6	16'h001E/'d30	0.60
400	75	0.6	16'h002D/'d45	0.60
400	100	0.6	16'h003C/'d60	0.60
400	125	0.6	16'h004B/'d75	0.60
400	1000	0.6	16'h0258/'d600	0.60

28.6.11 IC_FS_SCL_LCNT register(0x020)

The IC_FS_SCL_LCNT is a 16 bit RW register which allows to set the low period of the SCL clock for fast-speed mode. The IC_FS_SCL_LCNT bit assignments are given in *Table 553*.

Note: 1 This register can be written only when the I^2C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to 'b0. Write at other times has no effect.

2 This register must be set before any I^2C bus transaction can take place in order to ensure proper I/O timing.

It is not necessary to configure this register if the I2C Controller is enabled as slave

Table 553. IC_FS_SCL_LCNT register bit assignments

Bit	Name	Type	Reset value	Description
[15:00]	IC_FS_SCL_LCNT	RW	16'h064	SCL clock low period count for fast speed. This 16 bit field states the SCL clock low period count for fast speed. The minimum valid value is 8, and hardware prevents that a value less than this minimum will be written (setting 8 if attempted). It is used in high speed mode to send the master code and START BYTE or general CALL.

Table 554. IC_FS_SCL_LCNT sample calculations

I^2C data rate - FS (Kbps)	SCL clock frequency (MHz)	SCL low time required min (µs)	IC_FS_SCL_LCNT (hex/decimal)	SCL low time actual (µs)
400	10	1.3	16'h000D/'d13	1.30
400	25	1.3	16'h0021/'d33	1.32
400	50	1.3	16'h0041/'d65	1.30
400	75	1.3	16'h0062/'d98	1.31
400	100	1.3	16'h0082/'d130	1.30
400	125	1.3	16'h00A3/'d163	1.30
400	1000	1.3	16'h0514/'d1300	1.30

28.6.12 IC_HS_SCL_HCNT register(0x024)

The IC_HS_SCL_HCNT is a 16 bit RW register which allows to set the high period of the SCL clock for high-speed mode. The IC_HS_SCL_HCNT bit assignments are given in *Table 555*.

Note: 1 This register can be written only when the I²C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to 'b0. Write at other times has no effect.

2 This register must be set before any I²C bus transaction can take place in order to ensure proper I/O timing.

It is not necessary to configure this register if the I2C Controller is enabled as slave.

Table 555. IC_HS_SCL_HCNT register bit assignments

Bit	Name	Type	Reset value	Description
[15:00]	IC_HS_SCL_HCNT	RW	16'h0 00a	SCL clock high period count for high speed. This 16 bit field states the SCL clock high period count for high speed. The minimum valid value is 6, and hardware prevents that a value less than this minimum will be written (setting 6 if attempted).

Table 556. IC_HS_SCL_HCNT sample calculations

I²C data rate - HS (Kbps)	SCL clock frequency (MHz)	SCL high time required min (µs)	I²C bus loading (pF)	IC_HS_SCL_HCNT (hex/decimal)	SCL high time actual (µs)
3400	100	60	100	16'h0006/'d6	60
3400	125	60	100	16'h0008/'d8	64
3400	1000	60	100	16'h003C/'d60	60
3400	100	120	400	16'h000C/'d12	120
3400	125	120	400	16'h000F/'d15	120
3400	1000	120	400	16'h0078/'d120	120

28.6.13 IC_HS_SCL_LCNT register(0x028)

The IC_HS_SCL_LCNT is a 16 bit RW register which allows to set the low period of the SCL clock for high-speed mode. The IC_HS_SCL_LCNT bit assignments are given in Table 557.

Note: 1 This register can be written only when the I²C controller is disabled, which corresponds to the IC_ENABLE (Section 28.6.21) register being set to 'b0. Write at other times has no effect.

2 This register must be set before any I²C bus transaction can take place in order to ensure proper I/O timing.

Table 557. IC_HS_SCL_LCNT register bit assignments

Bit	Name	Reset value	Description
[15:00]	IC_HS_SCL_LCNT	16'h001b	SCL clock low period count for high speed. This 16 bit field states the SCL clock low period count for high speed. The minimum valid value is 8, and hardware prevents that a value less than this minimum will be written (setting 8 if attempted).

Table 558. IC_HS_SCL_LCNT sample calculations

I^2C data rate - HS (Kbps)	SCL clock frequency (MHz)	SCL low time required min (µs)	I^2C bus loading (pF)	IC_HS_SCL_LCNT (hex/decimal)	SCL low time actual (µs)
3400	100	160	100	16‘h0010/’d16	160
3400	125	160	100	16‘h0014/’d20	160
3400	1000	160	100	16‘h00A0/’d160	160
3400	100	320	400	16‘h0020/’d32	320
3400	125	320	400	16‘h0028/’d40	320
3400	1000	320	400	16‘h0140/’d320	320

28.6.14 IC_INTR_STAT register(0x02C)

The IC_INTR_STAT is a RO register which indicates the interrupt status of the I^2C controller. As bit assignments show in *Table 559*, each bit in this register is associated to an interrupt source (*Section 28.5*), and if a bit is set it indicates that relevant interrupt has been issued. These bits are then cleared by reading the corresponding interrupt clear 1 bit register (*Section 28.6.20*).

Each bit has a corresponding mask bit in the IC_INTR_MASK register (*Section 28.6.15*). The raw version of these bits (prior to masking) is available in the IC_RAW_INTR_STAT register (*Section 28.6.16*).

Table 559. IC_INTR_STAT register bit assignments

Bit	Name	Type	Reset value	Description
[15:12]	Reserved		-	Read: undefined.

Table 559. IC_INTR_STAT register bit assignments (continued)

Bit	Name	Type	Reset value	Description
[11]	R_GEN_CALL	RO	1'h0	Refer to *Section 28.5* for a detailed description of these interrupt sources.
[10]	R_START_DET	RO	1'h0	
[09]	R_STOP_DET	RO	1'h0	
[08]	R_ACTIVITY	RO	1'h0	
[07]	R_RX_DONE	RO	1'h0	
[06]	R_TX_ABRT	RO	1'h0	
[05]	R_RD_REQ	RO	1'h0	
[04]	R_TX_EMPTY	RO	1'h0	
[03]	R_TX_OVER	RO	1'h0	
[02]	R_RX_FULL	RO	1'h0	
[01]	R_RX_OVER	RO	1'h0	
[00]	R_RX_UNDER	RO	1'h0	

28.6.15 IC_INTR_MASK register(0x030)

The IC_INTR_MASK is a RW register which allows to set the interrupt mask. As bit assignments show in *Table 560*, each bit in this register is associated to an interrupt source (*Section 28.5*), and if a bit is set it masks the relevant bit in the IC_INTR_STAT register (*Section 28.6.14*). They are active high, a value of 'b0 prevents a bit from generating an interrupt.

Table 560. IC_INTR_MASK register bit assignments

Bit	Name	Type	Reset value	Description
[15:12]	Reserved		-	Read: undefined. Write: should be zero.
[11]	M_GEN_CALL	RW	1'h1	Mask the corresponding bit in the IC_INTR_STAT register (see *Section 28.6.14*).
[10]	M_START_DET	RW	1'h0	
[09]	M_STOP_DET	RW	1'h0	
[08]	M_ACTIVITY	RW	1'h0	
[07]	M_RX_DONE	RW	1'h1	
[06]	M_TX_ABRT	RW	1'h1	
[05]	M_RD_REQ	RW	1'h1	
[04]	M_TX_EMPTY	RW	1'h1	
[03]	M_TX_OVER	RW	1'h1	
[02]	M_RX_FULL	RW	1'h1	
[01]	M_RX_OVER	RW	1'h1	
[00]	M_RX_UNDER	RW	1'h1	

Note: M_GEN_CALL bit should be set to 1 when IC_ACK_GENERAL_CALL register is set to 0.

28.6.16 IC_RAW_INTR_STAT register(0x034)

The IC_RAW_INTR_STAT is a RO register which indicates the raw interrupt status (prior to masking by IC_INTR_MASK register, *Section 28.6.15*) of the I^2C controller. As bit assignments show in *Table 561*, each bit in this register is associated to an interrupt source (*Section 28.5*), and if a bit is set it indicates that relevant interrupt has been issued – regardless of masking.

Note: Bit 9 and 10 are used only in debug mode.

There is no status bit for a RESTART condition because it is detected as a normal start condition. The I2C protocol does not care whether it is a START or RESTART because both conditions start from the IDLE state and send the message to all the slaves on the bus.

Table 561. IC_RAW_INTR_STAT register bit assignments

Bit	Name	Type	Reset value	Description
[15:12]	Reserved		-	Read: undefined.
[11]	GEN_CALL	RO	1'h0	Refer to *Section 28.5* for a detailed description of these interrupt sources.
[10]	START_DET	RO	1'h0	
[09]	STOP_DET	RO	1'h0	
[08]	ACTIVITY	RO	1'h0	
[07]	RX_DONE	RO	1'h0	
[06]	TX_ABRT	RO	1'h0	
[05]	RD_REQ	RO	1'h0	
[04]	TX_EMPTY	RO	1'h0	
[03]	TX_OVER	RO	1'h0	
[02]	RX_FULL	RO	1'h0	
[01]	RX_OVER	RO	1'h0	
[00]	RX_UNDER	RO	1'h0	

28.6.17 IC_RX_TL register(0x038)

The IC_RX_TL is a 8 bit RW register which controls the level of entries (or above) in the receive FIFO that triggers the RX_FULL interrupt. The IC_RX_TL bit assignments are given in *Table 562*.

Note: This register is automatically cleared by hardware when buffer level goes below the threshold.

Table 562. IC_RX_TL register bit assignments

Bit	Name	Type	Reset value	Description
[15:08]	Reserved		-	Read: undefined. Write: should be zero.
[07:00]	RX_TL	RW	8'h00	RX_FULL interrupt threshold. This 8 bit field value is the number of entries in the receive FIFO of the I^2C controller which defines the RX_FULL interrupt threshold, as (RX_TL + 1). The RX_TL valid range is 0 (8'h00) to 255 (8'hFF), resulting in threshold ranging from 1 to 256. Apart from numerical valid range, an additional restriction is that hardware does not allow the RX_TL value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer.

28.6.18 IC_TX_TL register(0x03C)

The IC_TX_TL is a 8 bit RW register which controls the level of entries (or below) in the transmit FIFO that triggers the TX_EMPTY interrupt. The IC_TX_TL bit assignments are given in *Table 563*.

Note: This register is automatically cleared by hardware when buffer level goes above the threshold.

Table 563. IC_TX_TL register bit assignments

Bit	Name	Reset value	Description
[15:08]	Reserved	-	Read: undefined. Write: should be zero.
[07:00]	TX_TL	8'h0	TX_EMPTY interrupt threshold. This 8 bit field value is the number of entries in the transmit FIFO of the I^2C controller which directly defines the TX_EMPTY interrupt threshold. The TX_TL valid range is 0 (8'h00) to 255 (8'hFF), resulting in threshold ranging from 0 to 255,), Apart from numerical valid range, an additional restriction is that hardware does not allow the TX_TL value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer.

28.6.19 IC_CLR_INTR register(0x040)

The IC_CLR_INTR is a RO register which allows to clear the combined interrupt, all individual interrupts and the TX_ABRT_SOURCE register (*Section 28.6.24*). To clear a specific interrupt, relevant clearing register has to be used (*Section 28.6.20*).

The IC_CLR_INTR bit assignments are given in *Table 564*.

Table 564. IC_CLR_INTR register bit assignments

Bit	Name	Type	Reset value	Description
[15:01]	Reserved		-	Read: undefined.
[00]	CLR_INTR	RO	1'h0	Reading this register causes interrupt to be cleared.

28.6.20 Interrupt clearing registers(0x044 - 0x068)

With the aim to clear an individual interrupt (among those supported by the I^2C controller, and listed in *Section 28.5*), a specific RO register must be read, according to below:

Table 565. Interrupt clearing registers

Register to be read	Relevant interrupt to be cleared (*Section 28.5*)
IC_CLR_RX_UNDER	RX_UNDER
IC_CLR_RX_OVER	RX_OVER
IC_CLR_TX_OVER	TX_OVER
IC_CLR_RD_REQ	RD_REQ
IC_CLR_TX_ABRT	TX_ABRT
IC_CLR_RX_DONE	RX_DONE
IC_CLR_ACTIVITY	ACTIVITY
IC_CLR_STOP_DET	STOP_DET
IC_CLR_START_DET	START_DET
IC_CLR_GEN_CALL	GEN_CALL

Note: RX_FULL and TX_EMPTY interrupts have no a specific clearing register, because they are automatically cleared by hardware when buffer level goes below/above the threshold, respectively.

28.6.21 IC_ENABLE register(0x06C)

The IC_ENABLE is a RW register which allow enabling/disabling the I^2C controller. The IC_ENABLE bit assignments are given in *Table 566*.

Table 566. IC_ENABLE register bit assignments

Bit	Name	Type	Reset value	Description
[15:01]	Reserved		-	Read: undefined. Write: should be zero.
[00]	ENABLE	RW	1'h0	I^2C controller enable. Setting this bit, the I^2C controller is enabled, otherwise (bit cleared) it is disabled. Software should not disable the I^2C controller while it is active. With this aim, the ACTIVITY bit in IC_STATUS register (*Section 28.6.22*) can be polled by software. When disabled, if the module was transmitting, the I^2C controller stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module was receiving, the I^2C controller stops the current transfer at the end of the current byte and does not acknowledge the transfer.

28.6.22 IC_STATUS register(0x070)

The IC_STATUS is a RO register which is used to indicate the current transfer status and the FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt. The IC_STATUS bit assignments are given in *Table 567*.

Table 567. IC_STATUS register bit assignments

Bit	Name	Reset value	Description
[15:07]	Reserved	-	Read: undefined.
[06]	SLV_ACTIVITY	1'h0	Slave FSM activity status. This bit reports the slave Finite State Machine (FSM) status, according to the encoding: 'b0, In IDLE state. = Not active. 'b1, Not in IDLE state. = Active.
[05]	MST_ACTIVITY	1'h0	Master FSM activity status. This bit reports the master FSM status, according to the encoding: 'b0, In IDLE state. = Not active. 'b1,Not in IDLE state. = Active. *Note: ACTIVITY field (bit[0]) in this register is the OR of SLV_ACTIVITY and MST_ACTIVITY bits.*
[04]	RFF	1'h0	Receive FIFO completely full. If set, this bit indicates that the receive FIFO is completely full. This bit is cleared when the receive FIFO contains one or more empty locations.

Table 567. IC_STATUS register bit assignments (continued)

Bit	Name	Reset value	Description
[03]	RFNE	1'h0	Receive FIFO not empty. If set, this bit indicates that the receive FIFO contains one or more entries. This bit is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO.
[02]	TFE	1'h1	Transmit FIFO completely empty. If set, this bit indicates that the transmit FIFO is completely empty. This bit is cleared when the FIFO contains one or more valid entries. This bit does not request an interrupt.
[01]	TFNF	1'h1	Transmit FIFO not full. If set, this bit indicates that the transmit FIFO contains one or more empty location (that is, it is not full). This bit is cleared when it is full.
[00]	ACTIVITY	1'h0	I^2C activity status.

28.6.23 IC_TXFLR and IC_RXFLR registers (0x074 - 0x078)

The IC_TXFLR (transmit FIFO level) and the IC_RXFLR (receive FIFO level) are RO registers which contain the number of valid entries in the transmit and in the receive FIFO buffer, respectively. The IC_TXFLR and IC_RXFLR bit assignments are given in *Table 568*.

These registers increment whenever data is placed into the transmit or receive FIFO, and decrement when data is taken from the transmit or receive FIFO. They are cleared when either the I^2C controller is disabled or whenever there is a transmission abort. Besides, the IC_TXFLR register is cleared also if the slave bulk transfer mode (*Section 28.4.1: Slave mode on page 612*) is aborted.

Table 568. IC_TXFLR and IC_RXFLR register bit assignments

Bit	Name	Type	Reset value	Description
[31:04]	Reserved		-	Read: undefined. Write: should be zero.
[03:00]	TXFLR/ RXFLR	RO	4'h0	Transmit (or receive) FIFO level.

28.6.24 IC_TX_ABRT_SOURCE register (0x080)

The IC_TX_ABRT_SOURCE (Transmit Abort Source) is a RW register which indicates the source of the transmission abort signal. This register is cleared whenever the processor reads it or when the processor issues a clear signal to all interrupts. The IC_TX_ABRT_SOURCE bit assignments are given in *Table 569*.

Table 569. IC_TX_ABRT_SOURCE register bit assignments

Bit	Name	Type	Reset value	Description
[31:16]	Reserved	RW	-	Read: undefined. Write: should be zero.
[15]	ABRT_SLVRD_INTX	RW	1'h0	Slave requesting data to transmit. If set, this bit indicates that the slave is requesting data to transmit and the user wrote a read command into the transmit FIFO.
[14]	ABRT_SLV_ARBLOST	RW	1'h0	Slave lost the bus. If set, this bit indicates that the slave lost the bus while it was transmitting data to a remote master. ARB_LOST bit in this register will be set at the same time.
[13]	ABRT_SLVFLUSH_TXFIFO	RW	1'h0	Slave receive a read command. If set, this bit indicates that the slave has received a read command and some data exists in the transmit FIFO, so the slave issues a TX_ABRT to flush old data in transmit FIFO.
[12]	ARB_LOST	RW	1'h0	Master lost arbitration. If set,this bit indicates that either master has lost arbitration (*Section 28.4.3*) or, if ABRT_SLV_ARBLOST bit in this register is also set, the slave transmitter has lost arbitration.
[11]	ARB_MASTER_DIS	RW	1'h0	Attempt to use disabled master. If set, this bit indicates that user attempt to use disabled master.
[10]	ABRT_10B_RD_NORSTRT	RW	1'h0	Disable restart and master send a read command. If set, this bit indicates that the restart is disabled (IC_RESTART_EN bit cleared in IC_CON register, *Section 28.6.3*) and the master sends a read command in 10 bit addressing mode.
[09]	ABRT_SBYTE_NORSTRT	RW	1'h0	Disable restart and user send a start byte. If set, this bit indicates that the restart is disabled (IC_RESTART_EN bit cleared in IC_CON register, *Section 28.6.3*) and the user is trying to send a start byte.
[08]	ABRT_HS_ NORSTRT	RW	1'h0	Disable restart and user try to use master to send data. If set, this bit indicates that the restart is disabled (IC_RESTART_EN bit cleared in IC_CON register, *Section 28.6.3*) and the user is trying to use the master to send data in high-speed mode.
[07]	ABRT_SBYTE_ACKDET	RW	1'h0	Master sent an acknowledge start byte. If set, this bit indicates that the master has sent a start byte which was acknowledged (wrong behavior).

Table 569. IC_TX_ABRT_SOURCE register bit assignments (continued)

Bit	Name	Type	Reset value	Description
[06]	ABRT_HS_ACKDET	RW	1'h0	Master in high speed mode. If set, this bit indicates that the master is in high-speed mode and the high-speed master code was acknowledge (wrong behavior).
[05]	ABRT_GCALL_READ	RW	1'h0	Master sent a general call. If set, this bit indicates that the master sent a general call (GCALL), but the user programmed the byte following the GCALL to be a read from the bus.
[04]	ABRT_GCALL_NOACK	RW	1'h0	Master sent a general call not acknowledge. If set, this bit indicates that the master sent a general call (GCALL) and no slave on the bus responded with an acknowledgement.
[03]	ABRT_TXDATA_ NOACK	RW	1'h0	Master receive acknowledge. If set, this bit indicates that the master has received an acknowledgement for the address but, when it sent data byte following the address, it did not receive an acknowledge from the remote slave.
[02]	ABRT_10ADDR2_ NOACK	RW	1'h0	Master in 10 bit addressing mode and 2nd address byte. If set, this bit indicates that the master is in 10 bit address mode and the 2nd address byte of the 10 bit address was not acknowledged by any slave.
[01]	ABRT_10ADDR1_ NOACK	RW	1'h0	Master in 10 bit addressing mode and 1st address byte. If set, this bit indicates that the master is in 10 bit address mode and the 1st address byte of the 10 bit address was not acknowledged by any slave.
[00]	ABRT_7B_ADDR_ NOACK	RW	1'h0	Master in 7 bit addressing mode. If set, this bit indicates that the master is in 7 bit address mode and the address sent was not acknowledged by any slave.

28.6.25 IC_DMA_CR register (0x088)

The IC_DMA_CR is a RW register which is used to enable the DMA controller interface operation.There is a separate bit for transmit and receive operation. The IC_DMA_CR bit assignments are given in *Table 570*

Note: This register can be programmed regardless of the state of IC_ENAABLE register.

Table 570. IC_DMA_CR register bit assignments

Bit	Name	Reset value	Description
[15:02]	Reserved		Read. undefined write: should be zero.
[01]	TDMAE	1'h0	Transmit DMA enable. Setting this bit, it enables the transmit FIFO DMA channel. Otherwise (bit cleared) it is disabled.
[00]	RDMAE	1'h0	Receive DMA enable. Setting this bit, it enables the receive FIFO DMA channel. Otherwise (bit cleared) it is disabled.

28.6.26 IC_DMA_TDLR register (0x08C)

The IC_DMA_TDLR (DMA transmit data level) is a RW register which allows controlling the DMA request as a function of transmit FIFO level. The IC_DMA_TDLR bit assignments are given in *Table 571*.

Table 571. IC_DMA_TDLR register bit assignments

Bit	Name	Reset value	Description
[15:03]	Reserved		Read: undefined. Write: should be zero.
[02:00	DMATDL	3'h0	Transmit data level. This filed controls the level at which a DMA request is made by the transmit logic. It means that a DMA request is generated when both the number of valid data entries in the transmit FIFO is equal to or below this field value, and the transmit FIFO channel is enabled (TDMAE is 'b1 in the IC_DMA_CR register)

28.6.27 IC_DMA_RDLR register (0x090)

The IC_DMA_RDLR (DMA receive data level) is a RW register which allows controlling the DMA request as a function of receive FIFO level. The IC_DMA_RDLR bit assignment are given in *Table 572*.

Table 572. IC_DMA_RDLR register bit assignments

Bit	Name	Reset value	Description
[15:03]	Reserved		Read: undefined. Write: should be zero.
[02:00]	DMARDL	3'h0	Receive data level. This bit field controls the level at which a DMA request is made by the receive logic. It means that a DMA request is generated when both the number of valid data entries in the receive FIFO is equal to or more than this field value+1, and the receive FIFO channel is enabled (RDMAE is 'b1 in the IC_DMA_CR register).

28.6.28 IC_COMP_PARAM1 register (0x0F4)

The IC_COMP_PARAM1 (component parameter register 1) is a RO register which contains encoded information about the component's parameter setting. The IC_COMP_PARAM1 bit assignments are given in *Table 573*.

Table 573. IC_COMP_PARAM register bit assignments

Bit	Name	Type	Reset value	Description
[31:24]	Reserved		-	Read: undefined.
[23:16]	TX_BUFFER_DEPTH	RO	8'h07	Transmission buffer depth. This 8 bit field reports the transmission buffer depth, according to the encoding: – 8'h00 = Reserved. – 8'h01 = 2 – 8'h02 = 3 … =... – 8'hFF = 256
[15:08]	RX_BUFFER_DEPTH	RO	8'h07	Receive buffer depth. This 8 bit field reports the receive buffer depth, according to the encoding: – 8'h00 = Reserved. – 8'h01 = 2 – 8'h02 = 3 … =... – 8'hFF = 256
[07]	ADD_ENCODED_PARAMS	RO	1'h1	Add encoded parameters. If set, this bit indicates that the encoded parameters can be read via software.
[06]	HAS_DMA	RO	1'h1	DMA interface. If set, this bit indicates that the I^2C controller provides for a set of DMA controller interface signals.
[05]	INTR_IO	RO	1'h1	Interrupt output port. This bit indicates whether all the interrupt sources are combined into a single output (INTR_IO set to 1'b1) or each interrupt source has its own output (INTR_IO set to 1'b0).
[04]	HC_COUNT_VALUES	RO	1'h0	Hard code count values. Controls the readability of the CNT register. If set, the CNT register is RO else the CNT register is RW.

Table 573. IC_COMP_PARAM register bit assignments (continued)

Bit	Name	Type	Reset value	Description
[03:02]	MAX_SPEED_MODE	RO	2'h3	Maximum speed mode. This 2 bit field indicates the maximum supported operation mode for the I^2C controller, according to the encoding: – 2'b00 = Reserved. – 2'b01 = Standard. – 2'b10 = Fast. – 2'b11 = High.
[01:00]	APB_DATA_WIDTH	RO	2'h1	Data width. This 2 bit field indicates the APB data bus width, according to the encoding: – 2'b00 = 8 bits – 2'b01 = 16 bits – 2'b10 = 32 bits – 2'b11 = Reserved

29 LS_Analog to digital convertor (ADC)

29.1 Overview

Within its Low Speed Subsystem, the device includes an Analog-to-Digital Converter (ADC), which is connected to the APB bus.

Main features of the ADC are listed below:

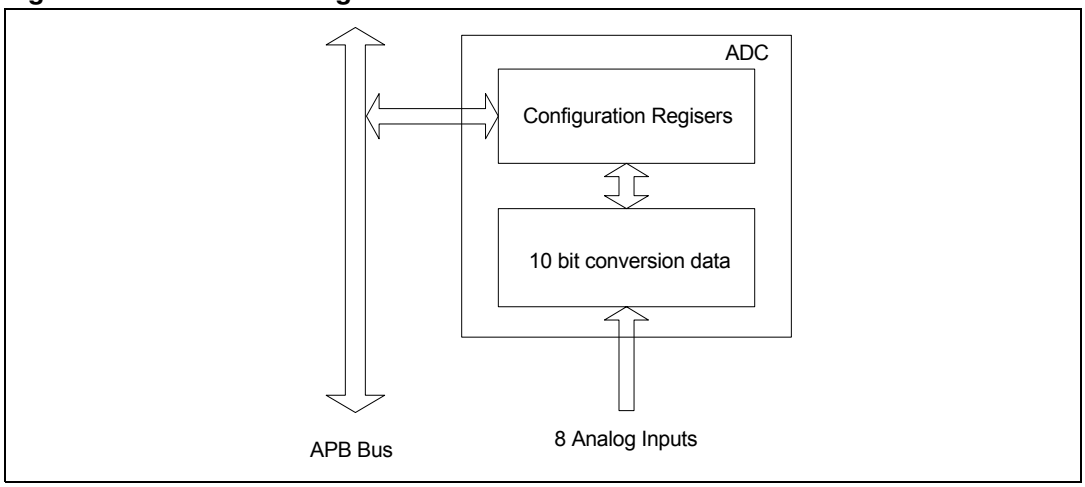
- Successive approximation conversion method.
- 10 bit resolution.
- 1 MSPS.
- 8 analog input (AIN) channels, ranging from 0 to 2.5V.
- For each input, the number of samples to be collected for average calculation can be 1 (no average) or up to 128 as 2's power (2, 4, 8...).
- INL ± 1 LSB, DNL ± 1 LSB.
- Programmable conversion speed, from a minimum conversion time of 1 µs.
- Normal or enhanced mode. In normal mode the conversion start upon CPU request while in enhanced mode the ADC converts continuously the selected channels inserting a selectable amount of time between two conversions.
- DC internal reference voltage is 1.95(Min), 2.0(Typ), 2.05(Max)

Positive and negative reference voltages can be supplied by dedicated pins to select different range (Default 0 - 2.5V).

29.2 Functional description

Figure 66 shows the functional block diagram of the ADC.

Figure 66. ADC block diagram



29.3 Operating sequence

29.3.1 Normal mode

As long as POWER DOWN bit in ADC_STATUS_REG register (*Section 29.5.1*) is set to logic'0', the ADC is inactive (disabled) and output latches contain last conversion.

Setting the POWER DOWN bit, the ADC enters in its functional mode after 50 us, when a conversion can be then initiated setting the ENABLE bit in ADC_STATUS_REG register (*Section 29.5.1*).

At first, the 10 bit Conversion Data field of the AVERAGE_REG register (*Section 29.5.2*) is reset to the value 10'b1000000000 and the acquisition from selected analog input channel occurs. After that, the conversion phase takes place, and 13 clock cycles are required for one complete conversion.

At the end of conversion, the CONVERSION READY bit in ADC_STATUS_REG is set, and the Conversion Data reading can begin. When the reading finishes, two different scenarios could occur:

- POWER DOWN bit is set (1'b1): the ENABLE bit is kept to 1'b1 (conversion enabled), and next conversion can takes place without waiting for the start up time (50 us).
- POWER DOWN bit is cleared (1'b0): the ADC is switched-off and next conversion requires again a start up time (after setting the ENABLE bit in the ADC_STATUS_REG register).

29.3.2 Enhanced mode

In this mode (ENM bit = 1) is possible to perform conversions on selected channels in a continuous way. The start of conversions may be external (EXT_SCAN_RATE bit = 1) or internal. In the first case you need of an external signal to start the conversions while in the internal mode is necessary to configure the SCAN_RATE register to set the number of APB Clock cycles between the start of two scan conversions.

To read conversion results you need to read CHx_DATA registers. CHx_DATA[17] is the VALID_DATA bit. It's logic '1' when read data is valid. It's logic '0' in following cases:

- ENM bit = 0;
- CHANNEL_EN of relative channel = 0;
- The controller is writing result in it.

On Channel 0 is possible to select a request to DMA (DMA_EN bit = 1) when conversion is finished. In this case at the end of conversion on this channel, the controller will perform a single request to DMA.When there will be next conversion on channel 0, the controller will check for the end of last DMA transfer continuing with a new conversion on this channel only if it is finished. In the other case the scan will continue with the other enabled channels.

29.4 Programming model

29.4.1 External pin connection

Table 574. External pin connection

Signal	Ball
VREF+	P14
VREF-	N14
AIN0	N16
AIN1	N15
AIN2	P17
AIN3	P16
AIN4	P15
AIN5	R17
AIN6	R16
AIN7	R15

The ADC can be fully configured by programming a set of 16 bit wide registers (listed in *Table 575*) which can be accessed at the base address 0xD008_0000.

Table 575. ADC registers summary

Register name	Offset	SIZE [bit]	Reset value	Type	Description
ADC_STATUS _REG	0x0000	16	16'h0	R/W	Status Register
AVERAGE_REG	0x0004	16	16'h0	RO	Report the data of requested conversion
SCAN_RATE	0x0008	32	32'h0	R/W	Scan rate for enhanced mode
ADC_CLK_REG	0x000C	16	16'h0	R/W	ADC Clock frequency selection.
CH0_CTRL	0x0010	4	4'h0	R/W	Channel 0 control register (enhanced mode)
CH1_CTRL	0x0014	4	4'h0	R/W	Channel 1 control register (enhanced mode)
CH2_CTRL	0x0018	4	4'h0	R/W	Channel 2 control register (enhanced mode)
CH3_CTRL	0x001C	4	4'h0	R/W	Channel 3 control register (enhanced mode)
CH4_CTRL	0x0020	4	4'h0	R/W	Channel 4 control register (enhanced mode)
CH5_CTRL	0x0024	4	4'h0	R/W	Channel 5 control register (enhanced mode)

Table 575. ADC registers summary (continued)

Register name	Offset	SIZE [bit]	Reset value	Type	Description
CH6_CTRL	0x0028	4	4'h0	R/W	Channel 6 control register (enhanced mode)
CH7_CTRL	0x002C	4	4'h0	R/W	Channel 7 control register (enhanced mode)
CH0_DATA	0x0030	11	11'h0	RO	Channel 0 Data register (Enhanced mode)
CH1_DATA	0x0034	11	11'h0	RO	Channel 1 Data register (Enhanced mode)
CH2_DATA	0x0038	11	11'h0	RO	Channel 2 Data register (Enhanced mode)
CH3_DATA	0x003C	11	11'h0	RO	Channel 3 Data register (Enhanced mode)
CH4_DATA	0x0040	11	11'h0	RO	Channel 4 Data register (Enhanced mode)
CH5_DATA	0x0044	11	11'h0	RO	Channel 5 Data register (Enhanced mode)
CH6_DATA	0x0048	11	11'h0	RO	Channel 6 Data register (Enhanced mode)
CH7_DATA	0x004C	11	11'h0	RO	Channel 7 Data register (Enhanced mode)

29.5 Register description

29.5.1 ADC_STATUS_REG register

The ADC_STATUS_REG is a RW register reporting the ADC status. This register can be written to only if both bit[8], CONVERSION READY, and bit[0], ENABLE, of the same register are set to 'b0.

Table 576. ADC_STATUS_REG register

Bit	Name	Reset value (1)	Type	Description
[15:14]	-	-	-	reserved
[13]	HIGH RESOLUTION	1'h0	RW	Enable HIGH Resolution (16 bit) output 1'b0 - Normal output (10 bit) 1'b1 - High Res. output (16 bit)

Table 576. ADC_STATUS_REG register (continued)

Bit	Name	Reset value (1)	Type	Description
[12]	DMA ENABLE	1'h0	RW	DMA Request Enable (For Channel 0 only) – 1'b0 - No DMA request – 1'b1 - DMA Request generated when the channel 0 conversion is done. *Note 29.5.2*
[11]	EXT_SCAN_RATE	1'h0	RW	Scan Rate type in enhanced mode. – 1'b0 - Scan rate is generated internally inside the ADC – 1'b1 - External Scan rate is generated outside (not implemented) *Note 29.5.2*
[10]	ENHANCED MODE	1'h0	RW	Operating mode: – 1'b0 - Normal mode – 1'b1 - Enhanced mode *Note 29.5.2*
[09]	VOLTAGE REFERENCE SELECT	1'h0	RW	Reference voltage source select: – 1'b0 - external (Pins ADC_VREFN and ADC_VREFP) – 1'b1 - Internal (range 0 - 2.5V) *Note: N - Negative Reference Voltage* *Note: P - Positive Reference Voltage*
[08]	CONVERSION READY	1'h0	RO	Conversion status. If set, this bit indicates that the requested conversion is completed and results are available. In contrast (bit cleared), the conversion is ongoing.
[07:05]	NUMBER OF AVERAGE SAMPLE	3'h0	RW	Number of samples to be averaged. This 3 bit field states the number of samples to be collect for average computation, according to encoding: – 3'b000 = No average, single data conversion. – 3'b001 = Average of 2 samples. – 3'b010 = Average of 4 samples. – 3'b011 = Average of 8 samples. – 3'b100 = Average of 16 samples. – 3'b101 = Average of 32 samples. – 3'b110 = Average of 64 samples. – 3'b111 = Average of 128 samples.
[04]	POWER DOWN	1'h0	RW	ADC enable. Setting this bit, the ADC is enabled, otherwise (bit cleared) the ADC is disabled.

Table 576. ADC_STATUS_REG register (continued)

Bit	Name	Reset value (1)	Type	Description
[03:01]	CHANNEL SELECT	3'h0	RW	Channel selection. This 3 bit field allows to select one of the 8 analog input (AIN) channels, according to encoding: – 3'b000 = AIN[0] – 3'b001 = AIN[1] – 3'b010 = AIN[2] – 3'b011 = AIN[3] – 3'b100 = AIN[4] – 3'b101 = AIN[5] – 3'b110 = AIN[6] – 3'b111 = AIN[7]
[00]	ENABLE	1'h0	RW	Conversion enable. Setting this bit, the conversion is enabled.

1. This bit can be set only when the ENABLE bit is reset.

29.5.2 AVERAGE_REG register

The AVERAGE_REG is a RO register which reports the resulting data of the requested ADC conversion, named as conversion data. It can return 10 or 17 bit according to the setting of HIGH RESOLUTION bit inside the register ADC_STATUS_REG (*Section 29.5.1*)

HIGH RESOLUTION = 0 (Normal mode) *Table 577*. The result of ADC conversion contains integer part only. The result is present in 10 bits **HIGH RESOLUTION =1** *Table 578*. The result of ADC conversion is present in 17 bit. The result contains both integer and fractional part of the ADC conversion.The number fractional part depends upon numbers of samples used.

Note: This register can be read only if both bit[8], CONVERSION READY, and bit[0], ENABLE, of the ADC_STATUS_REG register are set to 1'b1.

Table 577. Conversion data bits position in AVERAGE_REG (High Resolution = 0)

Bit	Name	Reset value	Type	Description
[09:00]	CONVERSION DATA	10'h0	RO	Contain the result of the ADC conversion

Table 578. Conversion data bits position in AVERAGE_REG (High Resolution = 1)

ADC_STATUS_REG[7:5]	Number of samples	Integer part of the result	Fractional part of the result
3'b000	1	bits [9:0]	-
3'b001	2	bits [10:1]	bit [0]
3'b010	4	bits [11:2]	bits [1:0]
3'b011	8	bits [12:3]	bits [2:0]

Table 578. Conversion data bits position in AVERAGE_REG (High Resolution = 1)

ADC_STATUS_REG[7:5]	Number of samples	Integer part of the result	Fractional part of the result
3‘b100	16	bits [13:4]	bits [3:0]
3’b101	32	bits [14:5]	bits [4:0]
3‘b110	64	bits [15:6]	bits [5:0]
3‘b111	128	bits [16:7]	bits [6:0]

HIGH RESOLUTION = 1

The AVERAGE_REG bit assignments allocate 16 bits for the conversion data.

29.5.3 SCAN RATE register

The SCAN RATE register is used only if the ADC is working in Enhanced mode. In this case it defines the number of APB Clock cycles that will be inserted in between the beginning of the current conversion and the start of the next one. The bit assignment are given in *Table 579*. Taking in account of the number of channels that are enabled and the value of this register is possible to calculate the real scan rate. To be noted that, is the value of the register do not cover one conversion time, the next conversion will start immediately when the previous one is ended.

Table 579. SCAN RATE register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	SCAN_RATE	32’h0000_0000	RW	Number of APB Clock cycles inserted in between two conversion start in enhanced mode.

29.5.4 ADC_CLK_REG register

The ADC_CLK_REG is a RW register which is used to program the frequency of ADC clock. The ADC_CLK_REG bit assignments are given in *Table 580*.

Note: This register can be written to only if both bit[8], CONVERSION READY, and bit[0], ENABLE, of the same register are set to 1‘b0.

Table 580. ADC_CLK_REG register bit assignments

Bit	Name	Reset value	Type	Description
[15:08]	-	2’h0	-	reserved
[07:04]	ADC_CLK_H	4’h0	RW	High state (as number of APB clock periods).
[03:00]	ADC_CLK_L	4’h0	RW	Low state (as number of APB clock periods).

The duty cycle of ADC clock results then from the ratio of the two values, while the frequency of ADC clock is the APB clock frequency divided by the sum of the same two values.

Because the frequency of ADC clock ranges from 3 MHz to 14 MHz, it follows that:

$$\left(\frac{\text{PB CLK Freq}}{14}\right) \leq (\text{ADC CLK H} + \text{ADC CLK L}) \leq \left(\frac{\text{APB CLK Freq}}{3}\right)$$

29.5.5 CHx CTRL register

The eight read/write Control registers are used only when enhanced mode is selected. They activate the particular channel during the scan and select the number of samples for the average. The CHx CTRL register bit assignments are given in *Table 581*

Note: This register can be written to only if the ENABLE bit is reset.

Table 581. CHx CTRL register bit assignments

Bit	Name	Reset value	Type	Description
[15:04]	-	-	-	reserved
[03:01]	AVERAGE	3'h0	RW	Number of samples to be averaged. This 3 bit field states the number of samples to be collect for average computation, according to encoding: – 3'b000 = No average, single data conversion. – 3'b001 = Average of 2 samples. – 3'b010 = Average of 4 samples. – 3'b011 = Average of 8 samples. – 3'b100 = Average of 16 samples. – 3'b101 = Average of 32 samples. – 3'b110 = Average of 64 samples. – 3'b111 = Average of 128 samples.
[00]	CHANNEL ENABLE	1'h0	RW	Enable the autoscan in enhanced mode: 1'b0 - No autoscan 1'b1 - autoscan

29.5.6 CHx DATA register

The eight read-only Data registers are used only when enhanced mode is selected. They contain the result of last conversion on relative channel. They have different bit assignments according to the setting of the bit 13 (HIGH RESOLUTION) on the register ADC_STATUS_REG (*Section 29.5.1*).

The register bit assignments when the bit is reset are given in the *Table 582*

The register bit assignments when the bit is set are given in the *Table 583*

Table 582. CHx DATA register (normal mode) bit assignments

Bit	Name	Reset value	Type	Description
[17]	VALID DATA-	1‘h0	RO	VALID DATA bit: 1’b0 - CONVERSION DATA field not valid 1’b1 - CONVERSION DATA field valid
[16:10]	RESERVED			reserved
[09:00]	CONVERSION DATA	10‘h0	RO	Contain the result of the last conversion

Table 583. CHx DATA register (HIGH RESOLUTION mode bit assignments

Bit	Name	Reset value	Type	Description
[17]	VALID DATA	1‘h0	RO	VALID DATA bit: 1’b0 - CONVERSION DATA field not valid 1’b1 - CONVERSION DATA field valid
[16:00]	CONVERSION DATA	16‘h0	RO	Contain the result of the last conversion

30 RS_Reconfigurable array subsystem (RAS) registers

30.1 RAS configurations

SPEAr300 can be programmed in various configurations. Out of these various configurations, RAS can be used with all its features available only in Dyn_cfg3_0 mode (*Table 158: SoC_CFG_CTR register bit assignments*).

Further in this mode, some RAS I/O ports (GPIO_0 - GPIO_50) are muxed with alternate function I/O. The user has been given the flexibility to use these pins either for RAS configuration function or the alternate function (See*Section 5.3*). This can be configured using RAS Configuration Register 1. Refer to *Table 30.4.1: RAS Register 1 (0x99000000)*

With only limited number of I/O ports available for RAS, all the features cannot be available simultaneously. To be able to use a specific set of functions of these IPs, the user has been given the option to use any one of the available 13 modes. Each of these modes has been provided keeping a particular set of applications in mind. These modes can be configured using RAS Configuration Register 2. Refer to*Table Note:: If the register bit is '1', then the particular IP in the fixed part will be connected to PL_GPIO. If the register bit is '0', then RAS_GPIO will be connected to PL_GPIO.*

The following modes can be selected by programming some control registers present in the reconfigurable array subsystem.

1. NAND Mode
2. NOR Mode
3. PHOTO_FRAME Mode (PHOTO FRAME)
4. LEND_IP_PHONE Mode (LOW END IP PHONE)
5. HEND_IP_PHONEMode (HIGH END IP PHONE)
6. LEND_WIFI_PHONE Mode (LOW END WI-FI PHONE)
7. HEND_WIFI_PHONE Mode (HIGH END WI-FI PHONE)
8. ATA_PABX_wI2S Mode (ATA PABX without I2S)
9. ATA_PABX_I2S Mode (ATA PABX with I2S)
10. CAMl_LCDw Mode (8-bit CAMERA without LCD)
11. CAMu_LCD Mode (14-bit CAMERA with LCD)
12. CAMu_wLCD Mode (14-bit CAMERA without LCD)
13. CAMl_LCD Mode (8 bit CAMERA with LCD)

Note: 1 RAS configurations are not supported with Dync_cfg1_0, Dyncfg1_1 and Dyn_cfg1_2 modes (Table 158: SoC_CFG_CTR register bit assignments).

2 In Dyn_cfg3_0,after reset chip is configured in nand mode and remains in this mode till application is configured through RAS register 2. In NAND Mode PLGPIO[36:4], PLGPIO[54:49],PLGPIO[80:59] are configured as output and drive logic '0' on these pads.

30.1.1 NAND mode

This mode is the default mode for SPEAr300. This mode supports the FSMC interface for NAND Flash connectivity along with some boot pins which gives information to the Boot

ROM about the way booting is to be done. The following sub-section explains the features supported by each IP present in this mode;

a) FSMC interface for NAND Flash connectivity (16 bits, 5 control signals)
b) Boot pins

30.1.2 NOR mode

This mode supports the FSMC interface for NOR Flash connectivity along with some boot pins which gives information to the Boot ROM about the way booting is to be done. The following sub-section explains the features supported by each IP present in this mode;

a) FSMC interface for NOR Flash connectivity
b) Boot Pins

30.1.3 Photoframe mode

In this mode, following IPs are present;

a) FSMC interface for NAND Flash connectivity (16 bits, 5 control signals)
b) CLCD controller interface
c) GPIO block supporting up to 8 general purpose IOs with interrupt capability
d) TDM block for voice/music capabilities
e) SD/SDIO/MMC host controller
f) Fully configurable Telecom GPIO8(7-4) and GPIO10(9-0)

30.1.4 LEND_IP_PHONE (LOW END IP PHONE mode)

In this mode, following IPs are present;

a) 9 x 9 Keyboard
b) 8 SPI/I2C Control signals
c) Digital-to-Analog converter (DAC)
d) I2S Block
e) TDM block capable of communicating with 8 external devices
f) SD/SDIO/MMC host controller
g) Fully configurable Telecom GPIO8(7-4) and GPIO10(9-0)
h) 8 Interrupt pins

30.1.5 HEND_IP_PHONE MODE (HIGH END IP PHONE)

In this mode, following IPs are present;

a) 9 x 9 Keyboard
b) LCD Controller Interface
c) 4 SPI/I2C Control signals
d) Digital-to-Analog converter (DAC)
e) I2S Block
f) TDM block capable of communicating with 2 external devices
g) SD/SDIO/MMC host controller
h) Fully configurable Telecom GPIO8(7-4) and GPIO10(9-0)

30.1.6 LEND_WIFI_PHONE MODE (LOW END WI-FI PHONE)

In this mode, following IPs are present;

a) 9 x 9 Keyboard
b) 8 Interrupt pins
c) 8 SPI/I2C Control signals
d) Digital-to-Analog converter
e) I2S Block
f) TDM block capable of communicating with 8 external devices
g) SD/SDIO/MMC host controller
h) Fully configurable Telecom GPIO8(7-4) and GPIO10(9-0)

30.1.7 HEND_WIFI_PHONE MODE (HIGH END WI-FI PHONE)

In this mode, following IPs are present;

a) 9 x 9 Keyboard
b) LCD Controller Interface
c) 4 SPI/I2C Control signals
d) Digital-to-Analog converter
e) I2S Block
f) TDM block capable of communicating with 2 external devices
g) SD/SDIO/MMC host controller
h) Fully configurable Telecom GPIO8(7-4) and GPIO10(9-0)

30.1.8 ATA_PABX_wI2S (ATA PABX without I2S)

In this mode, following IPs are present;

a) 7 Interrupt pins
b) 8 SPI/I2C Control signals
c) TDM block capable of communicating with 8 external devices
d) SD/SDIO/MMC host controller supporting at max 4 data lines
e) Fully configurable Telecom GPIO8(7-4) and GPIO10(9-0)

30.1.9 ATA_PABX_I2S MODE (ATA PABX with I2S)

In this mode, following IPs are present;

a) FSMC with 8 bit NOR/NAND Flash interface
b) 7 Interrupt pins
c) 8 SPI/I2C Control signals
d) Digital-to-Analog converter
e) I2S Block
f) TDM block capable of communicating with 4 external devices
g) SD/SDIO/MMC host controller supporting max. of 4 data lines
h) Fully configurable Telecom GPIO8(7-4) and GPIO10(9-0)

30.1.10 CAMI_LCDw MODE (8 bit CAMERA without LCD)

In this mode, following IPs are present;

a) 8 Bit Camera interface
b) 9 x 9 Keyboard
c) 4 SPI/I2C Control signals
d) Digital-to-Analog converter
e) I2S Block
f) TDM block capable of communicating with 8 external devices
g) SD/SDIO/MMC host controller
h) Fully configurable Telecom GPIO8(5-4) and GPIO10(9-4)

30.1.11 CAMu_LCD MODE (14 bit CAMERA with LCD)

In this mode, following IPs are present;

a) 14 Bit Camera interface
b) 7 x 5 Keyboard
c) LCD controller interface
d) Digital-to-Analog converter
e) I2S Block
f) TDM block capable of communicating with 2 external devices
g) SD/SDIO/MMC host controller
h) Fully configurable Telecom GPIO8(5-4) and GPIO10(9-4)

30.1.12 CAMu_wLCD MODE (14 bit CAMERA without LCD)

In this mode, following IPs are present;

a) 14 Bit Camera interface
b) 7 x 5 Keyboard
c) Digital-to-Analog converter
d) I2S Block
e) TDM block capable of communicating with 2 external devices
f) SD/SDIO/MMC host controller
g) Fully configurable Telecom GPIO8(5-4) and GPIO10(9-4)

30.1.13 CAMI_LCD MODE (8 bit CAMERA with LCD)

In this mode, following IPs are present;

a) 8 Bit Camera interface
b) LCD Controller interface
c) 9 x 9 Keyboard
d) 4 SPI/I2C Control signals
e) Digital-to-Analog converter
f) I2S Block
g) TDM block capable of communicating with 2 external devices
h) SD/SDIO/MMC host controller
i) Fully configurable Telecom GPIO8(5-4) and GPIO10(9-8)

30.2 Maximum number of GPIOs

The maximum number of GPIOs in each mode can be increased to the maximum indicated in *Table 12* by using other I/O pins if they not used for other purposes. For example, up to 62 general purpose I/Os are available in Mode 4 (LEND_IP_ph).

- In this mode the application can use:
  - 10 GPIOs in G10 block
  - 8 GPIOs in G8 block (0 to 3 in output mode only)
  - 18 GPIO (keyboard controller I/Os in GPIO mode)
  - 6 base GPIOs (basGPIO) (enabled as alternate functions (*Table 13*)
  - 8 IT pins (input only, with interrupt capability)
  - 4 SYNC outputs (SYNC4-7)
  - 8 SPI_I2C outputs

30.3 Address map

Table 584 gives the address map of the RAS IPs.

Table 584. RAS address space

Base Address	Address Space	IP
0x5000_0000	0x5000_0000 - 0x5FFF_FFFF	TELECOM
0x6000_0000	0x6000_0000 - 0x6FFF_FFFF	CLCD
0x7000_0000	0x7000_0000 - 0x7FFF_FFFF	SDIO
0x8000_0000	0x8000_0000 - 0x98FF_FFFF	FSMC
0x9900_0000	0x9900_0000 - 0x9FFF_FFFF	RAS Registers
0xA000_0000	0xA000_0000-0xAFFF_FFFF	APB Blocks

The address space for FSMC is further divided as follows:

Table 585. FSMC address space

Address Space	IP
0x8000_0000 - 0x83FF_FFFF	NAND on PCBank0
0x8400_0000 - 0x87FF_FFFF	NAND on PCBank1
0x8800_0000 - 0x8BFF_FFFF	NAND on PCBank2
0x8C00_0000 - 0x8FFF_FFFF	NAND on PCBank3
0x9000_0000 - 0x90FF_FFFF	NOR 0
0x9100_0000 - 0x91FF_FFFF	NOR 1
0x9200_0000 - 0x92FF_FFFF	NOR 2
0x9300_0000 - 0x93FF_FFFF	NOR 3
0x9400_0000 - 0x98FF_FFFF	FSMC Registers

The address space for APB blocks is allocated as follows:

Table 586. APB address space

Base Address	Address Space	IP
0xA0000000	0xA000_0000-0xA8FF_FFFF	Keyboard
0xA9000000	0xA900_0000-0xAFFF_FFFF	Independent GPIO block

30.4 RAS configuration registers

Two Registers 1 and 2 inside the RAS are used to configure different options available in RAS IPs. *Table 588* and *Table 589* explain in detail Register 1 and 2 respectively.

Table 587. RAS memory map

Start address	End address	Peripheral	Notes	Bus
0x4000_0000	0x4FFF.FFFF	Reserved		AHB
0x5000_0000	0x5000.FFFF	Telecom register		AHB

Table 587. RAS memory map

Start address	End address	Peripheral	Notes	Bus
0x5001_0000	0x5001_0FFF	TDM	Action memory	AHB
0x5003_0000	0x5003_7FFF	TDM	Buffer memory	AHB
0x5004_0000	0x5004_0FFF	TDM	Sync memory	AHB
0x5005_0000	0x5005_0FFF	I2S	I2S memory bank 1	AHB
0x5005_1000	0x5005_1FFF	I2S	I2S memory bank 2	AHB
0x6000_0000	0x6FFF_FFFF	CLCD		AHB
0x7000_0000	0x7FFF_FFFF	SDIO		AHB
0x8000_0000	0x83FF_FFFF	Static memory controller	NAND Bank0	AHB
0x8400_0000	0x87FF_FFFF	Static memory controller	NAND Bank1	AHB
0x8800_0000	0x8BFF_FFFF	Static memory controller	NAND Bank2	AHB
0x8C00_0000	0x8FFF_FFFF	Static memory controller	NAND Bank3	AHB
0x9000_0000	0x90FF_FFFF	Static memory controller	Parallel NOR Bank0	AHB
0x9100_0000	0x91FF_FFFF	Static memory controller	Parallel NOR Bank1	AHB
0x9200_0000	0x92FF_FFFF	Static memory controller	Parallel NOR Bank2	AHB
0x9300_0000	0x93FF_FFFF	Static memory controller	Parallel NOR Bank3	AHB
0x9400_0000	0x98FF_FFFF	Static memory controller	Configuration Register	AHB
0x9900_0000	0x9FFF_FFFF	Registers		AHB
0xA000_0000	0xA8FF_FFFF	Keyboard		APB
0xA900_0000	0xAFFF_FFFF	GPIO		APB
0xB000_0000	0xBFFF_FFFF	-	Reserved	

30.4.1 RAS Register 1 (0x99000000)

Due to limited number of GPIO pins available, they are shared between the IPs present in fixed part and those present in RAS. This register implements the sharing of the PL_GPIO pins between I/O alternate functions and the I/O functions of the RAS IPs.

If bit is '1', the I/Os alternate functions are connected to PL_GPIO pins.

If bit is '0', the IOs of corresponding RAS IP are connected to PL_GPIO pins.

Table 588. RAS Register 1 description

Field	Default Value	Description
[31:16]	16'h0	Reserved
[15]	1'h0	'0'- CLCD clock will be input to the RAS '1'- Clock from Pll2/ClkR_Synt(3) will be output on PL_CLK1
[14]	1'h0	FIRDA
[13	1'h0	I2C

Table 588. RAS Register 1 description

Field	Default Value	Description
[12]	1'h0	SSP enhanced (CS(2-3-4))
[11]	1'h0	SSP Basic (no CS(2-3-4))
[10]	1'h0	MAC Ethernet
[09]	1'h0	basGPIO (0)
[08]	1'h0	basGPIO(1)
[07]	1'h0	basGPIO(2)
[06]	1'h0	basGPIO(3)
[05]	1'h0	basGPIO(4)
[04]	1'h0	basGPIO(5)
[03]	1'h0	UART enhanced (also others UART signals, such as DCD....)
[02]	1'h0	UART basic (only RX & TX signals)
[01]	1'h0	Timer B (timers 3/4)
[00]	1'h0	Timer A (timers 1/2)

Note: If the register bit is '1', then the particular IP in the fixed part will be connected to PL_GPIO. If the register bit is '0', then RAS_GPIO will be connected to PL_GPIO.

30.4.2 RAS Register 2 (0x99000004)

This register is used to select between different modes of the RAS IPs.

Table 589. RAS Register 2 description

Field	Default Value	Description
[31]	1'h0	This bit is used for configuring the 48 MHz clock source for CLCD block. '1' - Clock from external source (PL_CLK1) '0' - 48 MHz clock
[30]	1'h0	This bit is used for configuring the ExtDevWidth input port of FSMC. '1' - ExtDevWidth = 2'b01 => 16 bit external bus '0' - ExtDevWidth = 2'b00 => 8 bit external bus
[29]	1'h0	This bit is used for configuring the accessibility of memory through either SDIO or I2S. '1' - SDIO is using the memory '0' - I2S is using the memory
[28]	1'h0	This bit is used for muxing the DAC O1, O2 ports and the GPIO10_3 and GPIO10_2 ports from the telecom block. This muxing is happening only when LEND_IP_PHONE or HEND_IP_PHONE or LEND_WIFI_PHONE or HEND_WIFI_PHONE is selected. '1' - GPIO10s from TELECOM '0' - DAC ports

Table 589. RAS Register 2 description (continued)

Field	Default Value	Description
[27:24]	4'h0	These bits are used for dynamic selection of NAND or NOR on each bank of FSMC. Each bit corresponds to a bank of FSMC. For e.g. Bit24 corresponds to Bank0 Bit25 corresponds to Bank1 and so on… If the bit is set i.e. '1' - NOR If the bit is clear i.e. '0' - NAND
[23]	1'h0	This bit is used to select between lower and higher numbered G8/G10 sets in modes where they are duplicated. In output mode, both sets of G8/G10 provide output data. In input mode, if this bit is 0, lower numbered G8/G10 set is active. If this bit is 1, higher numbered G8/G10 set is active.
[22:05]	18'h0	Reserved
[04]	1'h0	Determines if TDM_DOUT is available at Pin61 '1'- TDM_DOUT on Pin61 '0'- SPI_I2C7 on Pin61
[03:00]	4'h0	Selects different MODES as below: – 0000 NAND – 0001 NOR – 0010 PHOTO_FRAME – 0011 LEND_IP_PHONE – 0100 HEND_IP_PHONE – 0101 LEND_WIFI_PHONE – 0110 HEND_WIFI_PHONE – 0111 ATA_PABX_wI2S – 1000 ATA_PABX_I2S – 1100 CAM1_LCDw – 1101 CAMu_LCD – 1110 CAMu_LCDw – 1111 CAM1_LCD

30.5 RAS Interrupt assignments

Table 590. RAS Interrupt assignment

Generic RAS Interrupt	VIC	IPs
4	1	SDIO
3	30	CLCD
2	29	Reserved
1	28	TELECOM

30.6 RAS DMA configuration

Table 591. RAS DMA configuration

Request	RAS IP
RAS_DMA_REQ[0]	Not Used
RAS_DMA_REQ[1]	I2S
RAS_DMA_REQ[2]	TDM
RAS_DMA_REQ[3]	CAMERA
RAS_DMA_REQ[4]	Not Used
RAS_DMA_REQ[5]	Not Used
RAS_DMA_REQ[6]	Not Used
RAS_DMA_REQ[7]	Not Used
RAS_DMA_REQ[8]	Not Used
RAS_DMA_REQ[9]	Not Used
RAS_DMA_REQ[10]	Not Used
RAS_DMA_REQ[11]	Not Used
RAS_DMA_REQ[12]	Not Used
RAS_DMA_REQ[13	Not Used
RAS_DMA_REQ[14]	Not Used
RAS_DMA_REQ[15]	Not Used

Please refer to *Table 176* for DMA_CHN_CFG register assignments.

31 RS_Flexible static memory controller (FSMC)

31.1 Overview

Within its Reconfigurable Array Subsystem, SPEAr300 provides a Flexible Static Memory Controller (FSMC) which is intended to interface an AHB bus to external NAND/NOR Flash memories and to asynchronous SRAM memories.

The controller:

- Translates AHB protocol into the appropriate external storage device protocol,
- Meets the timing of the external devices, slowing down and counting an appropriate number of HCLK (AHB clock) cycles to complete the transaction to the external device.

Note: The external storage device cannot be faster than one AHB cycle.

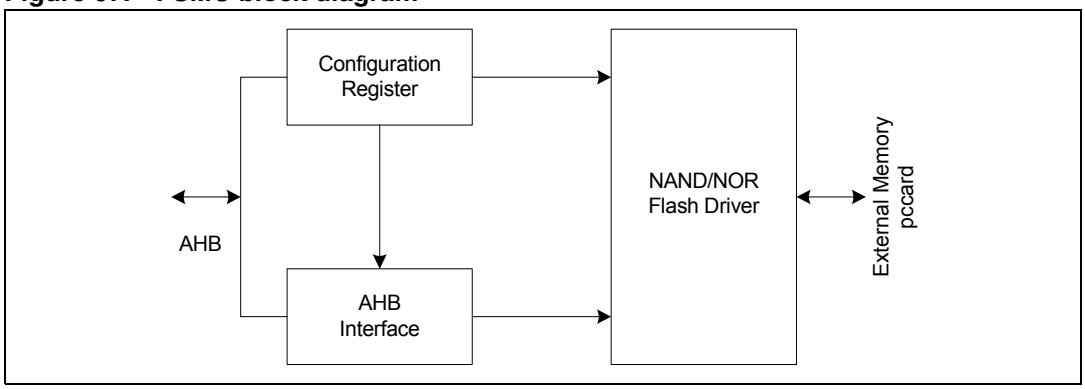
Main features of FSMC:

- AHB slave interface
- Interfaces static memory-mapped devices including RAM and Flash (both NAND and NOR).
- In case of Swims and Flash 8 and 16 bit wide, it provides external address and data paths;
- Performs only one access at a time;
- Supplies an independent configuration for each memory bank;
- Provides programmable timings to support a wide range of devices:
- programmable wait states (up to 31),
- programmable bus turn around cycles (up to 15),
- programmable output enable and write enable delays (up to 15).
- Provides independent chip select control for each memory bank;
- The address bus and the data bus are shared with all the external peripherals,
- Chip selects used to distinguish between each peripheral;
- Offers an external asynchronous wait control;
- Offers configurable size at reset for boot memory bank using RAS configuration registers.

31.2 Functional description

31.2.1 Block diagram

Figure 67. FSMC block diagram



31.3 Description

31.3.1 AHB interface

The *AHB Interface* block provides the FSMC interface to the AHB bus. It decomposes the system bus transfers into external accesses supported by the selected external device.

The RW control register values are accessed through the AHB, and their values are passed to the rest of the peripherals.

Following conditions cause an ERROR response:

- if a disabled external device is accessed;
- if a Flash memory is accessed when its Reset Powerdown bit (in GenMemCtrl register, *Section 31.4.6*) has been set to 0;
- if HSIZE is greater than 2, which means a transfer size larger than 32 bits.

In other cases, OKAY response is returned.

The AHB interface does not support the following AHB features:

- It does not generate SPLIT or RETRY responses;
- Protection control is not implemented, that is HPROT is not connected.

31.3.2 NAND flash controller

This block interfaces the AHB Interface block to the external NAND Flash.

For NAND Flash, following types of accesses are supported:

- **Common memory space access.** It is the normal way of accessing the NAND Flash. The data size is specified in the DevWidth field of configuration register (GenMemCtrl_PC register, *Table 599*), and corresponding timings must be specified in the GenMemCtrl_Comm register. The data used while accessing this region is passed to D(7:0) bits (or D(15:0) according to the Flash memory bus width). The few address bits A(17) and A(16), are used and drive directly ALE and CLE, respectively. Therefore

by accessing particular regions of the common memory space we can send to the NAND Flash a command (CLE high) or an address (ALE high).

- **Attribute memory space access.** The only difference with respect to previous common memory access mode is that the timings used are specified in the register GenMemCtrl_Attrib. In practice, this secondary access works exactly the same as common memory space accesses, but this can be used to implement a functionality that is needed in some (but not all) NAND Flash memories. Some Flash memories require that after writing the last part of the address, the controller has to wait and check that R/B (the wait signal from the Flash) goes low. This pre-wait time can be up to 100ns, then if there is such event the controller has to wait again until R/B goes high. During all this period, CE must be held low. Other NAND Flash memories (such as the Samsung 1 Gbit) do not require the pre-wait time; they just require that when the memory is being accessed and R/B is low, the controller should wait until R/B goes high.

31.3.3 Asynchronous SRAM and NOR parallel Flash controller

This block interfaces the AHB Interface block to external memory devices, such as SRAM and NOR Flash. All memories share the same signals (address, data and control) except chip select, that is unique for each controlled memory, with the following exceptions:

- SRAMs use also the byte lane (BL) to specify the byte to read or write, while NOR Flash has no such input;
- NOR Flash have address valid signal (ADV)

Note: Byte Lane and address valid signal have not been made available on SPEAr300 ports. So SRAMs which require BL inputs cannot be connected. Also, NOR Flash memories requiring ADV input cannot be connected.

- Address and control signals are automatically driven by this controller according to the type of memory in use.

Note: Synchronous memory access cannot be implemented in SPEAr300 due to non-availability of port to issue clock to the memory.

31.4 Programming model

31.4.1 External signals

Table 592. Parallel NOR flash

Signal	Direction	Description
DQ15:0	Bidir	Data lines.
A23:0	Out	Address lines.
/W	Out	Write Enable Active Low.
/R	Out	Read Enable Active Low.
/E1, /E2, /E3 and /E4	Out	Chip Enable Active Low.
R/B	In	Wait Signal Active Low.

Table 593. NAND Flash

Signal	Direction	Description
D[15:0]	Bidir	Data lines.
CL	Out	Command Latch Enable Active High.
AL	Out	Address Latch Enable Active High.
/W	Out	Write Enable Active Low.
/R	Out	Read Enable Active Low.
/E1, /E2, /E3 and /E4	Out	Chip Enable Active Low.
R/B	In	Wait Signal Active Low.

Note: Multiple chip enable pins allow to connect up to four different memory devices simultaneously. Address and data buses are shared. Controller performs only one access at a time.

31.4.2 Address map

The address space for FSMC is further divided as follows:

Table 594. FSMC Address Map

From	To	Description
0x8000_0000	0x83FF_FFFF	NAND on Bank0
0x8400_0000	0x87FF_FFFF	NAND on Bank1
0x8800_0000	0x8BFF_FFFF	NAND on Bank2
0x8C00_0000	0x8FFF_FFFF	NAND on Bank3
0x9000_0000	0x90FF_FFFF	NOR/SRAM on Bank0
0x9100_0000	0x91FF_FFFF	NOR/SRAM on Bank1
0x9200_0000	0x92FF_FFFF	NOR/SRAM on Bank2
0x9300_0000	0x93FF_FFFF	NOR/SRAM on Bank3
0x9400_0000	0x98FF_FFFF	FSMC Configuration Registers

Note: CL and AL signals for NAND Flash correspond to A16 and A17, respectively on FSMC. So, if we write at an address within the address range of a particular NAND bank such that the address bit 16/17 is '1', the CL/AL pin at output will go high. Data on data lines is thus interpreted as command and address respectively.

31.4.3 Register map

The FSMC can be fully configured by programming its 32-bit wide registers which can be accessed at the base address 0x9400_0000.

Independent configuration registers are associated with each chip select, and they allow to specify the type of external device (SRAM or Flash) as well as the associated timings (i.e., how many HCLK cycles to complete a transaction), and other external device characteristics so that FSMC can use the correct protocol.

The FSMC registers are usually initialized at boot time, however it is possible to change them at any moment.

FSMC registers can be logically arranged in two main groups:

- **Control and timing registers** (listed in *Table 595*): for FSMC configuration
- **Identification registers** (listed in *Table 596*): eight 8-bit RO registers reporting FSMC-specific information.

Note: In addition to reserved locations within the control and timing registers address space (Table 595), offset addresses from 'h0C0 to 'hFDC are reserved for test purposes. All these locations must not be used during normal operation.

Table 595. FSMC control and timing registers summary

Name	Offset	Type	Description
GenMemCtrl0	0x000	RW	Controls of bank 0.
GenMemCtrl_tim0	0x004	RW	Timings of bank 0.
GenMemCtrl1	0x008	RW	Controls of bank 1.
GenMemCtrl_tim1	0x00C	RW	Timings of bank 1.
GenMemCtrl2	0x010	RW	Controls of bank 2.
GenMemCtrl_tim2	0x014	RW	Timings of bank 2.
GenMemCtrl3	0x018	RW	Controls of bank 3.
GenMemCtrl_tim3	0x01C	RW	Timings of bank 3.
	0x020 to 0x03C	-	Reserved.
GenMemCtrl_PC0	0x040	RW	Controls of NAND 0.
	0x044	-	Reserved.
GenMemCtrl_Comm0	0x048	RW	Timings of NAND 0 in common memory mode
GenMemCtrl_Attrib0	0x04C	RW	Timings of NAND 0 in attribute memory mode
	0x050	-	Reserved.
GenMemCtrl_ECCr0	0x054	RO	NAND-Flash 0 ECC Result.
	0x058 to 0x05C	-	Reserved
GenMemCtrl_PC1	0x060	RW	Controls of NAND 1
	0x064	-	-
GenMemCtrl_Comm1	0x068	RW	Timings of NAND 1 in common memory mode.
GenMemCtrl_Attrib1	0x070	-	-
	0x070	-	-
GenMemCtrl_ECCr1	0x074	RO	NAND-Flash 1 ECC Result.
	0x078 to 0x07C	-	Reserved.
GenMemCtrl_PC2	0x080	RW	Controls of NAND 2
	0x084	-	-

Table 595. FSMC control and timing registers summary (continued)

Name	Offset	Type	Description
GenMemCtrl_Comm2	0x088	RW	Timings of NAND 2 in common memory mode.
GenMemCtrl_Attrib2	0x08C	RW	Timings of NAND 2in attribute memory mode.
	0x090 to 0x09C	-	Reserved.
GenMemCtrl_PC3	0x0A0	RW	Controls of NAND 3
	0x0A4	RW	-
GenMemCtrl_Comm3	0x0A8	RW	Timings of NAND 3 in common memory mode
GenMemCtrl_Attrib3	0x0AC	RW	Timings of NAND 3 in attribute memory mode
	0x0B0 to 0x0BC	-	Reserved

Table 596. FSMC identification registers summary

Name	Offset	Size	Type	Value	Description
GenMemCtrl_PeriphID0	0xFE0	8	RO	8'h90	Peripheral Identification
GenMemCtrl_PeriphID1	0xFE4	8	RO	8'h00	
GenMemCtrl_PeriphID2	0xFE8	8	RO	8'h08	
GenMemCtrl_PeriphID3	0xFEC	8	RO	8'h00	
GenMemCtrl_PCellID0	0xFF0	8	RO	8'h0D	IPCell Identification
GenMemCtrl_PCellID1	0xFF4	8	RO	8'hF0	
GenMemCtrl_PCellID2	0xFF8	8	RO	8'h05	
GenMemCtrl_PCellID3	0xFFC	8	RO	8'hB1	

31.4.4 Register description

31.4.5 GenMemCtrl(i) registers

Each GenMemCtrl(i) (with i = 0...3) is a RW register which contain the control information of i-th bank used for SRAMs and NOR Flash memories. The bit assignments of GenMemCtrl(i) are given in *Table 597*.

Table 597. GenMemCtrl(i) register bit assignments

Bit	Name	Reset value	Description
[31:14]	-	-	Reserved. Read: undefined. Write: should be zero.
[13]	Waiten	1'h1	Enable wait check. This bit enables wait check from memories that have wait signal, according to the encoding below: 0 - Disabled. 1 - Enabled (default).

Table 597. GenMemCtrl(i) register bit assignments (continued)

Bit	Name	Reset value	Description
[12]	If_we	1'h1	Interface write enable. This bit allows to enable write in NOR Flash and SRAM, according to the encoding below: 0 - Disabled. 1 - Enabled (default).
[11]	Wait_delay	1'b0	Wait timing. This bit manages the wait signal from NOR Flash, according to the encoding below: 0 - Wait is anticipated. 1 - Wait has the same data timing (delayed).
[10]	-	-	Reserved. Read: undefined. Write: should be zero.
[09]	WaitPolarity	1'h0	Polarity of wait signal. This bit specifies the polarity of the wait signal, according to encoding below: 0 - Active low (default). 1 - Active high
[08]	-	-	Reserved. Read: undefined. Write: should be zero.
[07]	Wprot	1'h1	Write protect signal to Flash memory. This field is applicable only to Flash memories. It directly represents the Wprot signal to the input pin of the Flash memory, according to the encoding below: 0 - Disabled. 1 - Enabled (default). Note: Wprot port is not available in SPEAr300.
[06]	RstPwr-Down	1'h1	Reset/power down signal to Flash memory. This field is applicable only to Flash memories. It directly represents the RstPwdwn signal to the input pin of the Flash memory. Note: Wprot port is not available in SPEAr300.
[05:04]	DevWidth	-	Device width. This 2-bit field is applicable to all types of memories, and it indicates the memory data size, according to the encoding below: – 00 - 8 bit – 01 - 16 bit – 10 - 32 bit (not used in SPEAr300) – 11 - Not used. Default Value is set using bit 30 of the RAS configuration register. Register_2, according to the encoding: 0-00 & 1-01
[03:02]	MemType	-	Memory type. This 2bit-field indicates the memory type, according to the encoding below: – 00 - SRAM – 01 - Reserved – 10 - NOR Flash. – 11 - Reserved Bank 0 has ‘b10 as default value, whereas all other banks have ‘b00

Table 597. GenMemCtrl(i) register bit assignments (continued)

Bit	Name	Reset value	Description
[01]	Muxed	1'h1	Muxed memory. This bit allows to enable muxed memory to use the data bus to get the addresses, according to the encoding below: 0 - Not muxed. 1 - Muxed (Not used in SPEAr300)
[00]	BankEnable	-	Enable bank. This bit allows to enable/disable the relevant bank, according to the encoding below: 0 - Disabled. 1 - Enabled. After reset Bank 0 is enabled and all the others are disabled. Accessing a disabled bank causes a HRESP to be ERROR on AHB. A NOR Flash memory that is in reset-power down (RstPowerDown = 'b0), is considered disabled, even if the BankEnable bit is 'b1. It follows that accessing the memory in such condition causes a HRESP to be ERROR on AHB.

31.4.6 GenMemCtrl_tim(i) registers

Each GenMemCtrl_tim(i) (with i = 0...3) is a RW register which contains the timing control information of each bank used for SRAMs and NOR Flash memories. The GenMemCtrl_tim(i) bit assignments are given in *Table 598*.

Table 598. GenMemCtrl_tim(i) register bit assignments

Bit	Name	Reset value	Description
[31:20]	-	-	Read: undefined. Write: should be zero.
[19:16]	BusTurn	4'hF	Bus turn around duration. This 4-bit field indicates the bus turn around duration (HCLK cycles + 1). Each time the FSMC ends a memory read, a single access or the last of a burst, it enters the bus turn around state, and it starts to count the HCLK cycles specified in this field.
[15:08]	Data_st	8'hFF	Duration of Data_st phase. This 8-bit field indicates the duration of "Data_st" phase (HCLK cycles + 1). This value cannot be 8'h0, its minimum value must be 8'h01. This field is applicable to all memory types.
[07:04]	Hold_addr	4'hF	Duration of Hold_addr phase. This 4-bit field indicates the duration of "Hold_addr" phase (HCLK cycles + 1). This field is applicable to SRAMs.
[03:00]	AddrST	4'hF	Duration of address state phase. This 4-bit field indicates the duration of “Address state” phase, expressed as (HCLK cycles + 1). This field is applicable to NOR Flash.

Note: For information on programming the timing register, refer Section 31.4.12: Calculating the FSMC timing parameters on page 673.

31.4.7 GenMemCtrl_PC(i) registers

Each GenMemCtrl_PC(i) (with i = 0...3) is a RW control registers used for NAND Flash. The GenMemCtrl_PC(i) bit assignments are given in *Table 599*.

Table 599. GenMemCtrl_PC(i) register bit assignments

Bit	Name	Reset value	Description
[31:17]	-	-	Reserved. Read: undefined. Write: should be zero.
[16:13]	tar	4'h0	ALE to REb delay. Used for NAND Flash, this 4-bit field indicates the time from ALE low to REb low, Tar, as an integer number of HCLK cycles according to following formula: Tar = HCLK period* (tar +1). The minimum value for this field is 4'b0000 (default), that is Tar is one HCLK cycle.
[12:09]	tclr	4'h0	CLE to REb delay. Used for NAND Flash, this 4-bit field indicates the time from CLE low to REb low, Tclr, as an integer number of HCLK cycles according to following formula: Tclr = HCLK period * (tclr +1). The minimum value for this field is 4'b0000 (default), that is Tclr is one HCLK cycle.
[08]	-	1'h0	Reserved. Read: undefined. Write: should be zero.
[07]	EccPLen	1'h0	ECC page length. This bit allows to define the page length of the NAND Flash memory device for configuring the ECC computation logic, according to the encoding below: 0 - 512 bytes (default) 1 - 256 bytes
[06]	Eccen	1'h0	ECC computation logic enable bit. This bit allows to enable the ECC computation logic, according to the encoding below: 0 - Disabled and reset (default) 1 - Enabled
[05:04]	DevWidth	-	Data width. This 2-bit field indicates the data width, according to the encoding below: – 00 - 8 – 01 - 16 – 10 - 32 (Not used in SPEAr300) – 11 - Not used. This field is valid only if Dev_type (see 'bit 3' below) is NAND Flash. Default Value is set using bit 30 of the RAS configuration, Register_2, according to the encoding: 0 - 00 1 - 01
[03]	DevType	1'h1	Type of device. This bit indicates the type of device, according to the encoding below: 0 - Not used 1 - NAND Flash (default).
[02]	Enable	1'h0	Enable NAND Active High
[01]	Wait_on	1'h0	Activates the wait feature for the NAND Active High
[00]	Reset	1'h0	Software reset for NAND Reset level = 1

31.4.8 GenMemCtrl_Comm(i)/GenMemCtrl_Attrib(i) registers

Each GenMemCtrl_Comm(i)/GenMemCtrl_Attrib(i) (with i = 0...3) is a RW register which contain the timing control information of each bank used for NAND Flash memories. The GenMemCtrl_Comm(i)/GenMemCtrl_Attrib(i) bit assignments are given in *Table 600*.

Table 600. GenMemCtrl_Comm(i)/GenMemCtrl_Attrib(i) register bit assignments

Bit	Name	Reset value	Description
[31:24]	Thiz	8'hFC	Time from address valid to data bus driven. (Write cycle only) The total time is: thiz = Tclk * Thiz. Min value for Thiz is 0.
[23:16]	Thold	8'hFC	Time from enable off to when address/data goes to high impedance. (Read and write cycle). The total time is: thold = Tclk * Thold. T period = tset + twait+ thold Min value for Thold is 1.
[15:08]	Twait	8'hFC	Time from enable on to enable off for all signals (Read and write cycle) The total time is: twait = Tclk * (Twait + 1). Min value for Twait is 1.
[07:00]	Tset	8'hFC	Time from address valid to RE/WE activation. (Read and write cycle) The total time is: tset = Tclk * (Tset + 1). Min value for Tset is 0.

Note: For information on programming the timing registers refer to Section 31.4.12: Calculating the FSMC timing parameters on page 673

31.4.9 GenMemCtrl_ECCr(i) registers

Each GenMemCtrl_ECCr(i) (with i = 0, 1) is a 32-bit RO register which contains the ECC (Error Correction Code) computation result for the corresponding NAND flash memory. The GenMemCtrl_ECCr(i) bit assignments are given in *Table 601*.

The ECC is actually a Hamming-based code which is used to preserve the consistency of data stored in NAND flash memories. The ECC algorithm consists in calculating the row and column parity of a page of memory and to place the 3-byte result in an ECC table, where it can be retrieved in order to check the consistency of the data. The GenMemCtrl_ECCr(i) reports this ECC 3-byte result.

Table 601. GenMemCtrlECCr(i) register bit assignments

Bit	Name	Reset value	Description
[31:24]	-	-	Reserved. Read undefined. Write: should be zero.
[23:16]	ECC3	8'hFF	MSB part of ECC
[15:08]	ECC2	8'hFF	ECC
[07:00]	ECC1	8'hFF	LSB part of ECC

31.4.10 GenMemCtrl peripheral identification registers (GenMemCtrlPeripID0-3)

The GenMemCtrlPeriphID0-3 registers are four read only 8-bit registers. The bit assignments are:-

Table 602. GenMemCtrlPeriphID0 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:00]	PartNumber0	8'h90	These bits read back as 0x90

Table 603. GenMemCtrlPeriphID1 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:04]	Designer0	4'h00	These bits read back as 0x0
[03:00]	PartNumber1	4‘h00	These bits read back as 0x0

Table 604. GenMemCtrlPeriphID2 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:04]	Revision	4'h00	These bits return the peripheral revision.
[03:00]	Designer1	4‘h08	These bits read back as 0x8

Table 605. GenMemCtrlPeriphID3 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:00]	Configuration	8'h00	These bits read back as 0x00

31.4.11 GenMemCtrl cell Identification registers (GenMemCtrlPCellID0-3)

The GenMemCtrlPCellID0-3 registers are four read only 8-bit registers. The bit assignments are -

Table 606. GenMemCtrlPeriphID3 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:00]	GenMemCtrlPCellID0	8'h0D	These bits read back as 0x0D

Table 607. GenMemCtrlPeriphID1 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:00]	GenMemCtrlPCellID1	8'hF0	These bits read back as 0xF0

Table 608. GenMemCtrlPCellID2 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:00]	GenMemCtrlPCellID2	8'h05	These bits read back as 0x05

Table 609. GenMemCtrlPCellID3 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved. Read undefined. Write: should be zero.
[07:00]	GenMemCtrlPCellID3	8'hB1	These bits read back as 0xB1

31.4.12 Calculating the FSMC timing parameters

NAND Flash Interface

NAND interface has 4 timing registers-

- tset = timing from address, CSx, reg set to oen, wen.
- twait= timing for eon, wen to last.
- thold = timing from oen, wen (high) to end of cycle.
- thiz = timing from start of cycle and enable data out bus (only for write mode).

The delays are:

tset = (Tset_value + 1) * clock period

twait = (Twait_value + 1) * clock period

thold = (Thold_value) * clock period

thiz = (Thiz_value) * clock period

So timing registers must be programmed with this value:

Tset =(timing for tset/clock period) -1

Twait =(timing for twait/clock period) -1

Thiz=(timing for thiz/clock period)

About Thold_value the right value is:

(Timing period - tset - twait)/Tclk.

FSMC parameters are:

TCS = Clock cycles between Chip Selects = 3

TCLK = Hclk period.

TWAIT = number of TCLK to get wait signal active = 3

Chip delays:

toutdel = Output delay from the flip-flops to the board.

tindel = Input delay from the board to the flipflop

Dtoutdel = maximum difference between output delay: toutdel.

Some suggested values:

toutdel = 7

tindel = 5

Dtoutdel = 5

If you want to have some margin on it:

toutdel = 15

tindel =10

Dtoutdel = 10

Tset computation

Choose the biggest among:

tcs,

tcls - TCS*TCLK

tar - TCS*TCLK

tals

tclr

twh -TCS*TCLK

treh - (TCS+Thold)*TCLK

tir + tdh -(TCS+Thold)*TCLK

======> tset

Tset = int((tset + Dtoutdel)/TCLK)

Twait computation

Chose the biggest among:

twp,

trp.

trea

tcea-(Tset+1)*TCLK + toutdel + tindel (for data availability)

======> twait.

Twait = int(twait / TCLK)

Thold computation

No wait signal expected (common memory):
Chose the biggest among:
talh,
tch,
twc - (Tset + 1 + Twait + 1) * TCLK
trc - (Tset + 1 + Twait + 1) * TCLK
twhr - (Tset+1+TCS) * TCLK
tclh
======> thold
Thold = int(thold + trr + Dtoutdel/TCLK) +1

Wait signal expected (attribute memory):
Chose the biggest among:
talh,
tch,
twc - (Tset + 1 + Twait + 1) * TCLK
trc - (Tset + 1 + Twait + 1) * TCLK
twhr - (Tset+1+TCS) * TCLK
twb
tclh
======> thold
Thold* =int((thold + trr + toutdel + tindel)/TCLK) + TWAIT +1

Thiz computation

The biggest among:
tchz
trhz - Thold * TCLK
tdh - Thold * TCLK
======> thiz
Thiz =int((thiz + toutdel + tindel)/TCLK+1) -TCS

NOR Flash Interface

taddr_st=max between:

tavlh, twhwl.

=====> taddr_st

Taddr_st = int((taddr_st+Dtoutdel)/TCLK)

Thold_add = int((tlhax+Dtoutdel)/TCLK)

Tdata_st = int((tllqv + tdelout + tdelin)/TCLK) - Taddr_st -1

tBusTurn is the max between:

tehqz

tghqz.

TBusTurn=int((tBusTurn + tindel + toutdel)/TCLK)

The minimum value for TBusTurn is 3.

32 RS_SDIO controller

32.1 Overview

Within the Reconfigurable Array Subsystem, SPEAr300 can provides an SDIO host controller that has an AMBA compatible interface and conforms to the SD host Controller Standard Specification Version 2.0. It handles SDIO/SD Protocol at transmission level, packing data, adding cyclic redundancy check (CRC), start/end bit and checking for transaction format correctness. The host controller provides programmed IO and DMA data transfer method.

32.2 Main features

- Meets SD Host Controller Standard Specification Version 2.0
- Meets SDIO card specification version 2.0
- Meets SD Memory Card Specification Draft version 2.0
- Meets SD Memory Card Security Specification version 1.01
- Meets MMC Specification version 3.31 and 4.2
- Supports both DMA and Non-DMA mode of operation
- Supports MMC Plus and MMC Mobile
- Card Detection (Insertion / Removal)
- Password protection of Cards
- Host clock rate variable between 0 and 48 MHz
- Supports 1 bit, 4 bit and 8 bit SD modes and SPI mode
- Supports Multi Media Card Interrupt mode
- Allows card to interrupt host in 1bit, 4 bit, 8 bit SD modes and SPI mode.
- Upto 100Mbits per second data rate using 4 parallel data lines (SD4 bit mode)
- Upto 416Mbits per second data rate using 8 bit parallel data lines (SD8 bit mode)
- Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
- Designed to work with I/O cards, Read-only cards and Read/Write cards
- Error Correction Code (ECC) support for MMC4.2 cards
- Supports Read wait Control, Suspend/Resume operation
- Supports FIFO Overrun and Underrun condition by stopping SD clock
- Conforms to AMBA specification AHB (2.0)

32.3 Signal interface

Figure 68. SDIO controller pin diagram

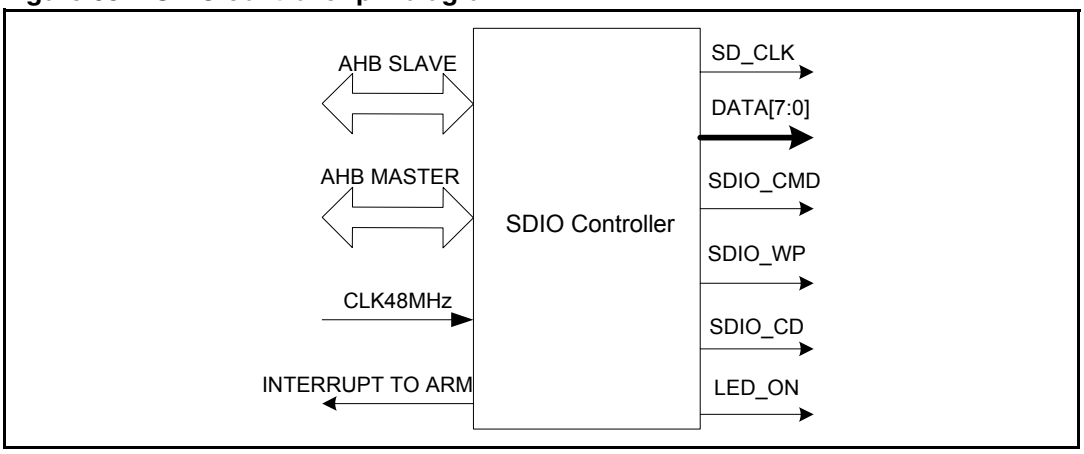


Table 610. Signal interface

Group	Signal name	Direction	Size	Description
GLOBAL	HCLK	In	1	AHB system clock
	HRESETn	In	1	AHB system reset
AHB SLAVE		In/Out		See AMBA Specification
AHB MASTER		In/Out		See AMBA Specification
CARD INTERFACE	SD_CLK	Out	1	CLK to external card
	DATA	In/Out	8	SD1/SD4/SD8 : Data line
	SDIO_CMD	In/Out	1	SD1/SD4/SD8 : Command line
	SDIO_WP	In	1	Active high. SD card write protect
	SDIO_CD	In	1	Active low. Card detection for single slot
	LED_ON	Out	1	To caution the user not to remove the card while the SD card is being accessed.
INTERRUPT	Interrupt to arm	Out	1	Interrupt request given to arm

32.4 Pin signals

The Arasan SD2.0/SDIO2.0 Host Controller has six main interface groups:

- ARM processor Interface signals.
- SD2.0/SDIO2.0/MMC4.2 Card Interface that forms the main card interface.
- System interface providing the clock and reset signals.
- RAM Interface signals

Table 611. AHB master/Target interface

Signal	DIR	Description
clk_ahb	In	AHB System Clock
m_hbusreq	OUT	AHB Bus request
m_hgrant	IN	AHB Bus grant
m_haddr[31:0]	OUT	DWord Address
m_hwdata[31:0]	OUT	AHB master write data
m_hrdata[31:0]	IN	Read data
m_hwrite	OUT	Write / Read Direction Indication
m_hsize[2:0]	OUT	Size (byte, half word or word)
m_hburst[2:0]	OUT	Burst Size
m_hready	IN	Ready signals
m_htrans[1:0]	OUT	Transfer type
m_hresp[1:0]	IN	Transfer response
int_to_arm	OUT	Interrupt to the ARM
t_hsel	IN	Slave select
t_haddr[5:0]	IN	DWord Address(256 bytes)
t_hwdata[31:0]	IN	Write data
t_hrdata[31:0]	OUT	Read data
t_hwrite	IN	Write / Read direction indication
t_hsize[2:0]	IN	Size (Byte, Half Word or Word)
t_htrans[1:0]	IN	Transfer Type
t_hready_in	IN	Slave ready input
t_hready	OUT	Slave ready
t_hresp[1:0]	OUT	Transfer response

Table 612. SD2.0/SDIO2.0/MMC4.2 card interface

Signal	DIR	Description
clk_sdcard_out	OUT	SD/SDIO/MMC Clock
clk_sdcard_in	IN	SD/SDIO/MMC Clock
CMD_IN	IN	SD1/SD4/SD8: Command input

Table 612. SD2.0/SDIO2.0/MMC4.2 card interface (continued)

Signal	DIR	Description
CMD_OUT	OUT	SD1/SD4/SD8 mode: Command output SPI : Command output and write data
CMD_OUT_EN	OUT	SD1/SD4/SD8 mode : Command output enable SPI mode: Command output enable and write data enable
DATA0_IN	IN	SD1/SD4/SD8 mode : Data0 input SPI mode : Command response input, read data and crc status for write data
DATA0_OUT	OUT	SD1/SD4/SD8 mode : Data0 output
DATA0_OUT_EN	OUT	SD1/SD4/SD8 mode : Data0 output enable
DATA1_IN	IN	SD1 mode : Interrupt SD4 mode : Data1 input or interrupt (optional) SD8 mode : Data1 input
DATA1_OUT	OUT	SD4/SD8 mode : Data1 ouptut
DATA1_OUT_EN	OUT	SD4/SD8 mode : Data1 output enable
DATA2_IN	IN	SD4/SD8 mode : Data2 input
DATA2_OUT	OUT	SD1 mode : Read wait (optional) SD4 mode : Data2 output or Read wait (optional) SD8 mode : Data2 output
DATA2_OUT_EN	OUT	SD1 mode : Read wait enable (optional) SD4 mode : Data2 output enable or Read wait enable (optional) SD8 mode : Data2 output enable
DATA3_IN	IN	SD4/SD8 mode : Data3 input
DATA3_OUT	OUT	SD4/SD8 mode : Data3 output SPI mode : chip select
DATA3_OUT_EN	OUT	SD4/SD8 mode : Data3 output enable SPI mode : chip select enable
DATA4_IN	IN	SD8 mode : Data4 input
DATA4_OUT	OUT	SD8 mode : Data4 output
DATA4_OUT_EN	OUT	SD8 mode : Data4 output enable
DATA5_IN	IN	SD8 mode : Data5 input
DATA5_OUT	OUT	SD8 mode : Data5 output
DATA5_OUT_EN	OUT	SD8 mode : Data5 output enable
DATA6_IN	IN	SD8 mode : Data6 input
DATA6_OUT	OUT	SD8 mode : Data6 output
DATA6_OUT_EN	OUT	SD8 mode : Data6 output enable
DATA7_IN	IN	SD8 mode : Data7 input
DATA7_OUT	OUT	SD8 mode : Data7 output
DATA7_OUT_EN	OUT	SD8 mode : Data7 output enable

Table 612. SD2.0/SDIO2.0/MMC4.2 card interface (continued)

Signal	DIR	Description
SDCD_n	IN	Active Low. Card Detection for single slot
SDWP	IN	Active High. SD card write protect
led_on	OUT	LED ON : Tp caution the user not to remove the card while the SD card is being accessed

Table 613. System Interface

Signal	DIR	Description
clk_xin	IN	This SD clock input is used to generate SD Clock. This clock is also used to derive Sleep clk which is used to detect Card insertion / removal and card interrupt. For maximum efficiency this should be around 52Mhz.
clk_sleep_in	IN	Input Sleep Clock used to detect Card Insertion / removal and card interrupt.
clk_sleep_out	OUT	Output Sleep Clock - Internally generated Sleep Clock from clk_xin. To use External Sleep Clock for Card Detection: Connect the External Sleep Clock to clk_sleep_in Input. To use Internally generated Sleep Clock for Card Detection: At the top, connect both clk_sleep_in & clk_sleep_out together.
rstahb_n	IN	External reset
clk_sdram	OUT	clk to RAM in sd clock domain
test mode	IN	test mode signal

Table 614. RAM Interface signals

Signal	DIR	Description
AA_RAM1[7:0]	OUT	Address for PORT A FIFO 1
DA_RAM1[31:0]	OUT	Data in for PORT A FIFO 1
CENA_RAM1	OUT	Chip enable for PORT A FIFO 1
WENA_RAM1	OUT	Write enable for PORT A FIFO 1
AB_RAM1[7:0]	OUT	Address for PORT B FIFO 1
DB_RAM1[31:0]	OUT	Data in for PORT B FIFO 1
CENA_RAM1	OUT	Chip enable for PORT B FIFO 1
WENB_RAM1	OUT	Write enable for PORT B FIFO 1
AA_RAM2[7:0]	OUT	Address for PORT A FIFO 2
DA_RAM2[31:0]	OUT	Data in for PORT A FIFO 2
CENA_RAM2	OUT	Chip enable for PORT A FIFO 2
WENA_RAM2	OUT	Write enable for PORT A FIFO 2
AB_RAM2[7:0]	OUT	Address for PORT B FIFO 2

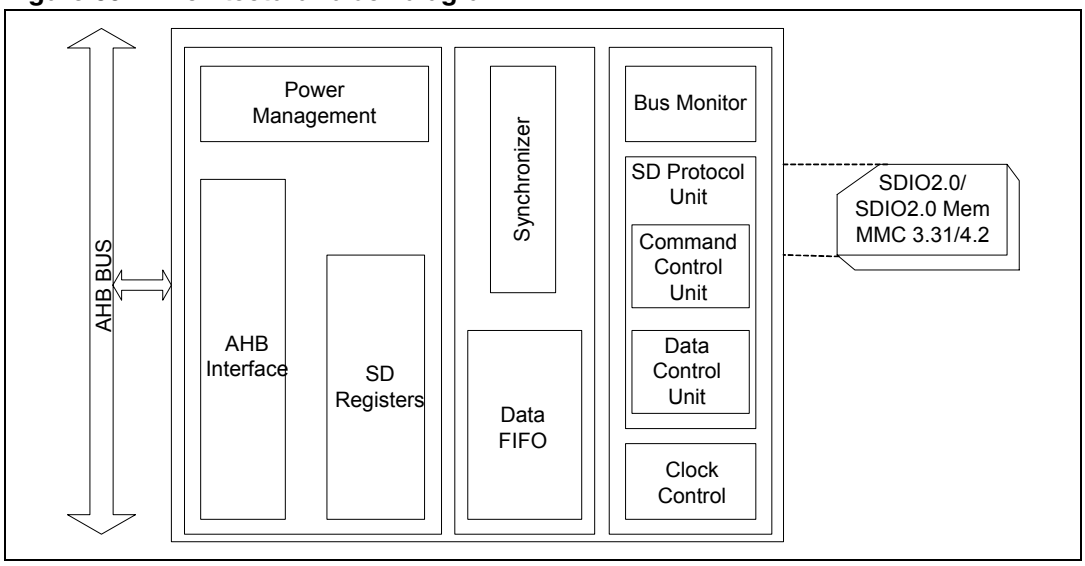
Table 614. RAM Interface signals (continued)

Signal	DIR	Description
DB_RAM2[31:0]	OUT	Data in for PORT B FIFO 2
CENB_RAM2	OUT	Chip enable for PORT B FIFO 2
WENB_RAM2	OUT	Write enable for PORT B FIFO 2
QA_RAM1[31:0]	IN	DATA OUT From PORT A FIFO 1
QB_RAM1[31:0]	IN	DATA OUT From PORT B FIFO 1
QA_RAM2[31:0]	IN	DATA OUT From PORT A FIFO 2
QB_RAM2[31:0]	IN	DATA OUT From PORT B FIFO 2

32.5 Functional overview

This section describes the functional overview of SDIO.

Figure 69. Architectural block diagram



The AHB interface acts as the bridge between AHB and Host Controller. The SD/SDIO controller registers are programmed by the ARM Processor through AHB target interface. Interrupts are generated to the ARM Processor based on the values set in the Interrupt status register and Interrupt enable registers. The Clock generation block will generate the SD clock depending on the value programmed by the ARM Processor in the Clock Control Register. The CRC7 and CRC16 generators calculate the CRC for command and Data respectively to send the CRC to the SD card. The CRC7 and CRC16 checker checks for any CRC error in the Response and Data sent by the SD/SDIO card.

The host controller alternatively uses two dual port 4KB FIFOs for read (data transferred from card to processor) and write (data transferred from processor to card) transactions.

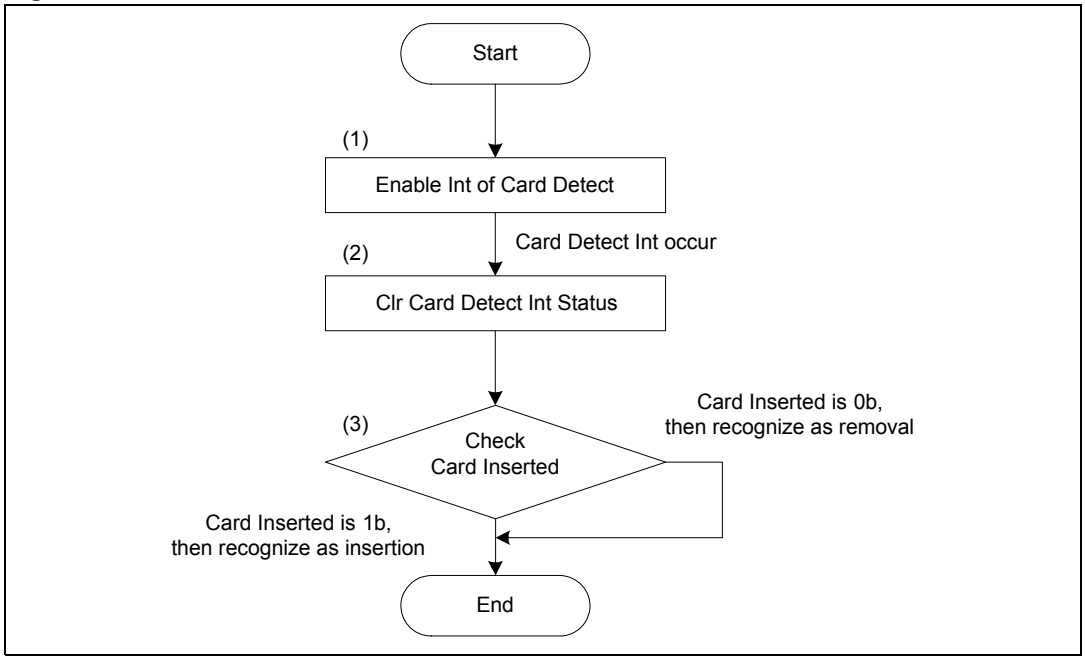
The DAT[0-7] control logic block transmits data in the data lines during write transaction and receives data in the data lines during read transaction.

The Command control logic block sends the command in the CMD line and receives the response coming from the SD2.0/SDIO2.0/MMC4.2 card.

32.5.1 Sequence

This section defines some basic sequence flows of initialization and data transfer with the card.

Figure 70. SD card detection

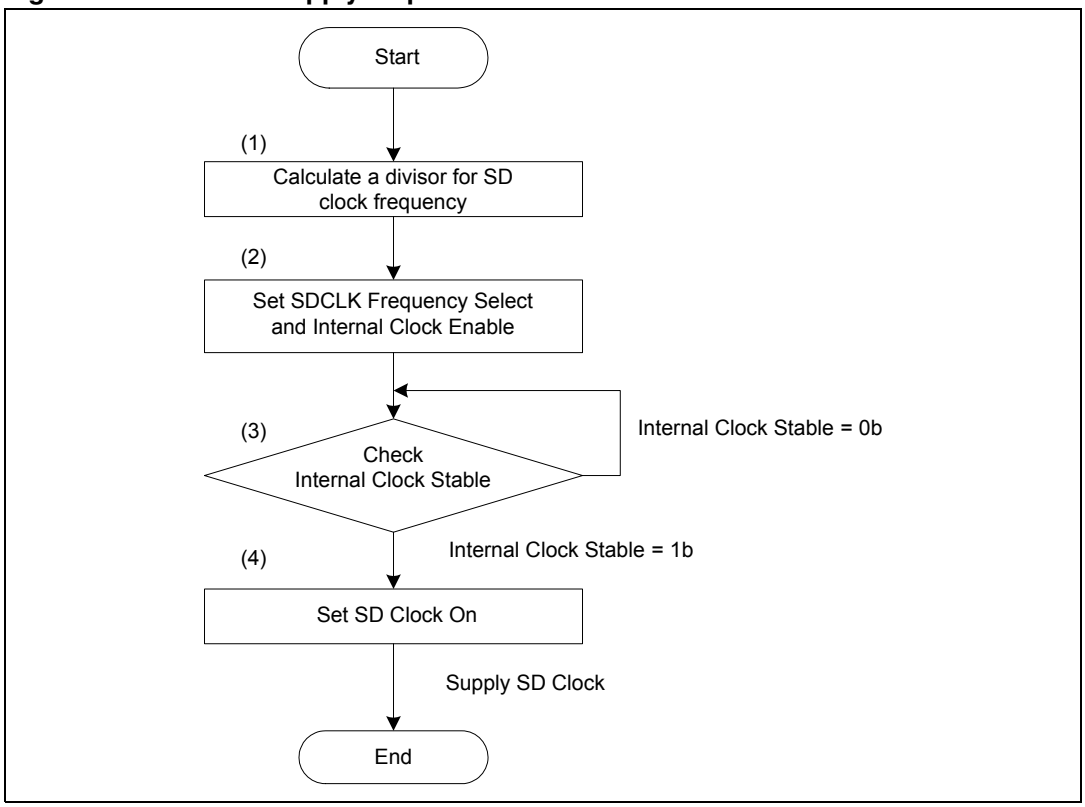


To enable interrupt for card detection, write 1 to the following bits:

- Card Insertion Status Enable in the Normal Interrupt Status Enable register
- Card Insertion Signal Enable in the Normal Interrupt Signal Enable register
- Card Removal Status Enable in the Normal Interrupt Status Enable register
- Card Removal Signal Enable in the Normal Interrupt Signal Enable register

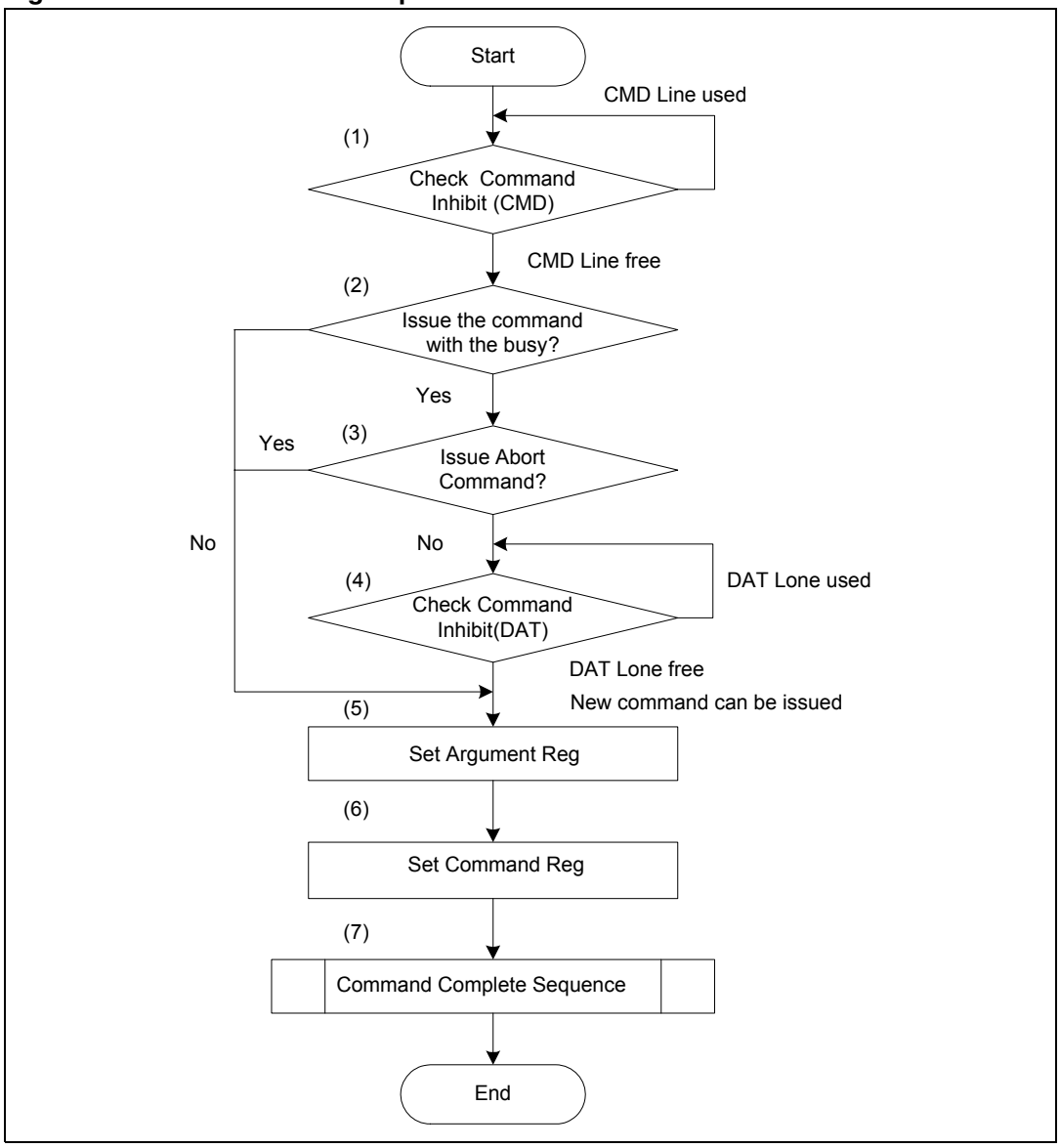
1. When the Host Driver detects the card insertion or removal, clear its interrupt statuses. If Card Insertion interrupt is generated, write logic ‘1’ to Card Insertion in the Normal Interrupt Status register. If Card Removal interrupt is generated, write logic ‘1’ to Card Removal in the Normal Interrupt Status register.
2. Check Card Inserted in the Present State register. In the case where Card Inserted is logic ‘1’, the Host Driver can supply the power and the clock to the SD card. In the case where Card Inserted is logic ‘0’, the other executing processes of the Host Driver shall be immediately closed.

Figure 71. SD clock supply sequence



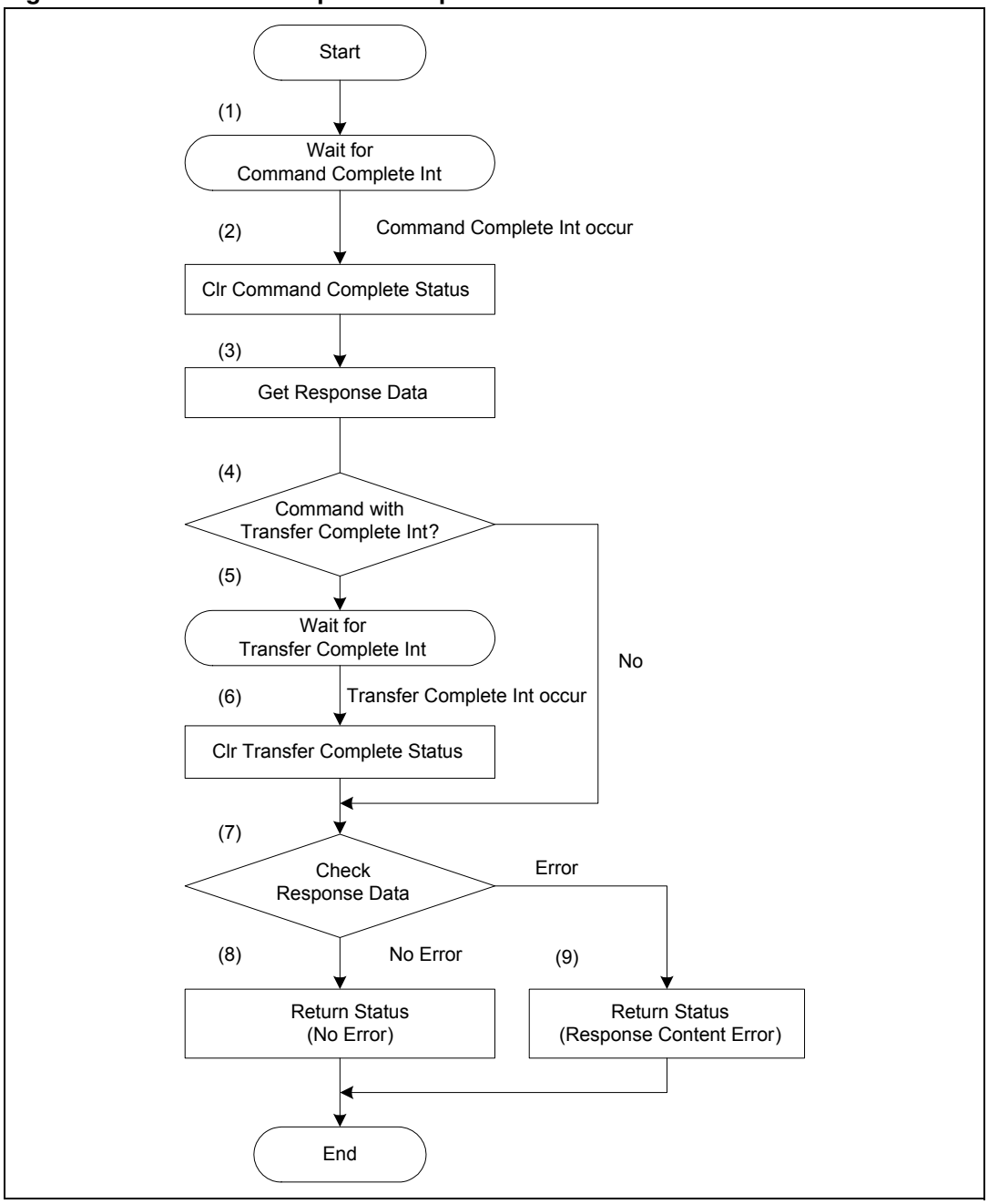
1. Calculate a divisor to determine SD Clock frequency.
2. Set Internal Clock Enable and SDCLK Frequency Select in the Clock Control register.
3. Check Internal Clock Stable in the Clock Control register. Repeat this step until Clock Stable is logic ‘1’.
4. Set SD Clock Enable in the Clock Control register to logic ‘1’. Then, the Host Controller starts to supply the SD Clock.

Figure 72. Command issue sequence



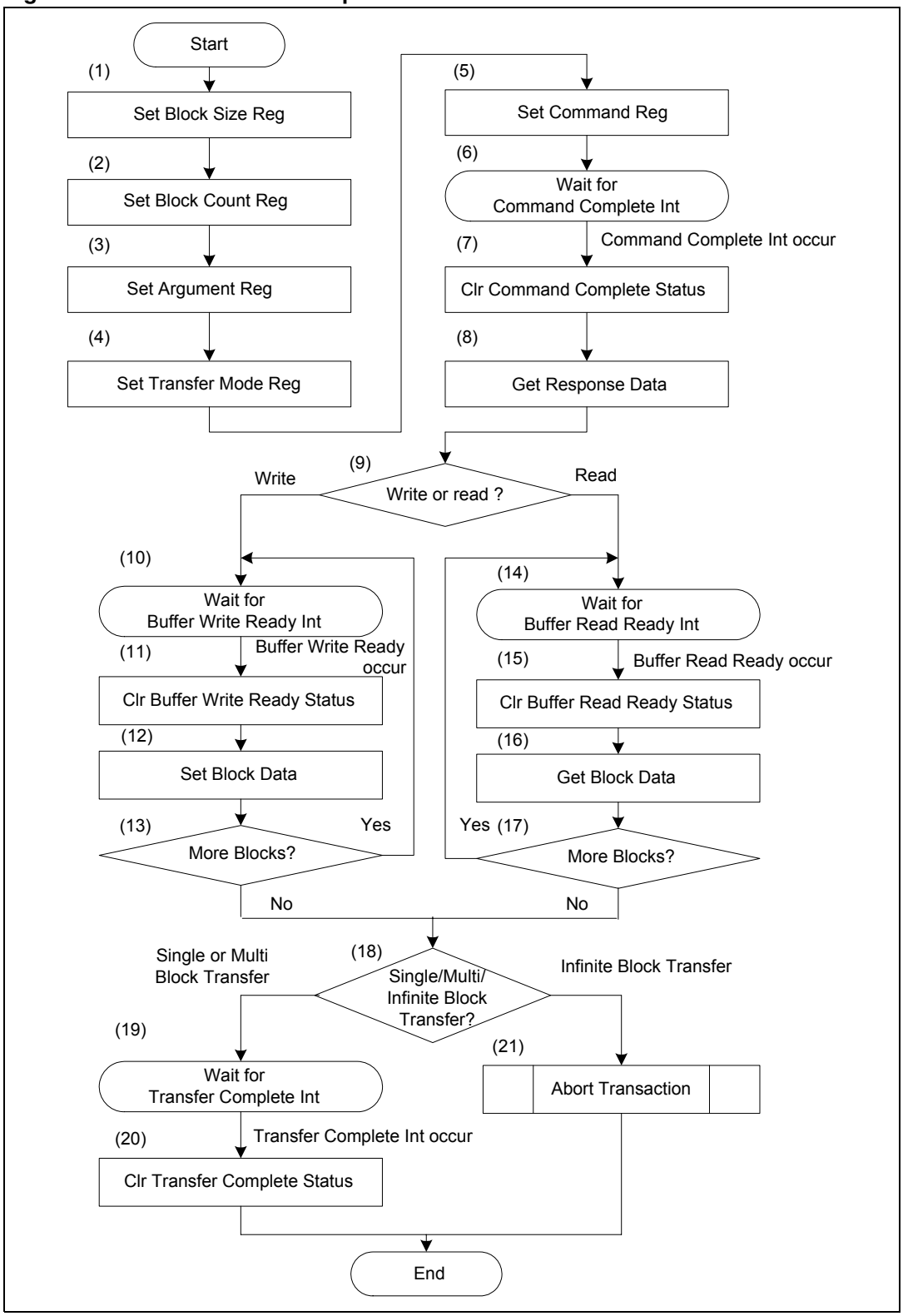
1. Check Command Inhibit (CMD) in the Present State register. Repeat this step until Command Inhibit (CMD) is logic ‘0’.
2. If the Host Driver issues a SD Command with busy signal, go to step (3). If without busy signal, go to step (5).
3. If the Host Driver issues an abort command, go to step (5). In 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 logic ‘0’.
5. Set the value of command argument to the Argument register.
6. Set the Command register.
7. Perform Command Completion Sequence.

Figure 73. Command completion sequence



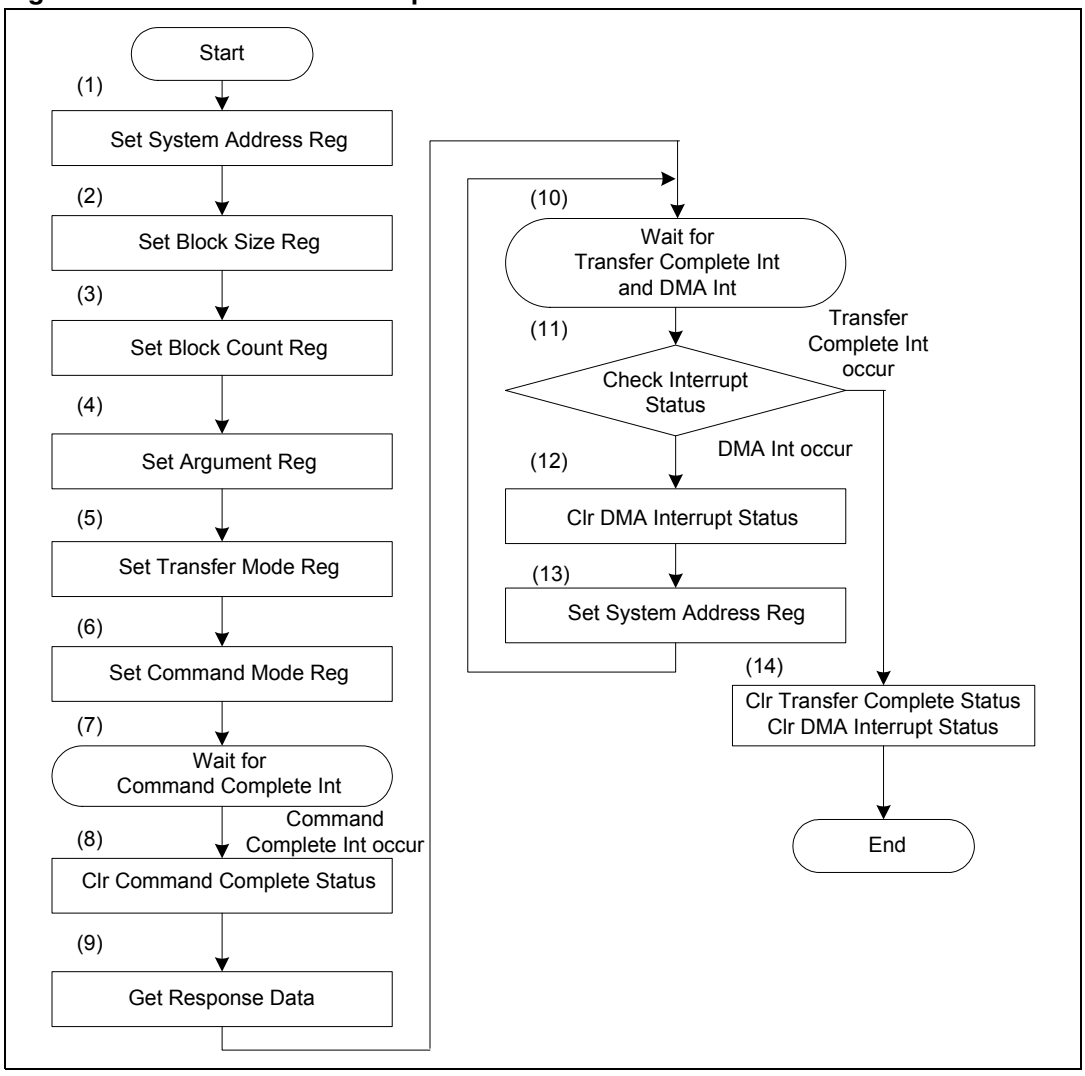
1. Wait for the Command Complete Interrupt. If the Command Complete Interrupt has occurred, go to step (2).
2. Write logic ‘1’ to Command Complete in the Normal Interrupt Status register to clear this bit.
3. Read the Response register and get necessary information of the issued command.
4. Judge whether 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 logic ‘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”.

Figure 74. Data transaction sequence without DMA



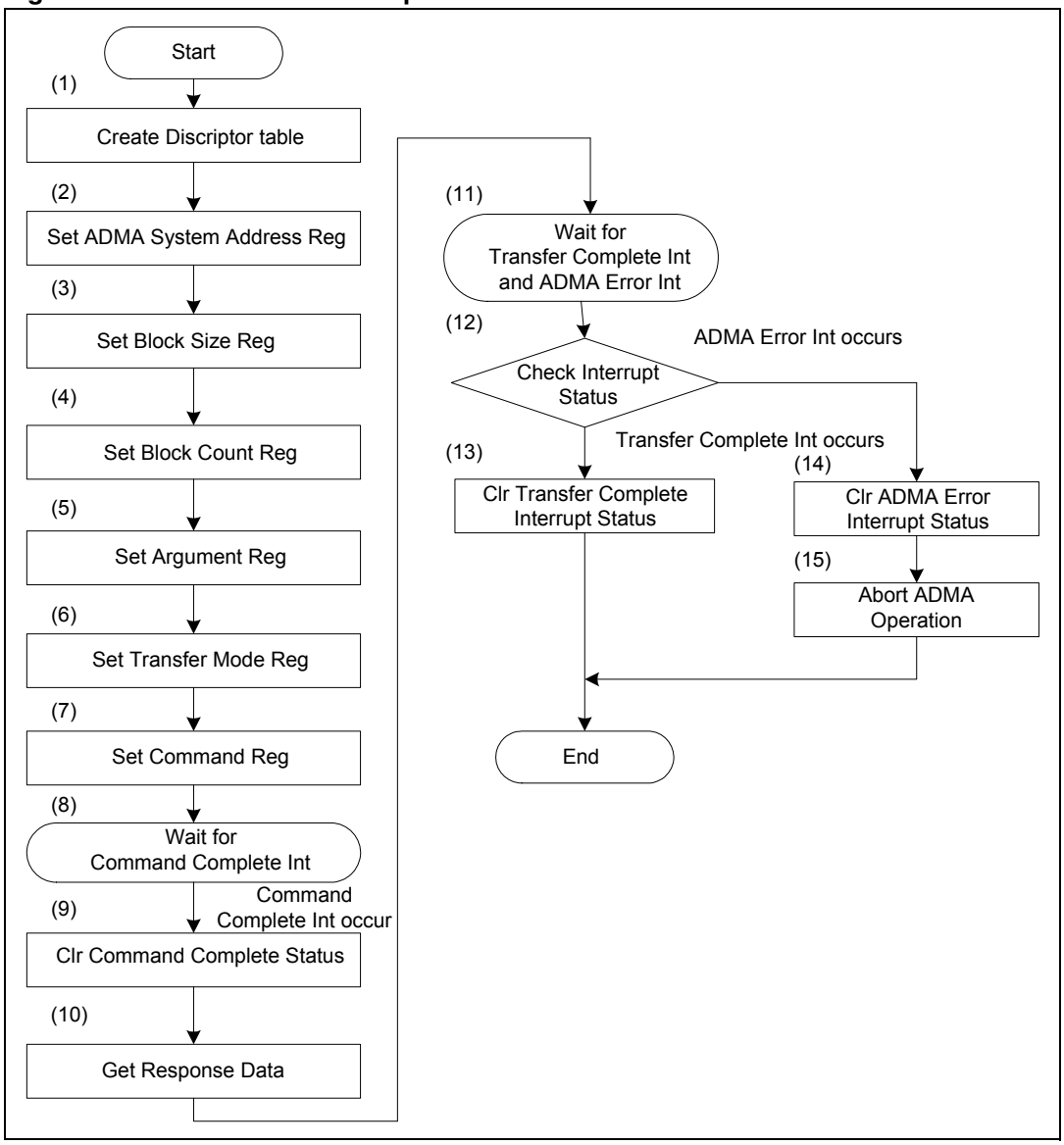
1. Set the value to Block Size register.
2. Set the value to Block Count register.
3. Set the argument value to Argument register.
4. Set the value to the Transfer Mode register. The host driver determines Multi / Single Block Select, Block Count Enable, Data Transfer Direction, Auto CMD12 Enable and DMA Enable.
5. Set the value to Command register.
6. Then, wait for the Command Complete Interrupt.
7. Write logic ‘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 logic ‘1’ to the Buffer Write Ready in the Normal Interrupt Status register for clearing this bit.
12. Write block data 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 logic ‘1’ to the Buffer Read Ready in the Normal Interrupt Status register for clearing this bit.
16. Read block data 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 logic ‘1’ to the Transfer Complete in the Normal Interrupt Status register for clearing this bit.
21. Perform the sequence for Abort Transaction

Figure 75. Data transaction sequence with SDMA



1. Data location of system memory is set to the SDMA System Address register.
2. Set the value to the Block Size register.
3. Set the value to the Block Count register.
4. Set the argument value to the Argument 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.
7. Then wait for the Command Complete Interrupt.
8. Write logic ‘1’ to the Command Complete in the Normal Interrupt Status register to clear 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 logic ‘1’, go to Step (14) else if DMA Interrupt is set to logic ‘1’, go to Step 12. Transfer Complete is higher priority than DMA Interrupt.
12. Write logic ‘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 logic ‘1’ to the Transfer Complete and DMA Interrupt in the Normal Interrupt Status register to clear this bit.

Figure 76. Data transaction sequence with ADMA

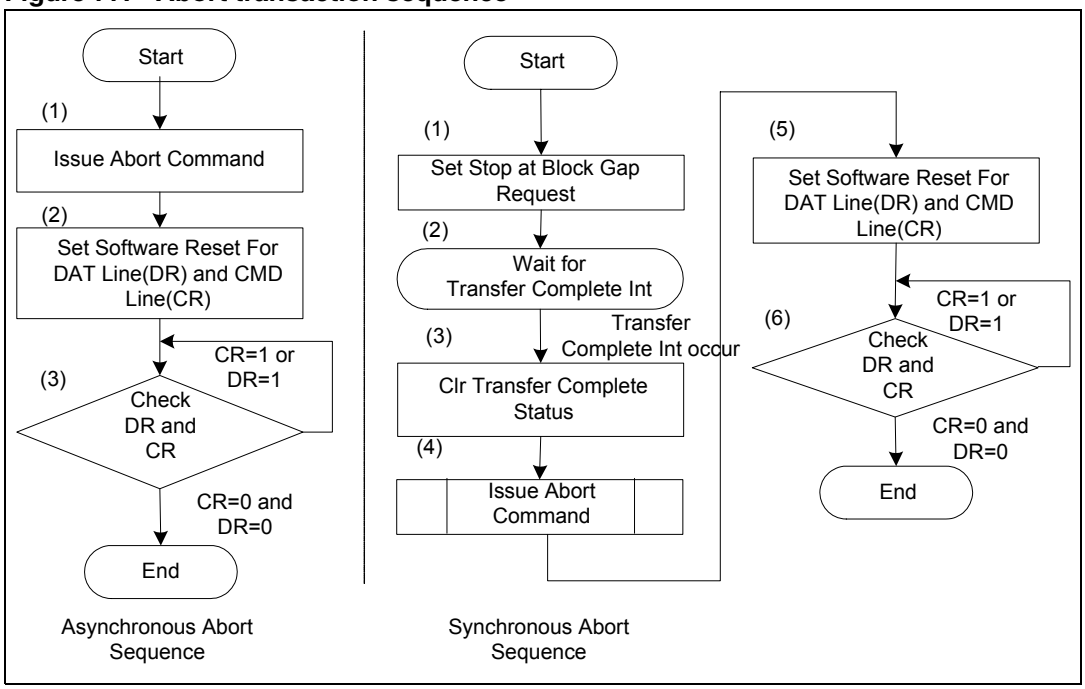


1. Create Descriptor table for ADMA in the system memory
2. Set the Descriptor address for ADMA in the ADMA System Address register.
3. Set the value corresponding to the Block Size register.
4. Set the value corresponding to the Block Count register. If the Block Count Enable in the Transfer Mode register is set to logic ‘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 16 bit Block Count register. If the Block Count Enable in the Transfer Mode register is set to logic ‘0’, total data length is designated by not Block Count register but the Descriptor Table. 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 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.
8. Then wait for the Command Complete Interrupt.
9. Write logic '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 logic '1', go to Step (13) else if ADMA Error Interrupt is set to logic '1', go to Step (14).
13. Write logic '1' to the Transfer Complete Status in the Normal Interrupt Status register to clear this bit.
14. Write logic '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.

Figure 77. Abort transaction sequence



An abort transaction is performed by issuing CMD12 for a SD memory card and by issuing CMD52 for a SDIO card. The host controller can do an abort transaction to stop infinite block transfers or stop transfers while a multiple block transfer is executing.

In an asynchronous abort sequence, the host controller can issue an abort command at anytime unless Command Inhibit (CMD) in the Present State register is set to logic '1'. In a

synchronous abort, the host controller shall issue an Abort Command after the data transfer stopped by using Stop At Block Gap Request in the Block Gap Control register.

32.6 Programmer's model

This section describes the programmer's model.

32.6.1 Register map

The SDIO Controller can be configured by programming registers through the AHB slave interface at base address. The registers are listed in detail in *Table 615.*

Table 615. SDIO registers map

Name	Offset	Size in bit	Description
SDMASysAddr	0x000	32	SDMA system address register
BLKSize	0x004	16	Block size register
BLKCnt	0x006	16	Block count register
CMDARG	0x008	32	Command argument register
TRMode	0x00C	16	Transfer mode register
CMD	0x00E	16	Command register
RESP0	0x010	32	Response register
RESP1	0x014	32	
RESP2	0x018	32	
RESP3	0x01C	32	
BufDataPort	0x020	32	Buffer data port register
PrState	0x024	32	Present state register
HOSTCTRL	0x028	8	Host control register
PWRCTRL	0x029	8	Power control register
BLKGAPCTRL	0x02A	8	Block gap control register
WKUPCTRL	0x02B	8	Wake up control register
CLKCTRL	0x02C	16	Clock control register
TMOUTCTRL	0x02E	8	Time out control register
SWRES	0x02F	8	Software reset register
NIRQSTAT	0x030	16	Normal Interrupt Status
ERRIRQSTAT	0x032	16	Error Interrupt Status
NIRQSTATEN	0x034	16	Normal interrupt Status Enable
ERRIRQSTATEN	0x036	16	Error Interrupt Status Enable
NIRQSIGEN	0x038	16	Normal Interrupt Signal Enable
ERRIRQSIGEN	0x03A	16	Error Interrupt Signal Enable
ACMD12ERSTS	0x03C	16	Auto Command 12 error status register

Table 615. SDIO registers map (continued)

Name	Offset	Size in bit	Description
-	0x03E	-	Reserved
CAP1	0x040	32	Capabilities registers
CAP2	0x044	32	Reserved
MAXCURR1	0x048	32	Maximum current capabilities register
MAXCURR2	0x04C	32	Reserved
ACMD12FEERSTS	0x050	16	Force event register for auto CMD12 error status
FEERRINTSTS	0x052	16	Force event register for error interrupt status
ADMAERRSTS	0x054	8	ADMA error status register
-	0x055 to0x057	-	Reserved
ADMAAddr1	0x058	32	ADMA LSB system address register
ADMAAddr2	0x05C	32	ADMA MSB system address register
-	0x060 to 0x0EE	-	Reserved
SPIIRQSUPP	0x0F0	8	SPI Interrupt request support register
-	0x0F2 to 0x0FA		Reserved
SLTIRQSTS	0x0FC	16	Slot Interrupts register
HCTRLVER	0x0FE	16	Host controller version register

Table 616. Register field types

Attribute	Description
RO	Read Only Register: Register bits are read only and cannot be altered by software or any reset operation. Write to these bits are ignored.
ROC	Read Only Status: These bits are initialized to zero at reset. Writes to these bits are ignored.
RW	Read-Write Register: Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read Only Status, Write logic ‘1’ to clear Status: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a logic ‘1’. Writing a logic ‘0’ to RW1C bits has no effect.
RWAC	Read-Write, automatic clear register: The Host driver requests a Host Controller operation by setting the bit. The Host Controller shall clear the bit automatically when the operation of complete. Writing a logic ‘0’ to RWAC bits has no effect.
Hwinit	Hardware Initialized: Register bits are freezed. Bits are read only after initialization, and writes to these bits are ignored.
Rsvd	Reserved: These bits are initialized to zero, and writes to them are ignored.

32.7 Register description

This section describes the register of SDIO.

32.7.1 SDMASysAddr register

This register contains the system memory address for a DMA transfer. When the Host Controller (HC) stops a DMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (that is after a transaction has stopped). Read operations during transfer return an invalid value. The Host Driver (HD) shall initialize this register before starting a DMA transaction.

After DMA has stopped, the next system address of the next contiguous data position can be read from this register.

The DMA transfer waits at every boundary specified by the Host DMA Buffer Size in the Block Size register. The Host Controller generates DMA Interrupt to request to update this register. The HD sets the next system address of the next data position to this register. When most upper byte of this register (0x003) is written, the HC restart the DMA transfer.

When restarting DMA by the resume command or by setting Continue Request in the Block Gap Control register, the HC shall start at the next contiguous address stored here in the System Address register.

The SDMASysAddr bit assignments are given in *Table 617*.

Table 617. SDMASysAddr register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	SDMASysAddr	32'h0	RW	This register contains the system memory address for a DMA transfer.

32.7.2 BLKSize register

The BLKSize bit assignments are given in *Table 618*.

Table 618. BLKSize register bit assignments

Bit	Name	Reset value	Type	Description
[15]	TBLKSize12	1'h0	Rsvd	Transfer Block Size 12th bit. This bit is added to support 4Kb Data block transfer.

Table 618. BLKSize register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[14:12]	HSDMABSize	3'h0	RW	To perform long DMA transfer, System Address register shall be updated at every system boundary during DMA transfer. These bits specify the size of contiguous buffer in the system memory. The DMA transfer shall wait at the every boundary specified by these fields and the HC generates the DMA Interrupt to request the HD to update the System Address register. These bits shall support when the DMA Support in the Capabilities register is set to logic '1' and this function is active when the DMA Enable in the Transfer Mode register is set to 1. 3'b000 - 4KB(Detects A11 Carry out) 3'b001 - 8KB(Detects A12 Carry out) 3'b010 - 16KB(Detects A13 Carry out) 3'b011 - 32KB(Detects A14 Carry out) 3'b100 - 64KB(Detects A15 Carry out) 3'b101 -128KB(Detects A16 Carry out) 3'b110 - 256KB(Detects A17 Carry out) 3'b111 - 512KB(Detects A18 Carry out)
[11:00]	TBKSize	12'h000	RW	This register specifies the block size for block data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. It can be accessed only if no transaction is executing (that is after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. 12'h0000 - No Data Transfer 12'h0001 - 1 Byte 12'h0002 - 2 Bytes 12'h0003 - 3 Bytes 12'h0004 - 4 Bytes --- --- 12'h01FF - 511 Bytes 12'h0200 - 512 Bytes --- --- 12'h0800 - 2048 Bytes

32.7.3 BLKCount register

The BLKCount bit assignments are given in *Table 619*.

Table 619. BLKCount register bit assignments

Bit	Name	Reset value	Type	Description
[15:00]	TBLKCount	16'h0000	RW	This register is enabled when Block Count Enable in the Transfer Mode register is set to logic '1' and is valid only for multiple block transfers. The HC decrements the block count after each block transfer and stops when the count reaches zero. It can be accessed only if no transaction is executing (that is after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. When saving transfer context as a result of Suspend command, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, the HD shall restore the previously save block count. 16'h0000 - Stop Count 16'h0001 - 1 block 16'h0002 - 2 blocks --- --- 16'hFFFF - 65535 blocks See also *Table 622*.

32.7.4 CMDARG register

The CMDARG bit assignments are given in *Table 620*.

Table 620. ARG register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	CMDARG	32'h0	RW	The SD Command Argument is specified as bit39-8 of Command Format.

32.7.5 TRMode register

The TRMode bit assignments are given in *Table 621*.

Table 621. TRMODE register bit assignments

Bit	Name	Reset value	Type	Description
[15:08]	-	-	Rsvd	Reserved.
[07]	SPIMode	1'h0	RW	SPI mode enable bit. 1'b1 - SPI mode 1'b0 - SD mode
[06]	-	-	Rsvd	Reserved.

Table 621. TRMODE register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[05]	MSBLKSel	1'h0	RW	This bit enables multiple block DAT line data transfers. 1'b0 - Single Block 1'b1 - Multiple Block See also *Table 622*.
[04]	DTDirSel	1'h0	RW	This bit defines the direction of DAT line data transfers. 1'b0 - Write (Host to Card) 1'b1 - Read (Card to Host)
[03]	-	-	Rsvd	Reserved.
[02]	ACMD12En	1'h0	RW	Multiple block transfers for memory require CMD12 to stop the transaction. When this bit is set to logic '1', the HC shall issue CMD12 automatically when last block transfer is completed. The HD shall not set this bit to issue commands that do not require CMD12 to stop data transfer. 1'b0 - Disable 1'b1 - Enable
[01]	BLKCntEn	1'h0	RW	This bit is used to enable the Block count register, which is only relevant for multiple block transfers. When this bit is logic '0', the Block Count register is disabled, which is useful in executing an infinite transfer. 1'b0 - Disable 1'b1 - Enable See also *Table 622*.
[00]	DMAEn	1'h0		DMA can be enabled only if DMA Support bit in the Capabilities register is set. If this bit is set to logic '1', a DMA operation shall begin when the HD writes to the upper byte of Command register (00Fh). 1'b0 - Disable 1'b1 - Enable

Table 622. Determination of transfer type

MSBLKSel	BLKCntEn	BLKCount	Function
0	Don't care	Don't care	Single transfer
1	0	Don't care	Infinite transfer
1	1	Not zero	Multiple transfer
1	1	Zero	Stop multiple transfer

32.7.6 CMD register

The CMD bit assignments are given in *Table 623*

Table 623. CMD register bit assignments

Bit	Name	Reset value	Type	Description
[15:14]	-	-	Rsvd	Reserved
[13:08]	CMDIndex	6'h0	RW	This bit shall be set to the command number (CMD0-63, ACMD0- 63).
[07:06]	CMDType	2'h0	RW	There are three types of special commands. Suspend, Resume and Abort. These bits shall be set to 2'b00 for all other commands. Suspend Command If the Suspend command succeeds, the HC shall assume the SD Bus has been released and that it is possible to issue the next command which uses the DAT line. The HC 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 HC shall maintain its current state. and the HD shall restart the transfer by setting Continue Request in the Block Gap Control Register. Resume Command The HD re-starts the data transfer by restoring the registers in the range of 0x000-0x00D. The HC shall check for busy before starting write transfers. Abort Command If this command is set when executing a read transfer, the HC shall stop reads to the buffer. If this command is set when executing a write transfer, the HC shall stop driving the DAT line. After issuing the Abort command, the HD should issue a software reset 2'b00 - Normal 2'b01 - Suspend 2'b10 - Resume 2'b11 - Abort
[05]	DPSel	1'h0	RW	This bit is set to logic '1' to indicate that data is present and shall be transferred using the DAT line. It is set to logic '0' for the following: Commands using only CMD line (ex. CMD52) Commands with no data transfer but using busy signal on DAT[0] line (R1b or R5b ex. CMD38) Resume Command 1'b0 - No Data Present 1'b1 - Data Present

Table 623. CMD register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[04]	IDXCkEn	1'h0	RW	If this bit is set to 1, the HC shall check the index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked. 1'b0 - Disable 1'b1 - Enable
[03]	CRCCkEn	1'h0	RW	If this bit is set to 1, the HC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. 1'b0 - Disable 1'b1 - Enable
[02]	-	-	Rsvd	Reserved
[01:00]	RESTypeSel	2'h0	RW	Response Type Select 2'b00 - No Response 2'b01 - Response length 136 2'b10 - Response length 48 2'b11 - Response length 48 check Busy after response

Table 624. Relation between parameters and the name of response type

RESTypeSel	IDXCkEn	CRCCkEn	Name of response type
00	0	0	No Response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R6, R5, R7
11	1	1	R1b, R5b

32.7.7 RESP(i) registers

The RESP bit assignments are given in *Table 625*.

Table 625. RESP register bit assignments

Bit	Name	Reset value	Type	Description
[127:00]	RESP	128'h0	ROC	*Table 626* describes the mapping of command responses from the SD Bus to this register for each response type. In the table, R[ ] refers to a bit range within the response data as transmitted on the SD Bus, RESP[ ] refers to a bit range within the Response register.

Table 626. Response bit definition for each response type

Kind of response	Meaning of response	Response field	Response register
R1, R1b (normal response)	Card Status	R[39:8]	RESP[31:0]
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R[39:8]	RESP[127:96]
R2 (CID, CSD Register)	CID or CSD reg. incl.	R[127:8]	RESP[119:0]
R3 (OCR Register)	OCR Register for memory	R[39:8]	RESP[31:0]
R4 (OCR Register)	OCR Register for I/O.	R[39:8]	RESP[31:0]
R5, R5b	SDIO Response	R[39:8]	RESP[31:0]
R6 (Published RCA response)	New published RCA[31:16].	R[39:8]	RESP[31:0]

32.7.8 Buf data port register

TheBufDataPort bit assignments are given in *Table 627*.

Table 627. BufDataPort register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	BUFDATA	-	RW	The Host Controller Buffer can be accessed through this 32 bit Data Port Register.

32.7.9 PRSTATE register

The PRSTATE bit assignments are given in *Table 628*.

Table 628. PRSTATE register bit assignments

Bit	Name	Reset value	Type	Description
[31:29]	-	-	Rsvd	Reserved
[28:25]	DAT[7:4]LSL	4'hF	RO	This status is used to check DAT line level to recover from errors, and for debugging. D28 - DAT[7] D27 - DAT[6] D26 - DAT[5] D25 - DAT[4]
[24]	CMDLSL	1'h1	RO	This status is used to check CMD line level to recover from errors, and for debugging.

Table 628. PRSTATE register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[23:20]	DAT[3:0]LSL	4'hF	RO	This status is used to check DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT[0]. D23 - DAT[3] D22 - DAT[2] D21 - DAT[1] D20 - DAT[0]
[19]	WPRSPL	1'h0	RO	The Write Protect Switch is supported for memory and combo cards. This bit reflects the SDWP# pin. 1'b0 - Write protected (SDWP# = 1) 1'b1 - Write enabled (SDWP# = 0)
[18]	CDPL	1'h0	RO	This bit reflects the inverse value of the SDCD# pin. 1'b0 - No Card present (SDCD# = 1) 1'b1 - Card present (SDCD# = 0)
[17]	CSS	1'h0	RO	This bit is used for testing. If it is logic '0', the Card Detect Pin Level is not stable. If this bit is set to logic '1', it means the Card Detect Pin Level is stable. The Software Reset For All in the Software Reset Register shall not affect this bit. 1'b0 - Reset of Debouncing 1'b1 - No Card or Inserted
[16]	CRDINS	1'h0	RO	This bit indicates whether a card has been inserted. 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. The Software Reset For All in the Software Reset register shall not affect this bit. If a Card is removed while its power is on and its clock is oscillating, the HC shall clear SD Bus Power in the Power Control register and SD Clock Enable in the Clock control register. In addition the HD should clear the HC by the Software Reset For All in Software register. The card detect is active regardless of the SD Bus Power. 1'b0 - Reset or Debouncing or No Card 1'b1 - Card Inserted
[15:12]	-	-	Rsvd	Reserved
[11]	BRE	1'h0	ROC	This status is used for non-DMA read transfers. This read only flag indicates that valid data exists in the host side buffer status. If this bit is logic '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. A change of this bit from 0 to 1 occurs when all the block data is ready in the buffer and generates the Buffer Read Ready Interrupt. 1'b0 - Read Disable 1'b1 - Read Enable.

Table 628. PRSTATE register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[10]	BWE	1'h0	ROC	This status is used for non-DMA write transfers. This read only flag indicates if space is available for write data. If this bit is logic '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. 1'b0 - Write Disable 1'b1 - Write Enable.
[09]	RTA	1'h0	ROC	This status is used for detecting completion of a read transfer. This bit is set to logic '1' for either of the following conditions: After the end bit of the read command When writing a logic '1' to continue Request in the Block Gap Control register to restart a read Transfer. This bit is cleared to 0 for either of the following conditions: When the last data block as specified by block length is transferred to the system. When all valid data blocks have been transferred to the system and no current block transfers are being sent as a result of the Stop At Block Gap Request set to logic '1'. A transfer complete interrupt is generated when this bit changes to 0. 1'b1 - Transferring data 1'b0 - No valid data
[08]	WTA	1'h0	ROC	This status indicates a write transfer is active. If this bit is logic '0', it means no valid write data exists in the HC. This bit is set in either of the following cases: After the end bit of the write command. When writing a logic '1' to Continue Request in the Block Gap Control register to restart a write transfer. This bit is cleared in either of the following cases: After getting the CRC status of the last data block as specified by the transfer count (Single or Multiple) After getting a CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. During a write transaction, a Block Gap Event interrupt is generated when this bit is changed to logic '0', as a result of the Stop At Block Gap Request being set. This status is useful for the HD in determining when to issue commands during write busy. 1'b1 - transferring data 1'b0 - No valid data

Table 628. PRSTATE register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[07:03]	-	-	Rsvd	Reserved
[02]	DATLA	1'h0	ROC	This bit indicates whether one of the DAT line on SD bus is in use. 1'b1 - DAT line active 1'b0 - DAT line inactive In the case of read transactions this status indicates if a read transfer is executing on the SD bus. Changes in this value from 1 to 0 between data blocks generate a Block Gap Event interrupt in the Normal Interrupt Status register. This bit shall be set in either of the following cases: 1. After the end bit of the read command 2. When writing a logic '1' to Continue Request in the Block Gap Control register to restart a read transfer. This bit shall be cleared in either of the following cases: 1. When the end bit of the last data block is sent from the SD bus to the HC. 2. When beginning a wait read transfer at a shop at the block gap initiated by a Stop At Block Gap Request.
[01]	CMDINBDAT	1'h0	ROC	This status bit is generated if either the DAT Line Active or the Read transfer Active is set to 1. If this bit is 0, it indicates the HC can issue the next SD command. Commands with busy signal belong to Command Inhibit (DAT) (ex. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Normal interrupt status register. Note: The SD Host Driver can save registers in the range of 0x000-0x00Dh for a suspend transaction after this bit has changed from 1 to 0. 1'b1 - cannot issue command which uses the DAT line 1'b0 - Can issue command which uses the DAT line
[00]	CMDINBCMD	1'h0	ROC	If this bit is logic '0', it indicates the CMD line is not in use and the HC can issue a SD command using the CMD line. This bit is set immediately after the Command register (0x00F) is written. This bit is cleared when the command response is received. Even if the Command Inhibit (DAT) is set to logic '1', Commands using only the CMD line can be issued if this bit is logic '0'. Changing from 1 to 0 generates a Command complete interrupt in the Normal Interrupt Status register. If the HC 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. Status issuing Auto CMD12 is not read from this bit.

32.7.10 HOSTCTRL register

The HOSTCTRL bit assignments are given in *Table 629*.

Table 629. HOSTCTRL register bit assignments

Bit	Name	Reset value	Type	Description
[07]	CDSD	1'h0	RW	This bit selects source for card detection. 1'b1- The card detect test level is selected 1'b0 -SDCD# is selected (for normal use)
[06]	CDTL	1'h0	RW	This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not. 1'b1 - Card Inserted 1'b0 - No Card
[05]	SD8MODE	1'h0	RW	This bit selects the data width of the HC. The HD shall select it to match the data width of the SD card. 1'b1 - 8 bit mode is selected 1'b0 - 8 bit mode is not selected
[04:03]	DMASEL	2'h0	RW	One of supported DMA modes can be selected. The host driver shall check support of DMA modes by referring the Capabilities register. 2'b00 - SDMA is selected 2'b01 - 32 bit Address ADMA1 is selected 2'b10 -32 bit Address ADMA2 is selected 2'b11 - 64 bit Address ADMA2 is selected
[02]	HSEN	1'h0	RW	This bit is optional. Before setting this bit, the HD shall check the High Speed Support in the capabilities register. If this bit is set to logic '0' (default), the HC outputs CMD line and DAT lines at the falling edge of the SD clock (up to 25 MHz). If this bit is set to logic '1', the HC outputs CMD line and DAT lines at the rising edge of the SD clock (up to 50 MHz) 1'b1 - High Speed Mode 1'b0 - Normal Speed Mode
[01]	DTW	1'h0	RW	This bit selects the data width of the HC. The HD shall select it to match the data width of the SD card. 1'b1 - 4 bit mode 1'b0 - 1 bit mode
[00]	LEDCTRL	1'b0	RW	This bit is used to caution the user not to remove the card while the SD card is being accessed. It is not necessary to change for each transaction. 1'b1 - LED on 1'b0 - LED off

32.7.11 PWRCTL register

The PWRCTRL bit assignments are given in *Table 630*.

Table 630. PWRCTRL register bit assignments

Bit	Name	Reset value	Type	Description
[07:04]	-	-	Rsvd	Reserved
[03:01]	SDBVS	3'h0	RW	By setting these bits, the HD selects the voltage level for the SD card. Before setting this register, the HD 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 3'b111 - 3.3 Flattop. 3'b110 - 3.0 V(Typ.) 3'b101 - 1.8 V(Typ.) 3'b100 - 3'b000 - Reserved
[00]	SDBPWR	1'h0	RW	Before setting this bit, the SD host driver shall set SD Bus Voltage Select. If the HC detects the No Card State, this bit shall be cleared. 1'b1 - Power on 1'b0 - Power off

Note: Power for the SD card is provided on board; hence programming this register is not essential.

32.7.12 BLKGAPCTRL register

The BLKGAPCTRL bit assignments are given in *Table 631*.

Table 631. BLKGAPCTRL register bit assignments

Bit	Name	Reset value	Type	Description
[07:04]	-	-	Rsvd	Reserved
[03]	IRQBK	1'h0	RW	This bit is valid only in 4 bit mode of the SDIO card and selects a sample point in the interrupt cycle. Setting to logic '1' enables interrupt detection at the block gap for a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to logic '0'. When the HD detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card.

Table 631. BLKGAPCTRL register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[02]	RDWCTRL	1'h0	RW	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 DAT[2] line. Otherwise the HC has to stop the SD clock to hold read data, which restricts commands generation. When the HD 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 logic '1' otherwise DAT line conflict may occur. If this bit is set to logic '0', Suspend / Resume cannot be supported 1'b1 - Enable Read Wait Control 1'b0 - Disable Read Wait Control
[01]	CNTREQ	1'h0	RW	This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. To cancel stop at the block gap, set Stop At block Gap Request to logic '0' and set this bit to restart the transfer. The HC automatically clears this bit in either of the following cases: In the case of a read transaction, the DAT Line Active changes from 0 to 1 as a read transaction restarts. In the case of a write transaction, the Write transfer active changes from 0 to 1 as the write transaction restarts. Therefore it is not necessary for Host driver to set this bit to logic '0'. If Stop At Block Gap Request is set to logic '1', any write to this bit is ignored. 1'b1 - Restart 1'b0 - Ignored

Table 631. BLKGAPCTRL register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[00]	STPBKGPREQ	1'h0	RW	This bit is used to stop executing a transaction at the next block gap for non- DMA,SDMA and ADMA transfers. Until the transfer complete is set to logic '1', indicating a transfer completion the HD shall leave this bit set to logic '1'. Clearing both the Stop At Block Gap Request and Continue Request shall not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The HC shall honour Stop At Block Gap Request for write transfers, but for read transfers it requires that the SD card support Read Wait. Therefore the HD shall not set this bit during read transfers unless the SD card supports Read Wait and has set Read Wait Control to logic '1'. In case of write transfers in which the HD writes data to the Buffer Data Port register, the HD shall set this bit after all block data is written. If this bit is set to logic '1', the HD shall not write data to Buffer data port register. This bit affects Read Transfer Active, Write Transfer Active, DAT line active and Command Inhibit (DAT) in the Present State register. 1'b1 - Stop 1'b0 - Transfer

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the HC issues a Suspend command or the SD card accepts the Suspend command.

1. If the HD does not issue Suspend command, the Continue Request shall be used to restart the transfer.
2. If the HD issues a Suspend command and the SD card accepts it, a Resume Command shall be used to restart the transfer.
3. If the HD issues a Suspend command and the SD card does not accept it, the Continue Request shall be used to restart the transfer.

Any time Stop At Block Gap Request stops the data transfer, the HD shall wait for Transfer Complete (in the Normal Interrupt Status register) before attempting to restart the transfer. When restarting the data transfer by Continue Request, the HD shall clear Stop At Block Gap Request before or simultaneously.

32.7.13 WKUPCTRL register

The WKUPCTRL bit assignments are given in *Table 632*.

Table 632. WKUPCTRL register bit assignments

Bit	Name	Reset value	Type	Description
[07:03]	-	-	Rsvd	Reserved

Table 632. WKUPCTRL register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[02]	WEECDR	1'h0	RW	Wakeup Event Enable On SD Card Removal This bit enables wakeup event via Card Removal assertion in the Normal Interrupt Status register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1'b1 - Enable 1'b0 - Disable
[01]	WEECDI	1'h0	RW	Wakeup Event Enable On SD Card Insertion This bit enables wakeup event via Card Insertion assertion in the Normal Interrupt Status register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1'b1 - Enable 1'b0 - Disable
[00]	WEEIRDQ	1'h0	RW	Wakeup Event Enable On Card Interrupt This bit enables wakeup event via Card Interrupt assertion in the Normal Interrupt Status register. This bit can be set to logic '1' if FN_WUS (Wake Up Support) in CIS is set to logic '1'. 1'b1 - Enable 1'b0 - Disable

32.7.14 CLKCTRL register

TheCLKCTRL bit assignments are given in *Table 633*.

Table 633. CLKCTRL register bit assignments

Bit	Name	Reset value	Type	Description
[15:08]	SDCLKFSEL	8'h00	RW	This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly; rather this register holds the divisor of the Base Clock Frequency For SD clock in the capabilities register. Only the following settings are allowed. 8'h80 - base clock divided by 256 8'h40 - base clock divided by 128 8'h20 - base clock divided by 64 8'h10 - base clock divided by 32 8'h08 - base clock divided by 16 8'h04 - base clock divided by 8 8'h02 - base clock divided by 4 8'h01 - base clock divided by 2 8'h00 - base clock (48 MHz) Setting 0x00 specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor. But multiple bits should not be set. 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. The frequency of the SDCLK is set by the following formula: Clock Frequency = (Baseclock) / divisor. Thus choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency. Maximum Frequency = 48 MHz (base clock) Minimum Frequency = 187.5 kHz (48 MHz / 256)
[07:03]	-	-	Rsvd	Reserved
[02]	SDCLKEN	1'h0	RW	The HC shall stop SDCLK when writing this bit to logic '0'. SDCLK frequency Select can be changed when this bit is logic '0'. Then, the HC shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK = 1'b0). If the HC detects the No Card state, this bit shall be cleared. 1'b1 - Enable 1'b0 - Disable

Table 633. CLKCTRL register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[01]	INCLKST	1'h0	ROC	This bit is set to logic '1' when SD clock is stable after writing to Internal Clock Enable in this register to logic '1'. The SD Host Driver shall wait to set SD Clock Enable until this bit is set to 1. Note: This is useful when using PLL for a clock oscillator that requires setup time. 1'b1 - Ready 1'b0 - Not Ready
[00]	INCLKEN	1'h0	RW	This bit is set to logic '0' when the HD is not using the HC or the HC awaits a wakeup event. The HC should stop its internal clock to go very low power state. Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to logic '1'. When clock oscillation is stable, the HC shall set Internal Clock Stable in this register to logic '1'. This bit shall not affect card detection. 1'b1 - Oscillate 1'b0 - Stop

32.7.15 TMOUTCTRL register

The TMOUTCTRL bit assignments are given in *Table 634*.

Table 634. TMOUTCTRL register bit assignments

Bit	Name	Reset value	Type	Description
[07:04]	-	-	Rsvd	Reserved
[03:00]	DATATMCNT	4'h0	RW	This value determines the interval by which DAT line time-outs are detected. Refer to the Data Time-out Error in the Error Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the base clock TMCLK by this value. When setting this register, prevent inadvertent time-out events by clearing the Data Time-out Error Status Enable (*Table 32.7.20*). 4'b1111 - Reserved 4'b1110 - TMCLK * 2^27 ---------- ---------- 4'b0001 - TMCLK * 2^14 4'b0000 - TMCLK * 2^13

Note: At the initialization of the HC, the HD shall set the Data Time-out Counter Value according to the Capabilities register.

32.7.16 SWRES register

The SWRES bit assignments are given in *Table 635*.

Table 635. SWRES register bit assignments

Bit	Name	Reset value	Type	Description
[07:03]	-	-	Rsvd	Reserved
[02]	SWRESDAT	-	RWAC	Only part of data circuit is reset. The following registers and bits are cleared by this bit: 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 Block Gap Event Transfer Complete 1'b1 - Reset 1'b0 - Work
[01]	SWRESCMD	-	RWAC	Only part of command circuit is reset. The following registers and bits are cleared by this bit: Present State register Command Inhibit (CMD) Normal Interrupt Status register Command Complete 1'b1 - Reset 1'b0 - Work
[00]	SWRESALL	-	RWAC	This reset affects the entire HC except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to logic '0'. During its initialization, the HD shall set this bit to logic '1' to reset the HC. The HC shall reset this bit to logic '0' when capabilities registers are valid and the HD 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 SD card shall reset itself and must be re initialized by the HD. 1'b1 - Reset 1'b0 - Work

Note: A reset pulse is generated when writing logic '1' to each bit of this register. After completing the reset, the HC shall clear each bit. Because it takes some time to complete software reset, the SD Host Driver shall confirm that these bits are logic '0'.

32.7.17 NIRQSTAT register

The Normal Interrupt Status Enable affects read of this register, but Normal Interrupt Signal does not affect these reads. An Interrupt is generated when the Normal Interrupt Signal Enable is enabled and at least one of the status bits is set to logic '1'. For all bits except Card Interrupt and Error Interrupt, writing logic '1' to a bit clears it. The Card Interrupt is cleared when the card stops asserting the interrupt: that is when the Card Driver services the Interrupt condition. The NIRQSTAT bit assignments are given in *Table 636*.

Table 636. NIRQSTAT register bit assignments

Bit	Name	Reset value	Type	Description
[15]	ERRINT	1'h0	ROC	If any of the bits in the Error Interrupt Status Register are set, then this bit is set. Therefore the HD can test for an error by checking this bit first. 1'b0 - No Error. 1'b1 - Error.
[14:09]	-	-	Rsvd	Reserved
[08]	CDINT	1'h0	ROC	Writing this bit to logic '1' does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1 bit mode, the HC shall detect the Card Interrupt without SD Clock to support wakeup. 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 card and the interrupt to the Host system. When this status has been set and the HD needs to start this interrupt service, Card Interrupt Status Enable in the Normal Interrupt Status register shall be set to logic '0' in order to clear the card interrupt statuses latched in the HC and stop driving the Host System. After completion of the card interrupt service (the reset factor in the SD card and the interrupt signal may not be asserted), set Card Interrupt Status Enable to logic '1 'and start sampling the interrupt signal again. 1'b0 - No Card Interrupt 1'b1 - Generate Card Interrupt
[07]	CDRINT	1'h0	RW1C	This status is set if the Card Inserted in the Present State register changes from 1 to 0. When the HD writes this bit to logic '1' to clear this status the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated. 1'b0 - Card State Stable or Debouncing 1'b1 - Card Removed

Table 636. NIRQSTAT register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[06]	CDIINT	1'h0	RW1C	This status is set if the Card Inserted in the Present State register changes from 0 to 1. When the HD writes this bit to logic '1' to clear this status the status of the Card Inserted in the Present State register should be confirmed. Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated. 1'b0 - Card State Stable or Debouncing 1'b1 - Card Inserted
[05]	BUFRDRDY	1'h0	RW1C	This status is set if the Buffer Read Enable changes from 0 to 1. 1'b0 - Not Ready to read Buffer. 1'b1 - Ready to read Buffer.
[04]	BUFWRRDY	1'h0	RW1C	This status is set if the Buffer Write Enable changes from 0 to 1. 1'b0 - Not Ready to Write Buffer. 1'b1 - Ready to Write Buffer.
[03]	DMAINT	1'h0	RW1C	This status is set if the HC detects the Host DMA Buffer Boundary in the Block Size register. 1'b0 - No DMA Interrupt 1'b1 - DMA Interrupt is Generated
[02]	BLKGAPE	1'h0	RW1C	If the Stop At Block Gap Request in the Block Gap Control Register is set, this bit is set. 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 must be supported in order to use this function). Write Transaction: This bit is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 1'b0- No Block Gap Event 1'b1 - Transaction stopped at Block Gap

Table 636. NIRQSTAT register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[01]	TRNCPL	1'h0	RW1C	This bit is set when a read / write transaction is completed. Read Transaction: This bit is set at the falling edge of Read Transfer Active Status. There are two cases in which the Interrupt is generated. 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). Write Transaction: This bit is set at the falling edge of the DAT Line Active Status. There are two cases in which the Interrupt is generated. The first is when the last data is written to the card as specified by data length and Busy signal is 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. (After valid data is written to the SD card and the busy signal is released). Transfer Complete has higher priority than Data Time-out Error. If both bits are set to logic '1', the data transfer can be considered complete. 1'b0 - No Data Transfer Complete 1'b1 - Data Transfer Complete
[00]	CMDCPL	1'h0	RW1C	This bit is set when get the end bit of the command response (Except Auto CMD12). Command Time-out Error has higher priority than Command Complete. If both are set to logic '1', it can be considered that the response was not received correctly. 1'b0 - No Command Complete 1'b1 - Command Complete

Table 637. Relation between transfer complete and data time out error

Transfer complete	Command time out error	Meaning of the status
0	0	Interrupted by Another Factor.
0	1	Timeout occur during transfer.
1	Don't care	Data Transfer Complete.

Table 638. Relation between command complete and time out error

Command complete	Command time out 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

32.7.18 ERRIRQSTAT register

Status defined in this register can be enabled by the Error Interrupt Status Enable Register, but not by the Error Interrupt Signal Enable Register. The Interrupt is generated when the Error Interrupt Signal Enable is enabled and a logic 't' least one of the statuses is set to logic '1'. Writing to logic '1' clears the bit and writing to 0 keeps the bit unchanged. More than one status can be cleared at the one register write.The ERRIRQSTAT bit assignments are given in *Table 639*.

Table 639. ERRIRQSTAT register bit assignments

Bit	Name	Reset value	Type	Description
[15:14]	VDSERRSTS	1'h0	RW1C	Vendor Specific Error Status Additional status bits can be defined in this register by the vendor.
[13]	CEATAERR	1'h0	RW1C	This occurs when ATA command termination has occurred due to an error condition the device has encountered. 1'b0 - no error 1'b1 - error
[12]	TGTRESERR	1'h0	RW1C	Target Response error Occurs when detecting ERROR in m_hresp (dma transaction) 1'b0 - no error 1'b1 - error
[11:10]	-	-	Rsvd	Reserved
[09]	ADMAERR	1'h0	RW1C	ADMA Error This bit is set when the Host 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. 1'b1- Error 1'b0 -No error

Table 639. ERRIRQSTAT register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[08]	ACMD12ERR	1'h0	RW1C	Auto CMD12 Error Occurs when detecting that one of the bits in AutoCMD12 Error Status register has changed from 0 to 1. This bit is set to logic '1' also when Auto CMD12 is not executed due to the previous command error. 1'b0 - No Error 1'b1 - Error
[07]	CURLERR	1'h0	RW1C	Current Limit Error By setting the SD Bus Power bit in the Power Control Register, the HC is requested to supply power for the SD Bus. If the HC 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 logic '1' means the HC is not supplying power to SD card due to some failure. Reading logic '0' means that the HC is supplying power and no error has occurred. This bit shall always set to be logic '0', if the HC does not support this function. 1'b0 - No Error 1'b1 - Power Fail
[06]	DATAEBERR	1'h0	RW1C	Data End Bit Error Occurs when detecting 0 at the end bit position of read data which uses the DAT line or the end bit position of the CRC status. 1'b0 - No Error 1'b1 - Error
[05]	DATACRCERR	1'h0	RW1C	Data CRC Error 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". 1'b0 - No Error 1'b1 - Error
[04]	DATATOERR	1'h0	RW1C	Data Timeout Error Occurs when detecting one of following timeout conditions: Busy Timeout for R1b, R5b type. Busy Timeout after Write CRC status Write CRC status Timeout Read Data Timeout 1'b0 - No Error 1'b1 - Timeout

Table 639. ERRIRQSTAT register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[03]	CMDIDXERR	1'h0	RW1C	Occurs if a Command Index error occurs in the Command Response. 1'b0 - No Error 1'b1 - Error
[02]	CMDEBERR	1'h0	RW1C	Occurs when detecting that the end bit of a command response is 0. 1'b0 - No Error 1'b1 - End Bit Error Generated
[01]	CMDCRCERR	1'h0	RW1C	Command CRC Error is generated in two cases: If a response is returned and the Command Time-out Error is set to logic '0', this bit is set to logic '1' when detecting a CRT error in the command response. The HC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the HC drives the CMD line to logic '1' level, but detects logic '0' level on the CMD line at the next SDCLK edge, then the HC shall abort the command (Stop driving CMD line) and set this bit to logic '1'. The Command Timeout Error shall also be set to logic '1' to distinguish CMD line conflict. 1'b0 - No Error 1'b1 - CRC Error Generated
[00]	CMDTOERR	1'h0	RW1C	Occurs only if the no response is returned within 64 SDCLK cycles from the end bit of the command. If the HC detects a CMD line conflict, in which case Command CRC Error shall also be set. This bit shall be set without waiting for 64 SDCLK cycles because the command will be aborted by the HC. 1'b0 - No Error 1'b1 - Timeout

Table 640. Relation between command CRC error end time out error

CMDCRCERR	CMDTOERR	Kind of error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

32.7.19 NIRQSTATEN register

The NIRQSTATEN bit assignments are given in *Table 641*.

Table 641. NIRQSTATEN register bit assignments

Bit	Name	Reset value	Type	Description
[15]	FIX0	1'h0	RO	The HC shall control error Interrupts using the Error Interrupt Status Enable register.
[14:09]	-	-	Rsvd	Reserved
[08]	CDIRQSTSEN	1'h0	RW	If this bit is set to logic '0', the HC 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 logic '1'. The HD should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this bit again after all Interrupt requests from the card are cleared to prevent inadvertent Interrupts. 1'b0 - Masked 1'b1 - Enabled
[07]	CDRSTSEN	1'h0	RW	1'b0 - Masked 1'b1 - Enabled
[06]	CDISTSEN	1'h0	RW	
[05]	BUFRDRDYEN	1'h0	RW	
[04]	BUFWRRDYEN	1'h0	RW	
[03]	DMAIRQSTSEN	1'h0	RW	
[02]	BLKGESTSEN	1'h0	RW	
[01]	TRNFCSTSEN	1'h0	RW	
[00]	CMDCSTSEN	1'h0	RW	

Note: Note: The host controller may sample the card Interrupt signal during interrupt period and may hold its value in the flip-flop. If the Card Interrupt Status Enable is set to logic '0', the HC shall clear all internal signals regarding Card Interrupt.

32.7.20 ERRIRQSTATEN register

The ERRIRQSTATEN bit assignments are given in *Table 642*.

Table 642. ERRIRQSTATEN register bit assignments

Bit	Name	Reset value	Type	Description
[15:14]	VDSERSTSEN	1'h0	RW1C	1'b0 - Masked 1'b1 - Enabled
[13]	CEATAERSTSEN	1'h0	RW1C	
[12]	TGTRESERSTSEN	1'h0	RW1C	
[11:10]	-	-	Rsvd	Reserved

Table 642. ERRIRQSTATEN register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[09]	ADMAERSTSEN			1'b0 - Masked 1'b1 - Enabled
[08]	ACMD12ERSTSEN			
[07]	CURLERSTSEN			
[06]	DATAEBSTSEN			
[05]	DATACRCERSTSEN			
[04]	DATATOERSTSEN			
[03]	CMDIDXERSTSEN			
[02]	CMDEBERSTSEN			
[01]	CMDCRCERSTSEN			
[00]	CMDTOERSTSEN			

32.7.21 NIRQSIGEN register

The NIRQSIGEN bit assignments are given in *Table 643*.

Table 643. NIRQSIGEN register bit assignments

Bit	Name	Reset value	Type	Description
[15]	FIX0	1'h0	RO	The HD shall control error Interrupts using the Error Interrupt Signal Enable register.
[14:09]	-	-	Rsvd	Reserved
[08]	CDSIGEN	1'h0	RW	1'b0 - Masked 1'b1 - Enabled
[07]	CDRSIGEN	1'h0	RW	
[06]	CDISIGEN	1'h0	RW	
[05]	BFRDRDYSIGEN	1'h0	RW	
[04]	BFWRRDYSIGEN	1'h0	RW	
[03]	DMAIRQSIGEN	1'h0	RW	
[02]	BLKGESIGEN	1'h0	RW	
[01]	TRFCPLSIGEN	1'h0	RW	
[00]	CMDCPLSIGEN	1'h0	RW	

32.7.22 ERRIRQSIGEN register

The ERRIRQSIGEN bit assignments are given in *Table 644*.

Table 644. ERRIRQSIGEN register bit assignments

Bit	Name	Reset value	Type	Description
[15:14]	VDSERSIGEN	1'h0	RW1C	1'b0 - Masked 1'b1 - Enabled
[13]	CEATAERSIGEN	1'h0	RW1C	
[12]	TGTRESERSIGEN	1'h0	RW1C	
[11:10]	-	-	Rsvd	Reserved
[09]	ADMAERSIGEN			1'b0 - Masked 1'b1 - Enabled
[08]	ACMD12ERSIGEN			
[07]	CURLERSIGEN			
[06]	DATAEBSIGEN			
[05]	DATACRCERSIGEN			
[04]	DATATOERSIGEN			
[03]	CMDIDXERSIGEN			
[02]	CMDEBERSIGEN			
[01]	CMDCRCERSIGEN			
[00]	CMDTOERSIGEN			

32.7.23 ACMD12ERSTS register

When Auto CMD12 Error Status is set, the HD shall check this register to identify what kind of error Auto CMD12 indicated. This register is valid only when the Auto CMD12 Error is set. The ACMD12ERSTS bit assignments are given in *Table 645*.

Table 645. ACMD12ERSTS register bit assignments

Bit	Name	Reset value	Type	Description
[15:08]	-	-	Rsvd	Reserved
[07]	CMDNIER	1'h0	ROC	This bit is set when the CMD_wo_DAT is not executed due to an Auto CMD12 error (bit 4:1) in this register. 1'b0 - No Error 1'b1 - Not Issued
[06:05]	-	-	Rsvd	Reserved
[04]	ACMD12IDXER	1'h0	ROC	Occurs if the Command Index error occurs in response to a command. 1'b0 - No Error 1'b1 - Error
[03]	ACMD12EBER	1'h0	ROC	Occurs when detecting that the end bit of command response is logic '0'. 1'b0 - No Error 1'b1 - End Bit Error Generated

Table 645. ACMD12ERSTS register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[02]	ACMD12CRCER	1'h0	ROC	Occurs when detecting a CRC error in the command response. 1'b0 - No Error 1'b1 - CRC Error Generated
[01]	ACMD12TOER	1'h0	ROC	Occurs if the no response is returned. within 64 SDCLK cycles from the end bit of the command. If this bit is set to logic '1', the other error status bits (bit 4:2) are meaningless. 1'b0 - No Error 1'b1 - Timeout
[00]	ACMD12NEX	1'h0	ROC	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. Setting this bit to logic '1' means the HC cannot issue Auto CMD12 to stop memory multiple block transfer due to some error. If this bit is set to logic '1', other error status bits (4:1) are meaningless. 1'b0 - Executed 1'b1 - Not Executed

Table 646. Relation between auto CMD12 CRC error and auto CMD12 timeout error

ACMD12CRCER	ACMD12TOER	Kind of error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

The timing of changing Auto CMD12 Error Status can be classified in three scenarios:

- When the HC is going to issue Auto CMD12.
  - Set bit 0 to 1 if Auto CMD12 cannot be issued due to an error in the previous command.
  - Set bit 0 to 0 if Auto CMD12 is issued.
- At the end bit of Auto CMD12 response.
  - Check received responses by checking the error bits 1, 2, 3, 4.
  - Set to 1 if Error is detected.
  - Set to 0 if Error is Not Detected.
- Before reading the Auto CMD12 Error Status bit 7
  - Set bit 7 to 1 if there is a command cannot be issued.
  - Set bit 7 to 0 if there is no command to issue.
  - Timing of generating the Auto CMD12 Error and writing to the Command register are Asynchronous.
  - Then bit 7 shall be sampled when driver never writing to the Command register. So just before reading the Auto CMD12 Error Status register is good timing to set the bit 7 status bit.

32.7.24 CAP1 register

The CAP1 bit assignments are given in *Table 647*.

Table 647. CAP1 register bit assignments

Bit	Name	Reset value	Type	Description
[31]	-	-	Rsvd	Reserved
[30]	SPIBLKMODE	1'h1	Hwinit	SPI block mode 1'b0 - Not Supported 1'b1 - Supported
[29]	SPIMODE	1'h1	Hwinit	SPI mode 1'b0 - Not Supported 1'b1 - Supported
[28]	64BITSUPP	1'h0	Hwinit	1'b1 - supports 64 bit system address 1'b0 - Does not support 64 bit system address
[27]	IRQMODE	1'h1	Hwinit	Interrupt mode 1'b0 - Not Supported 1'b1 - Supported
[26]	V18SUPP	1'h0	Hwinit	1'b0 - 1.8 V Not Supported 1'b1 - 1.8 V Supported
[25]	V30SUPP	1'h0	Hwinit	1'b0 - 3.0 V Not Supported 1'b1 - 3.0 V Supported
[24]	V33SUPP	1'h1	Hwinit	1'b0 - 3.3 V Not Supported 1'b1 - 3.3 V Supported

Table 647. CAP1 register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[23]	SUSRESSUP P	1'h1	Hwinit	This bit indicates whether the HC supports Suspend / Resume functionality. If this bit is logic '0', the Suspend and Resume mechanism are not supported and the HD shall not issue either Suspend / Resume commands. 1'b0 - Not Supported 1'b1 - Supported
[22]	SDMASUPP	1'h1	Hwinit	This bit indicates whether the HC is capable of using DMA to transfer data between system memory and the HC directly. 1'b0 - SDMA Not Supported 1'b1 - SDMA Supported.
[21]	HSSUPP	1'h1	Hwinit	This bit indicates whether the HC and the Host System support High Speed mode and they can supply SD Clock frequency from 25 MHz to 50 MHz. 1'b0 - High Speed Not Supported 1'b1 - High Speed Supported
[20]	-	-	Rsvd	Reserved
[19]	ADMA2SUPP	1'h1	Hwinit	1'b1 - ADMA2 support. 1'b0 - ADMA2 not support
[18]	EXTMDBSUP P	1'h1	Hwinit	This bit indicates whether the Host Controller is capable bus. 1'b1 - Extended Media Bus Supported 1'b0 - Extended Media Bus not Supported
[17:16]	MAXBLKLEN	2'h3	Hwinit	This value indicates the maximum block size that the HD can read and write to the buffer in the HC. The buffer shall transfer this block size without wait cycles. Three sizes can be defined as indicated below. 2'b00 - 512 byte 2'b01 - 1024 byte 2'b10 - 2048 byte 2'b11 - 4096 byte
[15:14]	-	-	Rsvd	Reserved

Table 647. CAP1 register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[13:08]	BCLKFREQ	6'h30	Hwinit	This value indicates the base (maximum) clock frequency for the SD clock. Unit values are 1 MHz. If the real frequency is 16.5 MHz, the larger value shall be set 0x11 (17 MHz) because the HD uses this value to calculate the clock divider value and it shall not exceed the upper limit of the SD clock frequency. The supported range is 10 MHz to 63 MHz. If these bits are all 0, the Host System has to get information via another method. Not 0 - 1 MHz to 63 MHz 0x00 - Get information via another method.
[07]	TOCLKU	1'h1	Hwinit	This bit shows the unit of base clock frequency used to detect Data Timeout Error. 1'b0 - kHz 1'b1 - MHz
[06]	-	-	Rsvd	Reserved
[05:00]	TOCLKFREQ	6'h30	Hwinit	This bit shows the base clock frequency used to detect Data Timeout Error. Not 0 - 1 kHz to 63 kHz or 1 MHz to 63 MHz 0x00- Get Information via another method.

32.7.25 CAP2 register

The CAP2 bit assignments are given in *Table 648*.

Table 648. CAP2 register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	-	-	Rsvd	Reserved

32.7.26 MAXCURR1 register

The MAXCURR1 bit assignments are given in *Table 649*.

Table 649. MAXCURR1 register bit assignments

Bit	Name	Reset value	Type	Description
[31:24]	-	-	Rsvd	Reserved
[23:16]	MAX18CURR	8'h00	Hwinit	Maximum current for 1.8V card. (See *Table 651*)
[15:08]	MAX30CURR	8'h00	Hwinit	Maximum current for 3.0V card. (See *Table 651*)
[07:00]	MAX33CURR	8'h01	Hwinit	Maximum current for 3.3V card. (See *Table 651*)

32.7.27 MAXCURR2 register

The MAXCURR2 bit assignments are given in *Table 650*.

Table 650. MAXCURR2 register bit assignments

Bit	Name	Reset value	Type	Description
[31:00]	-	-	Rsvd	Reserved

Table 651. Maximum current value definition

Register value (decimal)	Current value
0	Get information through another method
1	4 mA
2	8 mA
3	12 mA
---	---
---	---
255	1020 mA

32.7.28 ACMD12FEERSTS register

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Auto CMD12 Error Status Register can be written.

Writing logic ‘1’: set each bit of the Auto CMD12 Error Status Register

Writing logic ‘0’: no effect.

The ACMD12FEERSTS bit assignments are given in *Table 652*.

Table 652. ACMD12FEERSTS register bit assignments

Bit	Name	Reset value	Type	Description
[15:08]	-	-	Rsvd	Reserved
[07]	FECMDNI	1’h0	WO	Force Event for command not issued by Auto CMD12 Error 1’b1 - Interrupt is generated 1’b0 - no interrupt
[06:05]	-	-	Rsvd	Reserved
[04]	FEACMDIDX	1’h0	WO	Force Event for Auto CMD12 Index Error 1’b1 - Interrupt is generated 1’b0 - no interrupt
[03]	FEACMDEB	1’h0	WO	Force Event for Auto CMD12 End bit Error. 1’b1 - Interrupt is generated 1’b0 - no interrupt

Table 652. ACMD12FEERSTS register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[02]	FEACMDCRC	1'h0	WO	Force Event for Auto CMD12 CRC Error. 1'b1 - Interrupt is generated 1'b0 - no interrupt
[01]	FEACMDTO	1'h0	WO	Force Event for Auto CMD12 timeout Error. 1'b1 - Interrupt is generated 1'b0 - no interrupt
[00]	FEACMDNE	1'h0	WO	Force Event for Auto CMD12 NOT Executed. 1'b1 - Interrupt is generated 1'b0 - no interrupt

32.7.29 FEERRINTSTS register

The Force Event Register is not a physically implemented register. Rather, it is an address at which the Error Interrupt Status register can be written. The effect of a write to this address will be reflected in the Error Interrupt Status Register if the corresponding bit of the Error Interrupt Status Enable Register is set.

Writing logic '1': set each bit of the Error Interrupt Status Register

Writing logic '0': no effect

The FEERRINRSTS bit assignments are given in *Table 653*.

Table 653. FEERRINTSTS register bit assignments

Bit	Name	Reset value	Type	Description
[15:14]	FEVSERSTS	2'h0	WO	Force Event for Vendor Specific Error Status Additional status bits can be defined in this register by the vendor. 1'b1 - Interrupt is generated 1'b0 - No interrupt
[13]	FECEATAER	1'h0	WO	Force Event for Current Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[12]	FETRER	1'h0	WO	Force Event for Target Response Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[11:10]	-	-	Rsvd	Reserved
[09]	FEADMAER	1'h0	WO	Force Event for ADMA Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[08]	FEACMD12ER	1'h0	WO	Force Event for Auto CMD12 Error 1'b1 - Interrupt is generated 1'b0 - No interrupt

Table 653. FEERRINTSTS register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[07]	FECLER	1'h0	WO	Force Event for Current Limit Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[06]	FEDATAEBER	1'h0	WO	Force Event for Data End Bit Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[05]	FEDATACRCER	1'h0	WO	Force Event for Data CRC Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[04]	FEDATATOER	1'h0	WO	Force Event for Data Timeout Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[03]	FECMDIDXER	1'h0	WO	Force Event for Command Index Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[02]	FECMDEBER	1'h0	WO	Force Event for Command End Bit Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[01]	FECMDCRCER	1'h0	WO	Force Event for Command CRC Error 1'b1 - Interrupt is generated 1'b0 - No interrupt
[00]	FECMDTOER	1'h0	WO	Force Event for Command Timeout Error 1'b1 - Interrupt is generated 1'b0 - No interrupt

32.7.30 ADMAERRSTS register

When ADMA Error Interrupt occurs, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address Register holds the address around the error descriptor. The ADMAERRSTS bit assignments are given in *Table 654*.

Table 654. ADMAERRSTS register bit assignments

Bit	Name	Reset value	Type	Description
[07:03]	-	-	Rsvd	Reserved

Table 654. ADMAERRSTS register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[02]	ADMALMER	1'h0	RW	ADMA Length Mismatch Error This error occurs in the following 2 cases. 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. Total data length can not be divided by the block length. 1'b1 - Error 1'b0 - No error
[01:00]	ADMAERSTS	2'h0	RW	ADMA Error State 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. *Table 655* define these bits.

Table 655. ADMAERRSTS bits[1:0] definition

Bits[1:0]	ADMA Error State when error has 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 the next of the error descriptor

32.7.31 ADMAADDR1 register

The ADMAADDR1 bit assignments are given in *Table 656*.

32.7.32 ADMAADDR2 register

The ADMAADDR2 bit assignments are given in *Table 656*.

Table 656. ADMAADDR register bit assignments

Bit	Name	Reset value	Type	Description
[63:00]	ADMASYSADDR	64'h0	RW	This register holds byte address of executing command of the Descriptor table. 32 bit Address Descriptor uses lower 32 bit of this register. 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. ADMA2 ignores lower 2 bit of this register and assumes it to be 2'b00. See *Table 657* and *Table 658* to have more details.

Table 657. 32 bit address ADMA

ADMAADDR register value	32 bit system address
0x0000_0000	0x0000_0000
0x0000_0004	0x0000_0004
0x0000_0008	0x0000_0008
0x0000_000C	0x0000_000C
……	……
0xFFFF_FFFC	0xFFFF_FFFC

Table 658. 64 bit Address ADMA

ADMAADDR register value	64bit system address
0x0000_0000_0000_0000	0x0000_0000_0000_0000
0x0000_0000_0000_0004	0x0000_0000_0000_0004
0x0000_0000_0000_0008	0x0000_0000_0000_0008
0x0000_0000_0000_000C	0x0000_0000_0000_000C
……	……
0xFFFF_FFFF_FFFF_FFFC	0xFFFF_FFFF_FFFF_FFFC

32.7.33 SPIIRQSUPP register

The SPIIRQSUPP bit assignments are given in *Table 659*.

Table 659. SPIIRQSUPP register bit assignments

Bit	Name	Reset value	Type	Description
[07:00]	SPIIRQSUPP	8'h00	RW	This bit is set to indicate the assertion of interrupts in the SPI mode at any time, irrespective of the status of the card select (CS) line. If this bit is zero, then SDIO card can only assert the interrupt line in the SPI mode when the CS line is asserted.

32.7.34 SLTIRQSTS register

The SLTIRQSTS bit assignments are given in *Table 660*.

Table 660. SLTIRQSTS register bit assignments

Bit	Name	Reset value	Type	Description
[15:08]	-	-	Rsvd	Reserved
[07:00]	SLTIRQSIG	8'h00	ROC	These status bit indicate the logical OR of Interrupt signal and Wakeup signal for each slot. A maximum of 8 slots can be defined. If one interrupt signal is associated with multiple slots. the HD can know which interrupt is generated by reading these status bits. By a power on reset or by Software Reset For All, the Interrupt signal shall be de asserted and this status shall read 00h. Bit 00 - Slot 1 Bit 01 - Slot 2 Bit 02 - Slot 3 ----- ----- Bit 07 - Slot 8

32.7.35 HCTRLVER register

The HCTRLVER bit assignments are given in *Table 661*.

Table 661. HCTRLVER register bit assignments

Bit	Name	Reset value	Type	Description
[15:08]	VVN	8'h69	Hwinit	This status is reserved for the vendor version number. The HD should not use this status. This represents Host Controller IP release version.

Table 661. HCTRLVER register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[07:00]	SVN	8'h02	Hwinit	This Status indicates the Host Controller Spec Version. The Upper and Lower 4 bits indicate the version. 00 - SD Host Specification version 1.0 01 - SD Host Specification version 2.00 including only the feature of the Test Register. 02 - SD Host Specification version 2.00 including the feature of the Test Register and ADMA

33 RS_Color liquid crystal display controller (CLCD)

33.1 Overview

Within the Reconfigurable Array Subsystem, SPEAr300 can provide an ARM PrimeCell® Color Liquid Crystal Display Controller (CLCD) that provides all the necessary control signals to interface directly to a variety of color and monochrome LCD panels.

Main features of the Color Liquid Crystal Display Controller are:

- Compliance to the *AMBA specification (Rev 2.0)* onwards for easy integration into SoC implementation
- Dual 16-deep programmable 32 bit wide FIFOs for buffering incoming display data
- Supports single and dual panel mono *super twisted nematic* (STN) displays with 4 or 8 bit interfaces
- Supports single and dual-panel color and monochrome STN displays
- Supports thin film transistor (TFT) color displays
- Resolution programmable up to 1024 x 768
- 15 gray-level mono, 3375 color STN, and 32K color TFT support
- 1, 2, or 4 bits-per-pixel (bpp) palettized displays for mono STN
- 1, 2, 4 or 8 bpp palettized color displays for color STN and TFT
- 16 bits-per-pixel (bpp) true-color non-palettized, for color STN and TFT
- 24 bpp true-color non-palettized, for color TFT
- Programmable timing for different display panels
- 256 entry, 16 bit palette RAM, arranged as a 128 x 32 bit RAM physically frame, line and pixel clock signals
- AC bias signal for STN and data enable signal for TFT panels.
- Patented Gray Scale Algorithm.
- Supports little Endian, Big Endian and WindowsCE data formats.

Programmable parameters are:

- Horizontal front and back porch
- Horizontal synchronization pulse width
- Number of pixels per line
- Vertical front and back porch
- Vertical synchronization pulse width
- Number of lines per panel
- Number of panel clocks per line.
- Signal polarity, active HIGH or LOW
- AC panel bias
- Panel clock frequency
- Bits per pixel (Bpp)
- Display type, STN mono/color or TFT
- STN 4 or 8 bit interface mode
- STN dual or single panel mode
- Little-endian, big-endian, or WinCE mode
- Interrupt generation event.

LCD Panel resolutions can be programmed to values such as:

- 320x200, 320x240
- 640x200, 640x240, 640x480
- 800x600
- 1024x768.

Types of LCD panel supported are:

- Active matrix TFT panels with up to 24 bit bus interface
- Single-panel monochrome STN panels (4 bit and 8 bit bus interface)
- Dual-panel monochrome STN panels (4 bit and 8 bit bus interface per panel)
- Single-panel color STN panels, 8 bit bus interface
- Dual-panel color STN panels, 8 bit bus interface per panel.

33.1.1 Number of colors supported

Number of colors supported for TFT panels are:

- 1 bpp, palettized, 2 colors selected from available colors.
- 2 bpp, palettized, 4 colors selected from available colors.
- 4 bpp, palettized, 16 colors selected from available colors.
- 8 bpp, palettized, 256 colors selected from available colors.
- 16 bpp, direct 5:5:5 RGB, with one bpp not normally being used. This pixel is still output, and can be used as a bright bit to connect to the *least significant bit* (LSB) of R, G and B components of a 6:6:6 TFT panel.
- 24 bpp, direct 8:8:8 RGB, providing over 16 million colors.

Each 16 bit palette entry is composed of five bpp (RGB) plus a common intensity bit.

This gives better memory utilization and performance compared with a full six bpp structure. The total amount of colors supported can be doubled from 32K to 64K if the intensity bit is

used and applied to all three colors components simultaneously. Refer to *33.4* & *33.5* signal descriptions for more information.

Number of colors supported for color STN panels are:

- 1 bpp, palettized, 2 colors selected from 3375
- 2 bpp, palettized, 4 colors selected from 3375
- 4 bpp, palettized, 16 colors selected from 3375
- 8 bpp, palettized, 256 colors selected from 3375
- 16 bpp, direct 4:4:4 RGB, with 4 bpp not being used.

Number of colors supported for mono STN panels are:

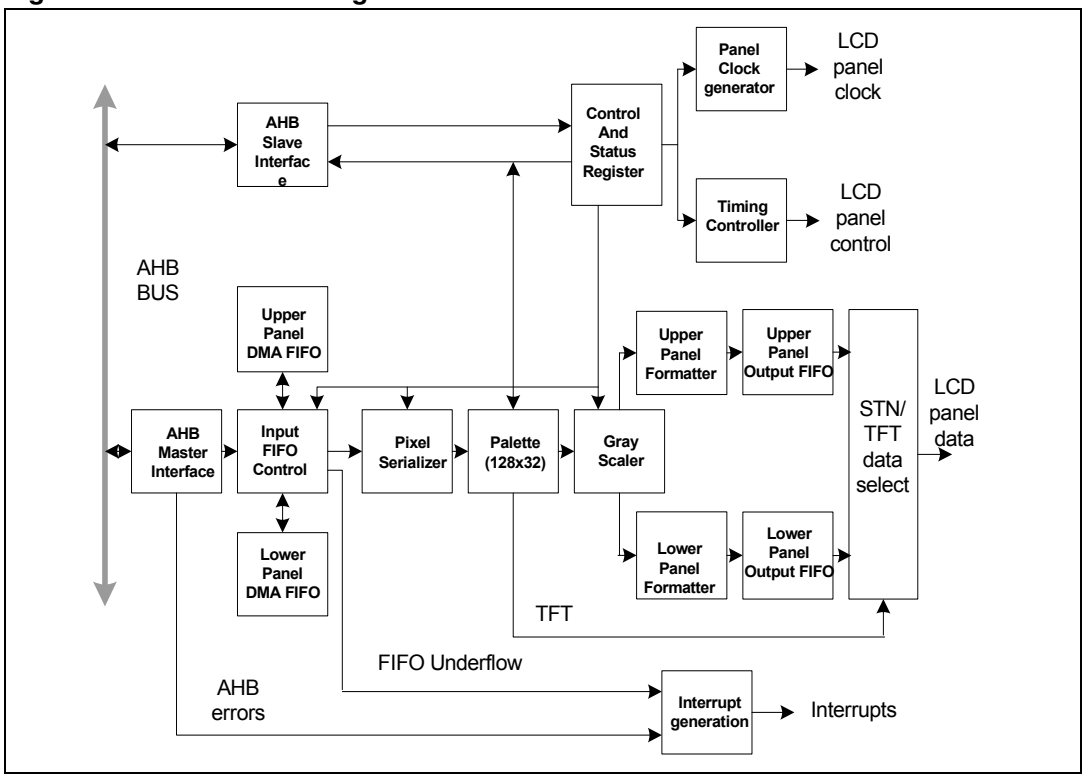
- 1 bpp, palettized, 2 gray scales selected from 15
- 2 bpp, palettized, 4 gray scales selected from 15
- 4 bpp, palettized, 16 gray scales selected from 15

You can program greater than four bpp for mono panels but using these modes does not make sense because the maximum number of gray scales supported on the display is 15.

33.2 Block diagram

Figure 78 shows the block diagram of CLCD.

Figure 78. CLCD block diagram



33.3 Signal interfaces

The CLCD directly interfaces with the signals summarized in *Table 662*.

Table 662. CLCD signal interface

Group	Signal name	Direction	Size (bit)	Description
External (to chip pads)	CLAC	Output	1	STN AC bias drive or TFT data enable output
	CLCP	Output	1	LCD panel clock
	CLD[23:0]	Output	24	LCD panel data
	CLFP	Output	1	Frame pulse (STN)/vertical synchronization pulse (TFT)
	CLLE	Output	1	Line end signal
	CLLP	Output	1	Line synchronization pulse (STN)/horizontal synchronization pulse (TFT)
	CLPOWER	Output	1	LCD panel power enable
Control signal	CLCD_INTR	Output	1	Combined OR version CLCD interrupt requests, to interrupt controller.
AHB slave	-	Input/Output	-	See AMBA specification.
AHB master	-	Input/Output	-	See AMBA specification.

33.4 LCD panel signal multiplexing details

The **CLLP**, **CLAC**, **CLFP**, **CLCP** and **CLLE** signals are common but the CLD[23:0] bus has eight modes of operation corresponding to:

- TFT 24 bit interface
- TFT 18 bit interface
- color STN single panel
- color STN Dual panel
- 4 bit mono STN single panel
- 4 bit mono STN dual panel
- 8 bit mono STN single panel
- 8 bit mono STN dual panel.

Note: 1 CUSTN = Color upper panel STN, dual and/or single panel

2 CLSTN = Color lower panel STN, single

3 MUSTN = Mono upper panel STN, dual and/or single panel

4 MLSTN = Mono lower panel STN, single.

Table 663 shows which **CLD[23:0]** pins are used to supply the pixel data to the STN panel for each of the above modes of operation.

Table 663. LCD STN panel signal multiplexing

External pin	Color STN single panel	Color STN dual panel	4 bit mono STN single panel	4 bit mono STN dual panel	8 bit mono STN single panel	8 bit mono STN dual panel
CLD [23]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [22]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [21]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [20]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [19]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [18]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [17]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [16]	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
CLD [15]	Reserved	CLSTN[0]	Reserved	Reserved	Reserved	MLSTN[0]
CLD [14]	Reserved	CLSTN[1]	Reserved	Reserved	Reserved	MLSTN[1]
CLD [13]	Reserved	CLSTN[2]	Reserved	Reserved	Reserved	MLSTN[2]
CLD [12]	Reserved	CLSTN[3]	Reserved	Reserved	Reserved	MLSTN[3]
CLD [11]	Reserved	CLSTN[4]	Reserved	MLSTN[0]	Reserved	MLSTN[4]
CLD [10]	Reserved	CLSTN[5]	Reserved	MLSTN[1]	Reserved	MLSTN[5]
CLD [9]	Reserved	CLSTN[6]	Reserved	MLSTN[2]	Reserved	MLSTN[6]
CLD [8]	Reserved	CLSTN[7]	Reserved	MLSTN[3]	Reserved	MLSTN[7]
CLD [7]	CUSTN[0]	CUSTN[0]	Reserved	Reserved	MUSTN[0]	MUSTN[0]
CLD [6]	CUSTN[1]	CUSTN[1]	Reserved	Reserved	MUSTN[1]	MUSTN[1]
CLD [5]	CUSTN[2]	CUSTN[2]	Reserved	Reserved	MUSTN[2]	MUSTN[2]
CLD [4]	CUSTN[3]	CUSTN[3]	Reserved	Reserved	MUSTN[3]	MUSTN[3]
CLD [3]	CUSTN[4]	CUSTN[4]	MUSTN[0]	MUSTN[0]	MUSTN[4]	MUSTN[4]
CLD [2]	CUSTN[5]	CUSTN[5]	MUSTN[1]	MUSTN[1]	MUSTN[5]	MUSTN[5]
CLD [1]	CUSTN[6]	CUSTN[6]	MUSTN[2]	MUSTN[2]	MUSTN[6]	MUSTN[6]
CLD [0]	CUSTN[7]	CUSTN[7]	MUSTN[3]	MUSTN[3]	MUSTN[7]	MUSTN[7]

Table 664 shows which CLD[23:0] pins are used to supply the pixel data to the TFT panel for each of the above modes of operation.

Table 664. LCD TFT panel signal multiplexing

External pin	TFT 24 bit	TFT 18 bit
CLD[23]	BLUE[7]	Reserved
CLD[22]	BLUE[6]	Reserved
CLD[21]	BLUE[5]	Reserved
CLD[20]	BLUE[4]	Reserved

Table 664. LCD TFT panel signal multiplexing (continued)

External pin	TFT 24 bit	TFT 18 bit
CLD[19]	BLUE[3]	Reserved
CLD[18]	BLUE[2]	Reserved
CLD[17]	BLUE[1]	BLUE[4]
CLD[16]	BLUE[0]	BLUE[3]
CLD[15]	GREEN[7]	BLUE[2]
CLD[14]	GREEN[6]	BLUE[1]
CLD[13]	GREEN[5]	BLUE[0]
CLD[12]	GREEN[4]	Intensity Bit
CLD[11]	GREEN[3]	GREEN[4]
CLD[10]	GREEN[2]	GREEN[3]
CLD[9]	GREEN[1]	GREEN[2]
CLD[8]	GREEN[0]	GREEN[1]
CLD[7]	RED[7]	GREEN[0]
CLD[6]	RED[6]	Intensity Bit
CLD[5]	RED[5]	RED[4]
CLD[4]	RED[4]	RED[3]
CLD[3]	RED[3]	RED[2]
CLD[2]	RED[2]	RED[1]
CLD[1]	RED[1]	RED[0]
CLD[0]	RED[0]	Intensity Bit

33.5 Main functions description

This section describes the main functions of crystal liquid crystal display controller.

33.5.1 AHB slave interface

The AMBA AHB slave interface connects the CLCD to the AMBA AHB bus and provides CPU accesses to the registers and palette RAM. For more information on AMBA AHB slave interfaces, refer to the *AMBA specification (Rev 2.0)*.

The following features are supported by the CLCD AMBA AHB slave interface:

- Standard write and read AMBA AHB accesses
- INCR4, INCR8, and undefined length WORD bursts only
- OKAY response only.

33.5.2 AHB master interface

The AMBA AHB master interface transfers display data from a memory to the PrimeCell CLCD DMA FIFOs.

The inherent AMBA AHB master interface state machine performs the following functions:

- Loads the upper panel base address into the AMBA AHB address incrementor on recognition of a new frame.
- Monitors both the upper and lower DMA FIFO levels and asserts HBUSREQM to request display data from memory, filling them to above the programmed water mark. HBUSREQM is re-asserted when there are at least four locations available within either FIFO (dual panel mode).
- Checks for 1KB boundaries during fixed-length bursts and appropriately adjusts the address in such occurrences.
- Generates the address sequences for fixed-length and undefined bursts.
- Controls the handshaking between the memory and DMA FIFOs. It inserts busy cycles if the FIFOs have not completed their synchronization and updating sequence.
- Fills up the DMA FIFOs, in dual panel mode, in an alternating fashion from a single HBUSREQM request and subsequent HGRANTM.
- Asserts the CLCDMBEINTR interrupt if an error occurs during an active burst.
- Responds to retry commands by restarting the failed access.

33.5.3 Dual DMA FIFOs and associated control logic

The pixel data accessed from memory is buffered by two DMA FIFOs that can be independently controlled to cover single and dual-panel LCD types. Each FIFO is 16 words deep by 32 bits wide and can be cascaded to form an effective 32-word deep FIFO in single-panel mode. The input ports of the FIFOs are connected to the AMBA AHB interface and the output port feeds the pixel serializer.

Synchronization logic is used to transfer the pixel data from the AMBA AHB `HCLK` domain to the CLCDCLK clock domain, the DMA FIFOs being clocked by the former.

The water level marks within each FIFO are set so that each FIFO requests data when at least four locations become available.

An interrupt signal is asserted if an attempt is made to read either of the two DMA FIFOs when they are empty, in other words an underflow condition has occurred.

33.5.4 Pixel serializer

This block reads the 32 bit wide LCD data from output port of the DMA FIFO and extracts 24, 16, 8, 4, 2, or 1 bpp data, depending on the current mode of operation. The CLCD supports big-endian, little-endian, and WinCE data formats. In dual panel mode, data is alternately read from the upper and lower DMA FIFOs. Depending upon the mode of operation, you can use the extracted data to point to a color/gray scale value in the palette RAM or it can be a true color value that you can apply directly to an LCD panel input.

The following tables shows the structure of the data in each DMA FIFO word corresponding to the endianness and bpp combinations. For each of the three supported data formats, the required data for each panel display pixel must be extracted from the data word.

The nomenclature used in the figures is:

- Little endian byte, little endian pixel (LBLP) order (*Table 665* and *Table 666*)
- Big endian byte, big endian pixel (BBBP) order (*Table 667* and *Table 668*)
- Little endian byte, big endian pixel (LBBP) order (this is the WinCE format) (**Powering up and down sequences***Table 669* and *Table 670*)

Table 665. LBLP, DMA FIFO output bit 31 to bit 16

bpp	DMA FIFO Output Bits															
	31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
1	p31	p30	p29	p28	p27	p26	p25	p24	p23	p22	p21	p20	p19	p18	p17	p16
2	p15		p14		p13		p12		p11		p10		p9		p8	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p7				p6				p5				p4			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	-	-	-	-	-	-	-	-	23	22	21	20	19	18	17	16

Table 666. LBLP, DMA FIFO output bit15 to bit 0

bpp	DMA FIFO Output Bits															
	15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
1	p15	p14	p13	p12	p11	p10	p9	p8	p7	p6	p5	p4	p3	p2	p1	p0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	p7		p6		p5		p4		p3		p2		p1		p0	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p3				p2				p1				p0			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 667. BBBP, DMA FIFO output bit 31 to bit 16

bpp	DMA FIFO Output Bits															
	31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
1	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
2	p0		p1		p2		p3		p4		p5		p6		p7	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p0				p1				p2				p3			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p0								p1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	-	-	-	-	-	-	-	-	23	22	21	20	19	18	17	16

Table 668. BBBP, DMA FIFO output bit15 to bit 0

bpp	DMA FIFO Output Bits															
	15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
1	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	31
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	p8		p9		p10		p11		p12		p13		p14		p15	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p4				p5				p6				p7			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0

Table 668. BBBP, DMA FIFO output bit15 to bit 0 (continued)

bpp	DMA FIFO Output Bits															
	15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
8	p2								p3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
									23	22	21	20	19	18	17	16

Table 669. LBBP, DMA FIFO output bit 31 to bit 16

bpp	DMA FIFO Output Bits															
	31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
1	p24	p25	p26	p27	p28	p29	p30	p31	p16	p17	p18	p19	p20	p21	p22	p23
2	p12		p13		p14		p15		p8		p9		p10		p11	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p6				p7				p4				p5			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p3								p2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p1															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	-	-	-	-	-	-	-	-	23	22	21	20	19	18	17	16

Table 670. LBBP, DMA FIFO output bit15 to bit 0

bpp	DMA FIFO Output Bits															
	15	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
1	p8	p9	p10	p11	p12	p13	p14	p15	p0	p1	p2	p3	p4	p5	p6	p7
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	p4		p5		p6		p7		p0		p1		p2		p3	
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
4	p2				p3				p0				p1			
	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	p1								p0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
16	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24	p0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 671. RGB Mode data format

DMA FIFO output bit	24 bit RGB	16 bit 1:5:5:5 RGB	16 bit 5:6:5 RGB	16 bit 4:4:4 RGB
[31]	0	p1- Intensity bit	p1 - Blue4	0
[30]	0	p1 - Blue4	p1 - Blue3	0
[29]	0	p1 - Blue3	p1 - Blue2	0
[28]	0	p1 - Blue2	p1 - Blue1	0
[27]	0	p1 - Blue1	p1 - Blue0	p1 - Blue3
[26]	0	p1 - Blue0	p1 - Green5	p1 - Blue2

Table 671. RGB Mode data format (continued)

DMA FIFO output bit	24 bit RGB	16 bit 1:5:5:5 RGB	16 bit 5:6:5 RGB	16 bit 4:4:4 RGB
[25]	0	p1 - Green4	p1 - Green4	p1 - Blue1
[24]	0	p1 - Green3	p1 - Green3	p1 - Blue0
[23]	p0 - Blue7	p1 - Green2	p1 - Green2	p1 - Green3
[22]	p0 - Blue6	p1 - Green1	p1 - Green1	p1 - Green2
[21]	p0 - Blue5	p1 - Green0	p1 - Green0	p1 - Green1
[20]	p0 - Blue4	p1 - Red4	p1 - Red4	p1 - Green0
[19]	p0 - Blue3	p1 - Red3	p1 - Red3	p1 - Red3
[18]	p0 - Blue2	p1 - Red2	p1 - Red2	p1 - Red2
[17]	p0 - Blue1	p1 - Red1	p1 - Red1	p1 - Red1
[16]	p0 - Blue0	p1 - Red0	p1 - Red0	p1 - Red0
[15]	p0 - Green7	p0- Intensity bit	p0 - Blue4	0
[14]	p0 - Green6	p0 - Blue4	p0 - Blue3	0
[13]	p0 - Green5	p0 - Blue3	p0 - Blue2	0
[12]	p0 - Green4	p0 - Blue2	p0 - Blue1	0
[11]	p0 - Green3	p0 - Blue1	p0 - Blue0	p0 - Blue3
[10]	p0 - Green2	p0 - Blue0	p0 - Green5	p0 - Blue2
[09]	p0 - Green1	p0 - Green4	p0 - Green4	p0 - Blue1
[08]	p0 - Green0	p0 - Green3	p0 - Green3	p0 - Blue0
[07]	p0 - Red7	p0 - Green2	p0 - Green2	p0 - Green3
[06]	p0 - Red6	p0 - Green1	p0 - Green1	p0 - Green2
[05]	p0 - Red5	p0 - Green0	p0 - Green0	p0 - Green1
[04]	p0 - Red4	p0 - Red4	p0 - Red4	p0 - Green0
[03]	p0 - Red3	p0 - Red3	p0 - Red3	p0 - Red3
[02]	p0 - Red2	p0 - Red2	p0 - Red2	p0 - Red2
[01]	p0 - Red1	p0 - Red1	p0 - Red1	p0 - Red1
[00]	p0 - Red0	p0 - Red0	p0 - Red0	p0 - Red0

33.5.5 RAM palette

The RAM-based palette is a 256 x 16 bit dual-port RAM physically structured as 128 x 32 bit. This allows two entries to be written into the palette from a single word write access. The least significant bit of the serialized pixel data is used to select between upper and lower halves of the palette RAM. Which half is selected depends on the bytes ordering mode. In little-endian mode, the LSB being set selects the upper half, but in big-endian, the lower half of the palette is selected. WinCE byte ordering is little-endian, so the former case applies.

Pixel data values can be written and verified through the AMBA AHB slave interface.

For information on the numbers of colors supported, refer to *Section 33.1.1: Number of colors supported*

The palette RAM is a dual port RAM with independent controls and addresses for each port. Port1 is used as a read/write port and is connected to the AMBA AHB slave interface. The palette entries can be written and verified through this port. Port2 is used as a read-only port and is connected to the unpacker and gray scaler.

Table 672 shows the bit representation of each word in the palette.

Table 672. Palette data storage

Bit	Name	Description
[31]	I	Intensity/unused
[31:26]	B[4:0]	Blue palette data
[25:20]	G[4:0]	Green palette data
[19:16]	R[4:0]	Red palette data
[15]	I	Intensity/unused
[14:10]	B[4:0]	Blue palette data
[09:05]	G[4:0]	Green palette data
[04:00]	R[4:0]	Red palette data

For mono STN mode only the red palette field bits [4:1] are used. However, in STN color mode the green and blue [4:1] are also used.

The red and blue pixel data can be swapped to support BGR data format using a control register bit.

In 16 and 24 bpp TFT mode, the palette is bypassed and the output of the pixel serializer is used as the TFT panel data.

33.5.6 Gray scaler

A patented gray scale algorithm drives mono and color STN panels. This provides 15 gray scales for mono displays. In the case of STN color displays, the three color components (red, green, and blue) are gray scaled simultaneously which results in 3 375 (15x15x15) colors being available. The gray scaler transforms each 4 bit gray value into a sequence of activity-per-pixel over several frames, relying to some degree on the display characteristics, to give the representation of gray scales and color.

33.5.7 Upper and lower panel formatters

Each formatter consists of three 3 bit (red, green, and blue) shift left registers. Red, green and blue pixel data bit values from the gray scaler are concurrently shifted into the respective registers. When enough data is available, a byte is constructed by multiplexing the registered data to the correct bit position to satisfy the RGB data pattern of LCD panel. The byte is transferred to the three-byte FIFO which has enough space to store eight color pixels.

33.5.8 Panel clock generator

The output of the panel clock generator block is the panel clock. This is a divided down version of CLCDCLK. It can be programmed in the range CLCDCLK/2 to CLCDCLK/33 to match the bpp data rate of the LCD panel.

33.5.9 Timing controller

The primary function of the timing controller block is to generate the horizontal and vertical timing panel signals. It also provides panel bias/enable signal. These timings are all register programmable through the AMBA AHB slave interface.

33.5.10 Interrupt generation

The CLCD provides four individually maskable interrupts and a single combined interrupt. The single combined interrupt is asserted if any of the combined interrupts are asserted and unmasked. The Interrupts are generated for the following events:

- Master bus error
- Vertical compare
- LCD next base address update
- FIFO underflow

Master bus error interrupt

The master bus error interrupt is asserted when an ERROR response is received by the master interface during a transaction with a slave. When such an error is encountered, the master interface enters an error state and remains in this state until clearance of the error has been signaled to it.

Vertical compare interrupt

The vertical compare interrupt asserts when one of four vertical display regions, selected using the LCD Control Register (*Section 33.6.10*), is reached.

- vertical synchronization
- back porch
- active video
- front porch.

LCD Next base address update interrupt

The LCD next base address update interrupt asserts when either the LCDUPBASE or LCDLPBASE values have been transferred to the LCDUPCURR or LCDLPCURR incrementers respectively. This signals to the system that it is safe to update the LCDUPBASE or the LCDLPBASE Registers with new frame base addresses if required.

FIFO underflow interrupt

The FIFO underflow interrupt asserts when internal data is requested from an empty DMA FIFO. Internally, individual upper and lower panel DMA FIFO underflow interrupt signals are generated and CLCDFUFINTR is the single combined version of these.

33.5.11 Bus architecture

The CLCD incorporates a master and a slave interface.

The master interface is directly connected to a memory controller with an AMBA AHB slave interface, while the slave interface is connected to the AMBA AHB bus.

AMBA AHB supports a wide range of on-chip bus sizes, from eight bits up to 1 024 bits. The CLCD master and slave interfaces are implemented as 32 bit data bus devices only.

33.5.12 LCD powering up and down sequences

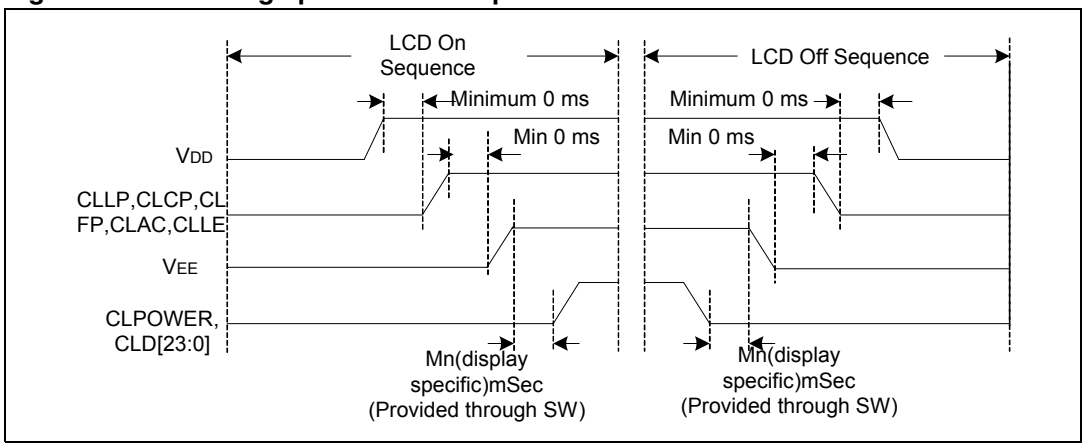
The CLCD requires the following power up sequence to be performed:

- Vdd is simultaneously applied to the SoC that contains the CLCD peripheral and panel display driver logic. The following signals are held LOW:
  - CLLP
  - CLCP
  - CLFP
  - CLAC
  - CLD[23:0]
  - CLLE.
- When Vdd is stabilized, a logic ‘1’ is written to the LcdEn bit in the LCDControl Register. (*Section 33.6.10*) This enables the following signals into their active states:
  - CLLP
  - CLCP
  - CLFP
  - CLAC
  - CLLE

  The CLD[23:0] signals remains low.
- When the signals in step 2 have stabilized, where appropriate, the contrast voltage, Vee (not controlled or supplied by the CLCD) is then applied.
- If required, you can use a software timer routine to provide the minimum display specific delay time between application of the control signals and power to the panel display. On completion of the software timer routine, power is applied to the panel by writing a logic ‘1’ to the LcdPwr bit in the LcdControl Register that, in turn, sets the CLPOWER signal HIGH and enables the CLD[23:0] signals into their active states. Normally, CLPOWER is used to gate the power to the LCD panel.

The power-down sequence is the reverse of the above four stages and you must follow it strictly, this time, writing the respective register bits with logic ‘0’.

Figure 79. Powering up and down sequences



33.6 Programming model

This section describes the programming model for color liquid crystal display controller.

33.6.1 External pin connection

For information on External pin connection see the *Chapter 5: Pin description*.

33.6.2 Register map

The CLCD can be fully configured by programming its 32 bit wide registers, which can be accessed through the AHB slave interface at the base address given in *Table 22: Reconfigurable array subsystem*.

CLCD registers can be logically arranged in three main groups:

- Configuration registers (listed in *Table 673*).
- Color palette register (listed in *Table 674*).
- Identification registers (listed in *Table 675*).

Table 673. CLCD configuration registers

Name	Offset	Type	Width (bit)	Reset value	Description
LCDTiming0	0x00	RW	32	32'h0	Horizontal axis panel control register
LCDTiming1	0x04	RW	32	32'h0	Vertical axis panel control register
LCDTiming2	0x08	RW	32	32'h0	Clock and signal polarity control register
LCDTiming3	0x0C	RW	17	17'h0	Line end control register
LCDUPBase	0x10	RW	32	32'h0	Upper panel frame base address register
LCDLPBase	0x14	RW	32	32'h0	Lower panel frame base address register
LCDMSC	0x18	RW	5	5'h0	Interrupt mask, set and clear register
LCDControl	0x1c	RW	16	16'h0	Control register
LCDRIS	0x20	RO	5	5'h0	Raw interrupt status register

Table 673. CLCD configuration registers (continued)

Name	Offset	Type	Width (bit)	Reset value	Description
LCDMIS	0x24	RO	5	5'h0	Mask interrupt status register
LCDICR	0x28	WO	5	5'h0	Interrupt clear register
LCDUPCUR	0x2C	RO	32	undefined	Upper panel current address value register
LCDLPCUR	0x30	RO	32	undefined	Lower panel current address value register

Table 674. Color palette register

Name	Offset	Type	Width (bit)	Reset value	Description
LCDPalette	0x200 to 0x3FC	RW	32	undefined	LCD color palette registers.

Table 675. Identification register

Name	Offset	Type	Width (bit)	Reset value	Description
PERIPHID0	0xFE0	RO	8	8'h10	Peripheral identification register 0
PERIPHID1	0xFE4	RO	8	8'h11	Peripheral identification register 1
PERIPHID2	0xFE8	RO	4	4'h4	Peripheral identification register 2
PERIPHID3	0xFEC	RO	8	8'h00	Peripheral identification register 3
PCELLID0	0xFF0	RO	8	8'h0D	PrimeCell identification register 0
PCELLID1	0xFF4	RO	8	8'hF0	PrimeCell identification register 1
PCELLID2	0xFF8	RO	8	8h05	PrimeCell identification register 2
PCELLID3	0xFFC	RO	8	8'hB1	PrimeCell identification register 3

33.6.3 Register description

This section describes the register of color liquid crystal display controller.

33.6.4 LCD timing 0 register

LCDTiming0 is a read/write (RW) register that controls the:

- Horizontal synchronization pulse width (HSW)
- Horizontal front porch (HFP) period
- Horizontal back porch (HBP) period
- Pixels-per-line (PPL).

Table 676. LCDTiming0 register bit assignments

Bit	Name	Reset value	Description
[31:24]	HBP	8'h0	Horizontal back porch is the number of CLCP periods between the falling edge of CLLP and the start of active data. Program with value minus 1. The 8 bit HBP field specifies the number of pixel clock periods inserted at the beginning of each line or row of pixels. After the line clock for the previous line has been deasserted, the value in HBP counts the number of pixel clocks to wait before starting the next display line. HBP can generate a delay of 1-256 pixel clock cycles.
[23:16]	HFP	8'h0	Horizontal front porch is the number of CLCP periods between the end of active data and the rising edge of CLLP. Program with value minus 1. The 8 bit HFP field sets the number of pixel clock intervals at the end of each line or row of pixels, before the LCD line clock is pulsed. When a complete line of pixels is transmitted to the LCD driver, the value in HFP counts the number of pixel clocks to wait before asserting the line clock. HFP can generate a period of 1-256 pixel clock cycles.
[15:08]	HSW	8'h0	Horizontal synchronization pulse width is the width of the CLLP signal in CLCP periods. Program with value minus 1. The 8 bit HSW field specifies the pulse width of the line clock in passive mode, or the horizontal synchronization pulse in active mode.
[07:02]	PPL	6'h0	Pixels-per-line. Actual pixels-per-line = 16 * (PPL + 1). The PPL bit field specifies the number of pixels in each line or row of the screen. PPL is a 6 bit value that represents between 16 and 1 024 PPL. PPL controls how much data is read from the DMA input buffers through to the gray scaler.
[01:00]	-	-	Reserved, do not modify, read as zero, write as zero.

Horizontal timing restrictions

DMA requests new data at the start of a horizontal display line. Some time must be allowed for the DMA transfer and for the data to propagate down the FIFO path in the LCD interface. The data path latency forces some restrictions on the usable minimum values for horizontal porch width in STN mode.

The minimum values are HSW = 2 and HBP = 2.

Single panel mode:

- HSW = 3
- HBP = 5
- HFP = 5
- Panel clock divisor (PCD) = 1 (CLCDCLK/3).

Dual panel mode:

- HSW = 3
- HBP = 5
- HFP = 5
- PCD = 5 (CLCDCLK/7).

If sufficient time is given at the start of the line (for example, setting HSW = 6, HBP = 10), data is not corrupted for PCD = 4 (minimum value).

33.6.5 LCD timing 1 register

LCDTiming1 is a read/write (RW) register that controls the:

- Number of lines per panel (LPP)
- Vertical synchronization pulse width (VSW)
- Vertical front porch (VFP) period
- Vertical back porch (VBP) period.

Table 677. LCDTiming1 register bit assignments

Bit	Name	Reset value	Description
[31:24]	VBP	8'h0	Vertical back porch is the number of inactive lines at the start of a frame, after vertical synchronization period. Program to 0 on passive displays or reduced contrast results. The 8 bit VBP field specifies the number of line clocks inserted at the beginning of each frame. The VBP count starts just after the vertical synchronization signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit field in passive mode. After this has occurred, the count value in VBP sets the number of line clock periods inserted before the next frame. VBP generates from 0-255 extra line clock cycles.
[23:16]	VFP	8'h0	Vertical front porch is the number of inactive lines at the end of frame, before vertical synchronization period. Program to 0 on passive displays or reduced contrast results. The 8 bit VFP field specifies the number of line clocks to insert at the end of each frame. When a complete frame of pixels is transmitted to the LCD display, the value in VFP is used to count the number of line clock periods to wait. After the count has elapsed the vertical synchronization signal, CLFP, is asserted in active mode, or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates from 0–255 line clock cycles.

Table 677. LCDTiming1 register bit assignments (continued)

Bit	Name	Reset value	Description
[15:10]	VSW	6'h0	Vertical synchronization pulse width is the number of horizontal synchronization lines. Must be small (for example, program to zero) for passive STN LCDs. Program to the number of lines required minus one. The higher the value the worse the contrast on STN LCDs. The 6 bit VSW field specifies the pulse width of the vertical synchronization pulse. The register is programmed with the number of line clocks in VSync minus one. Number of horizontal synchronization lines. Must be small (for example, program to 0) for passive STN LCDs. Program to the number of lines required minus 1. The higher the value the worse the contrast on STN LCDs.
[09:00]	LPP	10'h0	Lines per panel is the number of active lines per screen. Program to number of lines required minus 1. The LPP field specifies the total number of lines or rows on the LCD panel being controlled. LPP is a 10 bit value that allows 1-1 024 lines. The register is programmed with the number of lines per LCD panel minus 1. For dual panel displays this register is programmed with the number of lines on each of the upper and lower panels.

33.6.6 LCD timing 2 register

LCDTiming2 is a read/write (RW) register that controls the CLCD timing. *Table 678* shows the bit assignments for the LCDTiming2 register.

Table 678. LCDTiming2 register bit assignments

Bit	Name	Reset value	Description
[31:27]	PCD_HI	5'h0	Upper five bits of panel clock divisor.a The ten bit PCD field, comprising PCD_HI and PCD_LO (bits [4:0]), is used to derive the LCD panel clock frequency CLCP from the CLCDCLK frequency: CLCP = CLCDCLK/(PCD+2). For mono STN displays with a four or eight bit interface, the panel clock is a factor of four and eight down on the actual individual pixel clock rate. For color STN displays, 2 2/3 pixels are output per CLCP cycle, therefore the panel clock is 0.375 times. For TFT displays the pixel clock divider can be bypassed by setting the LCDTiming2[26] BCD bit.
[26]	BCD	1'h0	Bypass pixel clock divider. Setting this to 1 bypasses the pixel clock divider logic. This is mainly used for TFT displays.

Table 678. LCDTiming2 register bit assignments (continued)

Bit	Name	Reset value	Description
[25:16]	CPL	10'h0	Clocks per line. This field specifies the number of actual CLCP clocks to the LCD panel on each line. This is the number of PPL divided by 1 for TFT, 4 or 8 for mono passive, or 2 2/3 for color passive, minus one. This must be correctly programmed in addition to PPL for the LCD controller to work correctly.
[15]	-	-	Reserved, do not modify, read as zero, write as zero.
[14]	IEO	1'h0	Invert output enable: 1'b0 = CLAC output pin is active HIGH in TFT mode 1'b1 = CLAC output pin is active LOW in TFT mode. The invert output enable (IOE) bit is used to select the active polarity of the output enable signal in TFT mode. In this mode, the CLAC pin is used as an enable that indicates to the LCD panel when valid display data is available. In active display mode, data is driven onto the LCD data lines at the programmed edge of CLCP when CLAC is in its active state.
[13]	IPC	1'h0	Invert panel clock: 1'b0 = Data is driven on the LCDs data lines on the rising-edge of CLCP 1'b1 = Data is driven on the LCDs data lines on the falling-edge of CLCP. The IPC bit is used to select the edge of the panel clock on which pixel data is driven out onto the LCD data lines.
[12]	IHS	1'h0	Invert horizontal synchronization: 1'b0 = CLLP pin is active HIGH and inactive LOW 1'b1 = CLLP pin is active LOW and inactive HIGH. The invert HSync (IHS) bit is used to invert the polarity of the CLLP signal.
[11]	IVS	1'h0	Invert vertical synchronization: 1'b0 = CLFP pin is active HIGH and inactive LOW 1'b1 = CLFP pin is active LOW and inactive HIGH. The invert VSync (IVS) bit is used to invert the polarity of the CLFP signal.
[10:06]	ACB	5'h0	AC bias pin frequency. The AC bias pin frequency is only applicable to STN displays, which require the pixel voltage polarity to be periodically reversed to prevent damage due to DC charge accumulation. Program this field with the required value minus 1 to apply the number of line clocks between each toggle of the AC bias pin, CLAC. This field has no effect if the CLCD is operating in TFT mode when the CLAC pin is used as a data enable signal.
[05]	CLKSEL	1'h0	This bit drives the CLCDCLKSEL signal that is used as the select signal for the external CLCDCLK clock multiplexer. 1'b0 - HCLK 1'b1 - Other Clock

Table 678. LCDTiming2 register bit assignments (continued)

Bit	Name	Reset value	Description
[04:00]	PCD_LO	5'h0	Lower five bits of panel clock divisor.(1) The ten bit PCD field, comprising PCD_HI (bits [31:27]) and PCD_LO, is used to derive the LCD panel clock frequency CLCP from the CLCDCLK frequency, CLCP = CLCDCLK/(PCD+2). For mono STN displays with a four or eight bit interface, the panel clock is a factor of four and eight down on the actual individual pixel clock rate. For color STN displays, 2 2/3 pixels are output per CLCP cycle, so the panel clock is 0.375 times. You can bypass the pixel clock divider for TFT displays by setting the LCDTiming2[26] BCD bit.

1. The data path latency forces some restrictions on the usable minimum values for the panel clock divider in STN modes:

Single-panel color mode: PCD = 1 (CLCP = CLCDCLK/3)

Dual-panel color mode: PCD = 4 (CLCP = CLCDCLK/6)

Single-panel mono 4 bit interface mode: PCD = 2 (CLCP = CLCDCLK/4)

Dual-panel mono 4 bit interface mode: PCD = 6 (CLCP = CLCDCLK/8)

Single- panel mono 8 bit interface mode: PCD = 6 (CLCP = CLCDCLK/8)

Dual-panel mono 8 bit interface mode: PCD = 14 (CLCP = CLCDCLK/16)

33.6.7 LCD timing 3 register

LCDTiming3 is a read/write (RW) register that controls the enabling of line-end signal CLLE. When enabled, a positive pulse, four CLCDCLK periods wide, is output on CLLE after a programmed delay set by the LED bits. If the line-end signal is disabled then it is held permanently LOW.

Table 679. LCDTiming3 register bit assignments

Bit	Name	Reset value	Description
[31:17]	-	-	Reserved, do not modify, read as zero, write as zero.
[16]	LEE	1'h0	LCD Line end enable: 1'b0 = CLLE disabled (held LOW) 1'b1 = CLLE signal active.
[15:07]	-	-	Reserved, do not modify, read as zero, write as zero
[06:00]	LED	7'h0	Line-end signal delay from the rising-edge of the last panel clock, CLCP. Program with number of CLCDCLK clock periods minus 1.

33.6.8 LCDUPBASE and LCPLPBASE registers

LCDUPBASE and LCDLPBASE are the color LCD DMA frame address registers.

They are read/write registers used to program the base address of the frame buffer.

LCDUPBASE is used for:

- TFT displays
- Single panel STN displays
- The upper panel of dual panel STN displays.

LCDLPBASE is used for the lower panel of dual panel STN displays.

You must initialize LCDUPBASE (and LCDLPBASE for dual panels) before enabling the CLCD. You can change the value mid-frame to enable double-buffered video displays to be created. These registers are copied to the corresponding current registers at each LCD vertical synchronization. This event causes the LNBU bit and an optional interrupt to be generated. You can use the interrupt to reprogram the base address when generating double-buffered video.

Table 680. LCDUPBASE register bit assignments

Bit	Name	Reset value	Description
[31:02]	LCDUPBASE	29'h0	LCD upper panel base address. This is the start address of the upper panel frame data in memory and is word aligned.
[01:00]	-	-	Reserved, do not modify, read as zero, write as zero.

Table 681. LCDLPBASE register bit assignments

Bit	Name	Reset value	Description
[31:02]	LCDLPBASE	29'h0	LCD lower panel base address. This is the start address of the lower panel frame data in memory and is word aligned.
[01:00]	-	-	Reserved, do not modify, read as zero, write as zero.

33.6.9 LCDIMSC register

LCDIMSC is the interrupt mask set/clear register. Setting bits in this register enables the corresponding raw interrupt LCDRIS bit values to be passed to the LCDMIS,

Table 682. LCDIMSC register bit assignments

Bit	Name	Reset value	Description
[31:05]			
[04]	MBERRINTRENB	1'h0	AHB master error interrupt enable
[03]	VCOMPINTRENB	1'h0	Vertical compare interrupt enable
[02]	LNBUINTRENB	1'h0	Next base update interrupt enable

Table 682. LCDIMSC register bit assignments (continued)

Bit	Name	Reset value	Description
[01]	FUFINTRENB	1'h0	FIFO underflow interrupt enable
[00]	-	-	Reserved, do not modify, read as zero, write as zero

33.6.10 LCD control register

LCDControl is the control register. It is a read/write (RW) register that controls the mode in which the CLCD operates.

Table 683. LCDControl register bit assignments

Bit	Name	Reset value	Description
[31:17]	-	-	Reserved, do not modify, read as zero, write as zero
[16]	WATERMARK	1'h0	LCD DMA FIFO Watermark level: 1'b0 = HBUSREQM is raised when either of the two DMA FIFOs have four or more empty locations 1'b1 = HBUSREQM is raised when either of the DMA FIFOs have eight or more empty locations.
[15:14]	-	-	Reserved, do not modify, read as zero, write as zero
[13:12]	LCDVCOMP	2'h0	Generate interrupt at: – 2'b00 = Start of vertical synchronization – 2'b01 = Start of back porch – 2'b10 = Start of active video – 2'b11 = Start of front porch
[11]	LCDPWR	1'h0	LCD Power enable: 1'b0 = Power not gated through to LCD panel and CLD[23:0] signals disabled. (Held low) 1'b1 = Power gated through to LCD panel and CLD[23:0] signals enabled. (Active)
[10]	BEPO	1'h0	Big-endian pixel order within a byte: 1'b0 = little-endian pixel ordering within a byte 1'b1 = big-endian pixel order within a byte The BEPO bit selects between little and big-endian pixel packing for 1, 2 and 4 bpp display mode. It has no effect on 8 and 16 bpp pixel format. See pixel serializer table for more information on the data format.
[09]	BEBO	1'h0	Big-endian byte order: 1'b0 = little-endian byte order 1'b1 = big-endian byte order

Table 683. LCDControl register bit assignments (continued)

Bit	Name	Reset value	Description
[08]	BGR	1'h0	RGB of BGR format selection: 1'b0 = RGB normal output 1'b1 = BGR red and blue swapped.
[07]	LCDDUAL	1'h0	LCD interface is dual panel STN: 1'b0 = single panel LCD is in use 1'b1 = dual panel LCD is in µs
[06]	LCDMONO8	1'h0	Monochrome LCD has an 8 bits interface. This bit controls whether monochrome STN LCD uses a 4 or 8 bits parallel interface: 1'b0 = mono LCD uses 4 bits interface 1'b1 = mono LCD uses 8 bits interface LcdMono8 has no meaning in other modes and must be programmed to 0.
[05]	LCDTFT	1'h0	LCD is TFT: 1'b0 = LCD is an STN display, use gray scaler 1'b1 = LCD is TFT, do not use gray scaler
[04]	LCDBW	1'h0	STN LCD is monochrome (black and white): 1'b0 = STN LCD is color 1'b1 = STN LCD is monochrome This bit has no meaning in TFT mode.
[03:01]	LCDBPP	3'h0	LCD bits per pixel: – 3'b000 = 1 bpp – 3'b001 = 2 bpp – 3'b010 = 4 bpp – 3'b011 = 8 bpp – 3'b100 = 16 bpp – 3'b101 = 24 bpp – 3'b110 = reserved – 3'b111 = reserved
[00]	LCDEN	1'h0	LCD controller enable bit: 1'b0 = CLLP, CLCP, CLFP, CLAC, and CLLE disabled (held LOW) 1'b1 = CLLP, CLCP, CLFP, CLAC, and CLLE enabled (active).

33.6.11 LCDRIS register

LCDRIS is a read-only (RO) register. On a read it returns five bits that can generate interrupts when set.

Table 684. LCDRIS register bit assignments

Bit	Name	Reset value	Description
[13:05]			
[04]	MBERROR	1'h0	AHB bus master error status. Set when the AHB master encounters a bus error response from a slave.
[03]	VCOMP	1'h0	Vertical compare. Set when one of the four vertical regions, selected through the LCD control register, is reached.
[02]	LNBU	1'h0	LCD next address base update, mode dependent, set when the current base address registers have been successfully updated by the next address registers. Signifies that a new next address can be loaded if double buffering is in use.
[01]	FUF	1'h0	FIFO underflow, set when either the upper or lower DMA FIFOs have been read accessed hen empty causing an underflow condition to occur.
[00]	-	-	Reserved, read as zero

33.6.12 LCDMIS register

LCDMIS is a read-only (RO) register. It is a bit-by-bit logical AND of the LCDRIS register and the LCDIMSC register. Interrupt lines correspond to each interrupt. A logical OR of all interrupts is provided to the system interrupt controller.

Table 685. LCDMIS register bit assignments

Bit	Name	Reset value	Description
[31:05]	-	-	Reserved, read as zero
[04]	MBERRORINTR	1'h0	AHB Master errors interrupt status bit.
[03]	VCOMPINTR	1'h0	Vertical compare interrupt status bit.
[02]	LNBUINTR	1'h0	LCD next base address update interrupt status bit.
[01]	FUFINTR	1'h0	FIFO underflows interrupt status bit.
[00]	-	-	Reserved, read as zero.

33.6.13 LCDICR register

The LCDICR is a write-only (WO) register. Writing a logic 1 to the relevant bit clears the corresponding interrupt.

Table 686. LCDICR register bit assignments

Bit	Name	Reset value	Description
[31:05]	-	-	Reserved, do not modify, write as zero
[04]	MBERROR	1'h0	Clear AHB Master errors interrupt.
[03]	VCOMP	1'h0	Clear vertical compare interrupt.
[02]	LNBU	1'h0	Clear LCD next base address update interrupt.
[01]	FUF	1'h0	Clear FIFO underflows interrupt.
[00]	-	-	Reserved, do not modify, write as zero

33.6.14 LCDUPCURR and LCDLPCURR registers

LCDUPCURR and LCDLPCURR are read-only (RO) registers that contain an approximate value of the upper and lower panel data DMA addresses when read. The registers can change at any time and therefore can only be used as a mechanism for coarse delay.

Table 687. LCDUPCURR register bit assignments

Bit	Name	Reset value	Description
[31:00]	LCDUPCURR	32'h0	Contains the approximate current upper panel data DMA address.

Table 688. LCDLPCURR register bit assignments

Bit	Name	Reset value	Description
[31:00]	LCDLPCURR	32'h0	Contains the approximate current lower panel data DMA address.

33.6.15 LCDPalette register

The LCDPalette register contains 256 palette entries organized as 128 locations of two entries per word. Only TFT displays use all of the palette entry bits.

Each word location contains two palette entries. This means that 128 word locations are used for the palette. When configured for little-endian byte ordering, bits [15:00] are the lower numbered palette entry and bits [31:16] are the higher numbered palette entry.

When configured for big-endian byte ordering this is reversed because bits [31:16] are the low numbered palette entry and bits [15:00] are the high numbered entry.

Table 689. LCDPalette register bit assignments

Bit	Name	Reset value	Description
[31]	I	-	Intensity or unused.
[30:26]	B[4:0]	-	Blue palette data.
[25:21]	G[4:0]	-	Green palette data.

Table 689. LCDPalette register bit assignments (continued)

Bit	Name	Reset value	Description
[20:16]	R[4:0]	-	Red palette data.
[15]	I	-	Intensity bit. Can be used as the LSB of the R, G, and B inputs to a 6:6:6 TFT display, doubling the number of colors to 64K, where each color has two different intensities.
[14:10]	B[4:0]	-	Blue palette data.
[09:05]	G[4:0]	-	Green palette data.
[04:00]	R[4:0]	-	Red palette data. For STN displays, only the four MSBs (bits [4:1]) are used. For monochrome displays only the red palette data is used. All of the palette registers have the same bit fields.

33.6.16 PHERIPHID0-3 registers

The CLCDPERIPHID0-3 registers are four 8 bit registers, that span address locations 0xFE0 to 0xFEC. The registers can conceptually be treated as a single 32 bit register. The read-only (RO) registers provide the following options of the peripheral:

PartNumber[11:0] This is used to identify the peripheral. The product code 0x10 is used for the PrimeCell CLCD.

DesignerID[19:12] This is the identification of the designer. ARM limited is 0x41 (ASCII A).

Revision[23:20] This is the revision number of the peripheral. The revision number starts from 0 and is revision dependent.

Configuration[31:24] This is the configuration option of the peripheral. The configuration value is 0.

The PHERIPHID0-3 registers are hard-coded and the fields in the register determine the reset value.

Table 690. PHERIPHID0 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PartNumber0	8'h10	These bits read back as 0x10

Table 691. PHERIPHID1 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved, read as zero
[07:04]	Designer0	4'h1	These bits read back as 0x1
[03:00]	PartNumber1	4'h1	These bits read back as 0x1

Table 692. PHERIPHID2 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved, read as zero
[07:04]	Revision	4'h0	These bits read back as 0x0
[03:00]	Designer1	4'h4	These bits read back as 0x4

Table 693. PHERIPHID3 register bit assignments

Bit	Name	Reset value	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	Configuration	8'h0	These bits read back as 0x00

33.6.17 PCELLIDID0-3 registers

The PCELLIDID0-3 Registers are four 8 bit registers, that span address locations 0xFF0-0xFFC. The registers can conceptually be treated as a 32 bit register. The register is used as a standard cross-peripheral identification system.
The PCELLIDID0 Registers are hard-coded and the fields in the register determine the reset value.

Table 694. PCCELLIDIDO register bit assignments

Bit	Name	Reset Value	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLIDID0	8'h0D	These bits read back as 0x0D

Table 695: PCELLIDID1 register bit assignments

Bit	Name	Reset Value	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLIDID1	8'hF0	These bits read as 0xF0

Table 696. PCELLIDID2 register bit assignments

Bit	Name	Reset Value	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLIDID2	8'h05	These bits read back as 0x05

Table 697: PCELLIDID3 register bit assignments

Bit	Name	Reset Value	Description
[31:08]	-	-	Reserved, read as zero
[07:00]	PCELLIDID3	8'hB1	These bits read back as 0xB1

33.7 Interrupts

There are five interrupts generated by the CLCD. The following are individual maskable active HIGH interrupts:

- CLCDMBEINTR
- CLCDVCOMPINTR
- CLCDLNBUINTR
- CLCDFUFINTR

The output is a combined to give a single interrupt CLCDINTR. Each of the four individual maskable interrupts is enabled or disabled by changing the mask bits in the LCDIMSC Register.

The status of the individual interrupt sources can be read from the LCDRIS Register.

33.7.1 CLCDMBEINTR

The master bus error interrupt is asserted when an ERROR response is received by the master interface during a transaction with a slave. When such an error is encountered, the master interface enters an error state and remains in this state until clearance of the error has been signalled to it. On completion of the respective interrupt service routine, the master bus error interrupt can be cleared by writing a logic '1' to the MBERROR bit within the LCDICR Register. This action releases the master interface from its ERROR state to the start of FRAME state, enabling a fresh frame of data display to be initiated.

33.7.2 LCDVCOMPINTR

The vertical compare interrupt is asserted when one of four vertical display regions, elected using the Control Register, is reached. The interrupt can be made to occur at the start of:

- vertical synchronization
- back porch
- active video
- front porch.

It is possible to clear the interrupt by writing a logic '1' to the VComp bit in the LCDICR Register.

33.7.3 CLCDLNBUINTR

The LCD next base address update interrupt is asserted when either the LCDUPBASE or the LCDLPBASE values are transferred to the LCDUPCURR or LCDLPCURR incrementors respectively. This signals to the system that it is safe to update the LCDUPBASE or the LCDLPBASE Registers with new frame base addresses if required.

It is possible to clear the interrupt by writing a logic ‘1’ to the LNBU bit in the LCDICR Register.

33.7.4 CLCDFUFINTR

The FIFO underflow interrupt is asserted when internal data is requested from an empty DMA FIFO. Internally, individual upper and lower panel DMA FIFO underflow interrupt signals are generated and CLCDFUFINTR is the single combined version of these.

It is possible to clear the interrupt by writing a logic ‘1’ to the FUF bit in the LCDICR Register.

33.7.5 LCD powering up and powering down sequence support

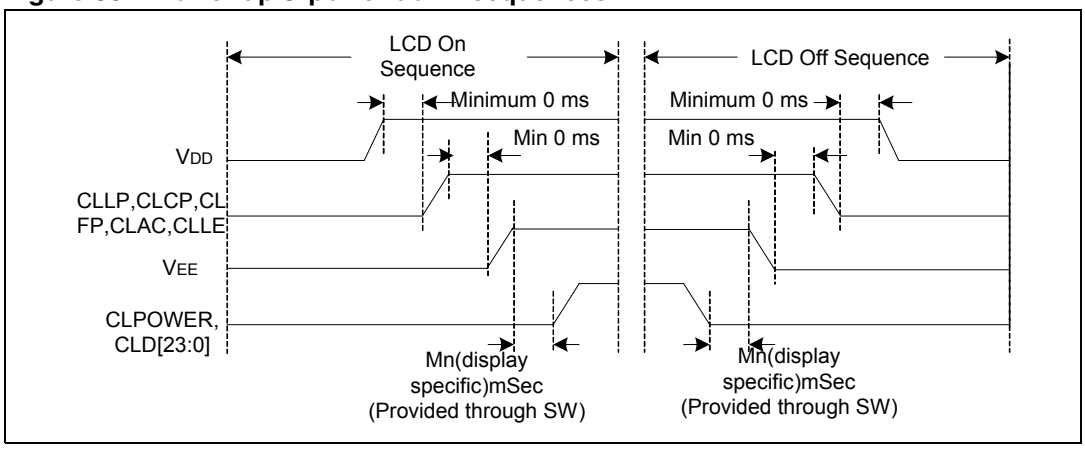
The PrimeCell CLCD (PL110) enables the following power up sequence:

1. Vdd is simultaneously applied to the SoC that contains the CLCD and panel display driver logic. The signals **CLLP**, **CLCP**, **CLFP**, **CLAC**, **CLD[23:0]**, and **CLLE** are held LOW.
2. When Vdd is stabilized, a logic ‘1’ is written to the LcdEn bit in the LCDControl Register. This puts the signals **CLLP**, **CLCP**, **CLFP**, **CLAC**, and **CLLE** into their active states but the **CLD[23:0]** signals remain LOW.
3. When the signals in Step 2 have stabilized, where appropriate, the contrast voltage Vee (this is not controlled or supplied by the CLCD) is then applied.
4. You can use a software timer routine, if required, to provide the minimum display specific delay time between application of the control signals and power to the panel display. On completion of the software timer routine, power is applied to the panel by writing a logic ‘1’ to the LcdPwr bit within the LcdControl Register which, in turn, sets the **CLPOWER** signal HIGH and puts the **CLD[23:0]** signals into their active state. The **CLPOWER** signal is expected to be used to gate the power to the LCD panel.

The power down sequence is the reverse of the above four stages and must be strictly followed, this time write the relevant register bits with logic ‘0’.

The power up and power down sequences are shown in *Figure 80: Power up & power down sequences*

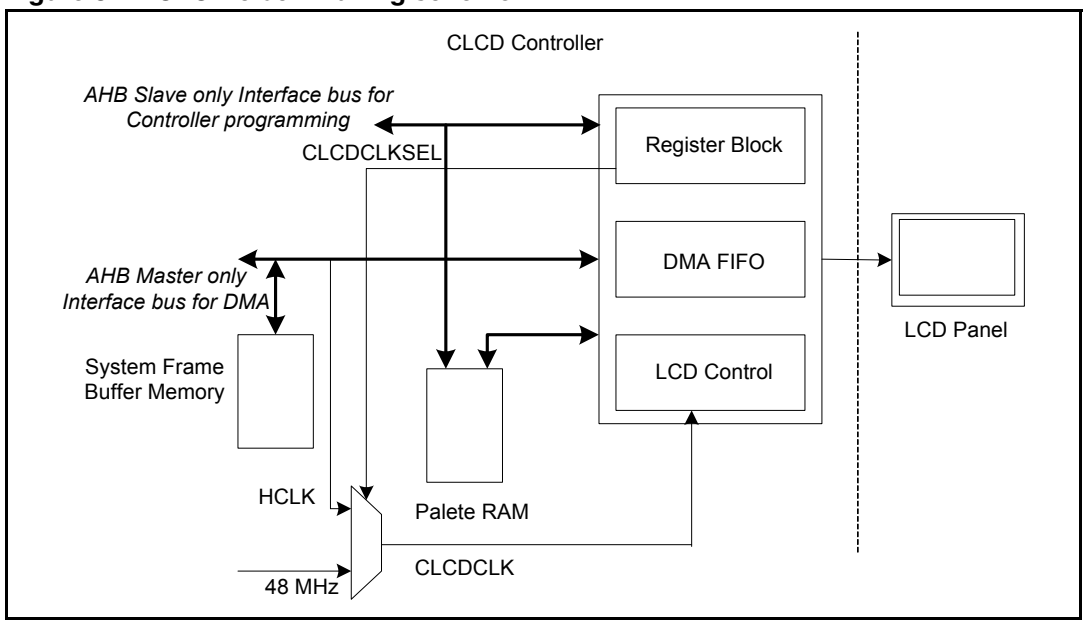
Figure 80. Power up & power down sequences



33.8 CLCD clock scheme

The CLCDCLK for the CLCD IP can be input from 2 sources. LCD Timing Register2 selects between HCLK and 48MHz clock.

Figure 81. CLCD clock muxing scheme



34 RS_Telecom IP

This section describes the telecom module.

34.1 Overview

Within the Reconfigurable Array Subsystem, SPEAr300 provides a Telecom IP which has an AMBA 2 compatible interface and supports various telephony, audio applications.

34.2 Main features

- Voice transmission for normal telecommunications (TDM Interface).
- Voice transmission for high quality VoIP application (I2S Interface).
- TDM interface with 512 timeslots and expansion of 16 bufferization channels.
- 32ms bufferization for 16 channels (of 4bytes each)
- 8 programmable sychronization signals for codecs
- 18 GPIOs for SLIC management
- Supports master and slave mode of operation
- Programmable clock and synchronization signals generation in master mode
- Clock and synchornization signals recovery in slave mode
- Multiple clock source options with master and slave mode functionality
- Expansion of SPI and I2C interface lines in fixed part of SPEAr300
- Interrupt control management
- 1 bit DAC
- Compatible to AMBA2 (Advanced Microcontroller Bus Architecture)

Note: All or some of these features are available in different RAS configuration modes (see Section 5.3).

34.3 Pin list

This section describes the functionality of the pin.

Table 698 lists all the input/output signals of the telecom IP.

Table 698. Telecom block pin signals

Signal	Direction	Description
TDM CLK	INOUT	TDM Clock
TDM DIN	IN	TDM Input
TDM DOUT	OUT	TDM Output
I2S CLK	INOUT	I2S Clock
I2S LRCK	INOUT	I2S Synchronizer signal
I2S DIN	IN	I2S Input

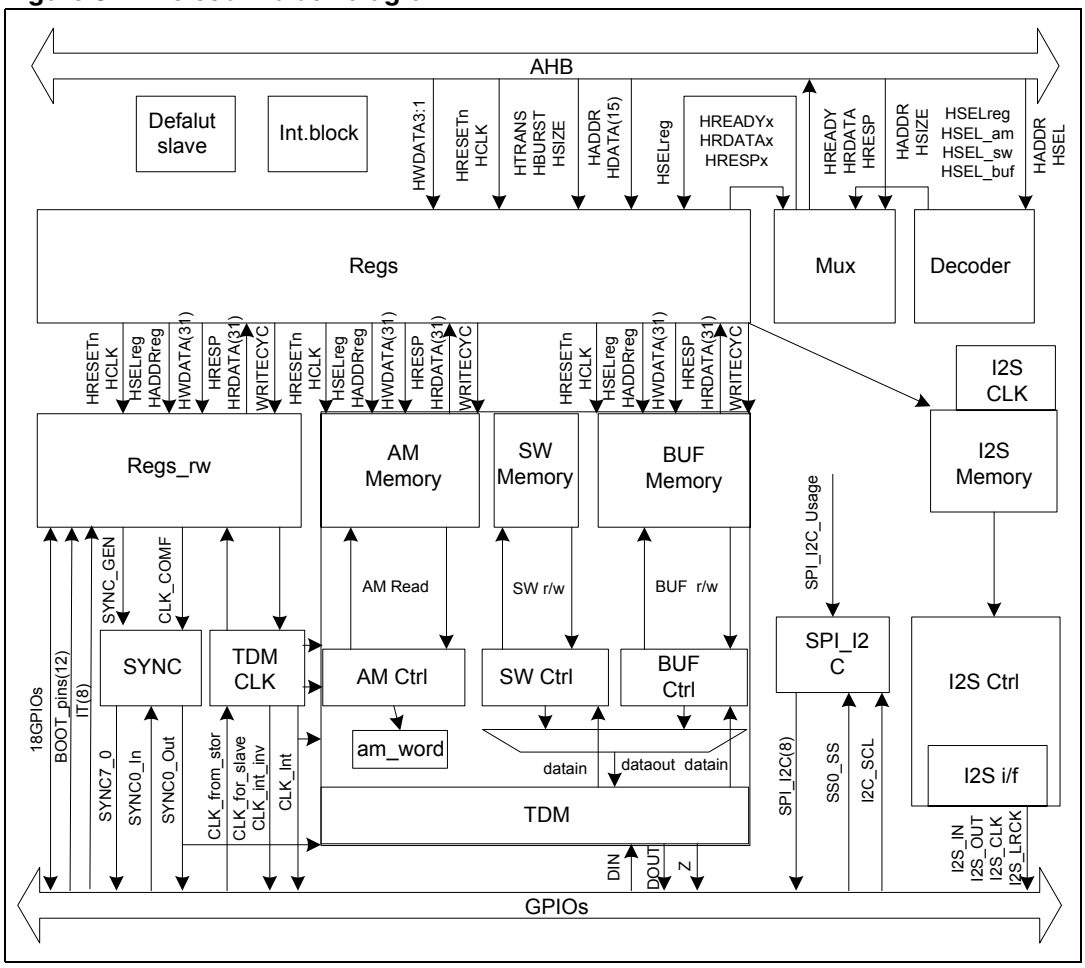
Table 698. Telecom block pin signals (continued)

Signal	Direction	Description
I2S DOUT	OUT	I2S Output
SYNC (7:0)	SYNC(7:1) -OUT	Synchronizer Signals
	SYNC(0) -INOUT	
SPI_I2C (7:0)	OUT	SPI Chip Select /I2C Clock
IT (7:0)	IN	Interrupt bus
G10 (10:0)	INOUT	GPIOs
G8 (7:0)	INOUT	GPIOs

34.4 Functional overview

The block diagram of Telecom module is shown below in *Figure 82*.

Figure 82. Telecom block diagram



The AHB side interfaces the TDM block with the ARM processor. The GPIO side is the interface to outside through PL_GPIOs.

The various sub blocks of Telecom IP are describes in the following sections:-

34.4.1 Regs and regs_rw blocks

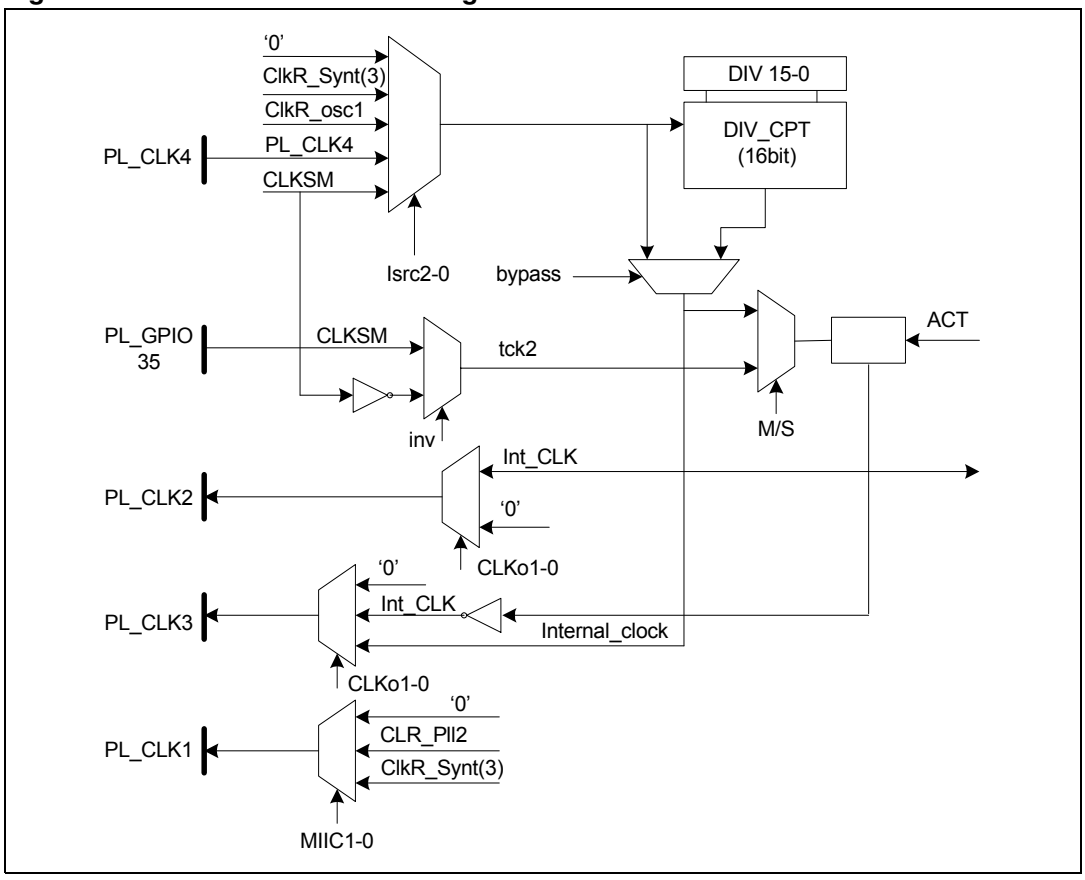
These blocks contain the set of telecom programming registers which are described in detail in the Programmer's Model *Section 34.5*

34.4.2 TDM clock block

The TDM clock block generates the clock in Master mode and recovers the clock in slave mode. This clock internally is named int_CLK and is used by all the blocks relative to TDM.

The block diagram of this cell is presented here below.

Figure 83. TDM clock cell block diagram



Externally the TDM clock block is connected to the four customization clock pins (PL_CLK1, PL_CLK2, PL_CLK3 & PL_CLK4) of the device plus the CLK pin (PL_GPIO35) that is used in slave mode.

- PL_CLK1 outputs a clock generated either from pll2 or ClkR_synt(3) {frequency synthesizer output 3}.
- PL_CLK2 outputs the internal int_CLK signal
- PL_CLK3 outputs either the inverted int_CLK signal, or the clock source selected by the block. This feature allows delivering a clock to an external master device.
- PL_CLK4 is an input targeted to receive a clock from an oscillator.

Internally, the master mode can select between four clock sources, then divide it or use as it is (for configuration please refer to Table 704: TDM_conf register (Offset 0x04)). The 16 bit divider allows division from 2 to 131072.

- CLKSM is the slave mode input clock signal on the PL_GPIO35 pin.
- ClkR_osci1 is the master clock crystal frequency 24 MHz.

34.4.3 TDM synchro block

Eight synchronization signals SYNC0-SYNC7 are generated by the TDM synchro block. The SYNC0 signal is bidirectional - generated in master mode and recovered in slave mode.

The reference clock for the TDM synchro block is the int_CLK clock delivered by the TDM clock block.

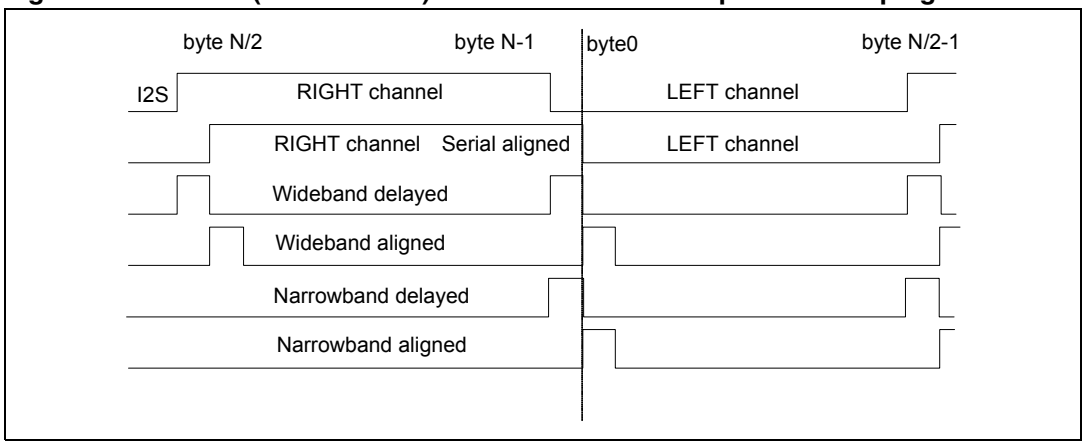
- In slave mode, SYNC0 signal is sampled at CLKSM rate.
- In master mode, SYNC0 is generated at int_CLK rate.
- All other SYNCx signals are generated at int_CLK rate.

Externally, the TDM synchro block is connected to SYNC0 to SYNC7 pins.

a) SYNC1 to SYNC3 (and SYNC0, if used master mode) are generated from an internal counter and can take a predefined shape. They can all be delayed with respect to each other by 8, 16 or 32 bits (counted on int_CLK clocks).

Either of the following shapes can be generated for any of them:

Figure 84. SYNC0 (slave/master) and SYNC1 to SYNC3 possible shaping



b) SYNC4 to SYNC7 are user defined. They are programmable bit by bit through a memory (the sync memory). SYNC memory is 512 words wide (32 bits). Each word defines the SYNCx signals for one timeslot.

Word0 defines the four waveforms for the first height bits of the frame.

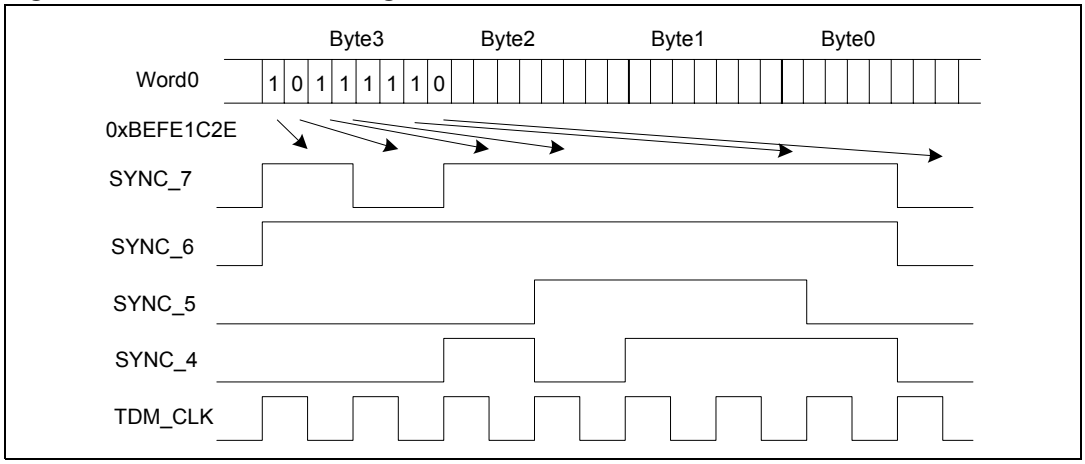
In each word,

- Byte0 defines the waveform for SYNC4,
- Byte1 defines the waveform for SYNC5,
- Byte2 defines the waveform for SYNC6 and
- Byte3 defines the waveform for SYNC7.

In each byte, the MSB is played first.

Here below is shown the generation of SYNC4-SYNC7 for the first height bits of the frame; Successive bits are generated by successive words.

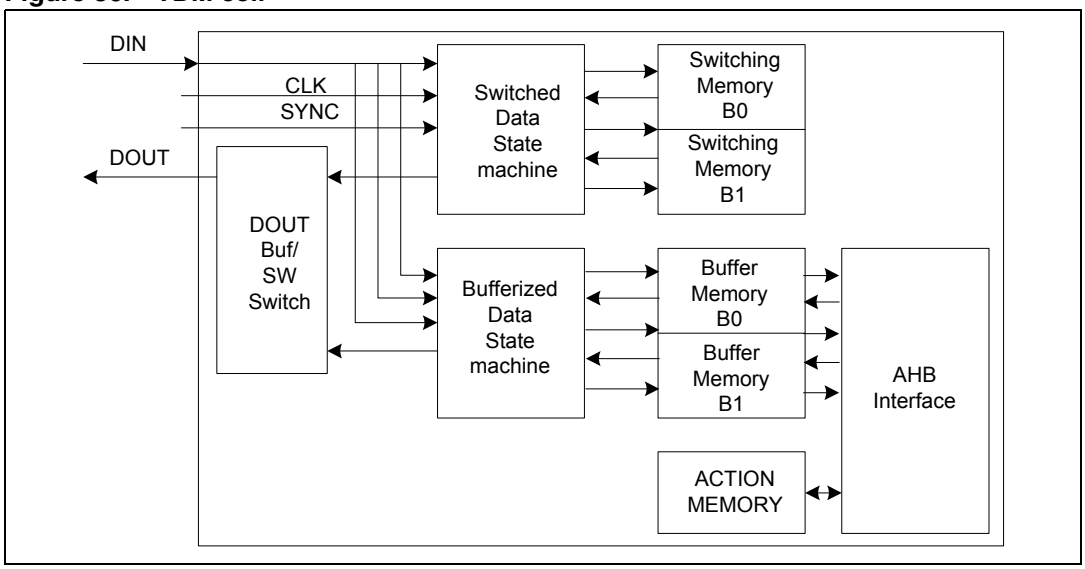
Figure 85. SYNC4 to SYNC7 generation



34.4.4 TDM block

The TDM block includes the switching and bufferization functions of the Telecom IP.

Figure 86. TDM cell



The TDM port supports synchronous data transfer. Each frame has a programmable length and can support up to 512 timeslots in full duplex. Data is synchronized by the bit clock and frame synchronization delimits frames that contain a programmable number of timeslots (please refer to *Table 716: TDM_timeselot_NBR register (Offset 0x38)*). Each timeslot (8 bit data) can be used for switching or bufferization. Action memory indicates what to do with each timeslot. It is a dual port memory connected both to the TDM module and the AHB Interface.

- Switching means that an output timeslot (a timeslot played on the DOUT pin) contains the data received during the previous frame on an input timeslot (a timeslot received on

the DIN pin). Switching data is stored in a single port Switching Memory without access from AHB.

- Bufferization means that data from DIN pin (one to four timeslots per channel per frame) is stored in the buffer memory during a programmable number of frames (please refer to *Table 717: TDM_Frame_NBR register (Offset 0x3C)*) and data from buffer memory is played at the same time. The buffer memory is shared between the module and the AHB and split into two banks. Up to 16 channels (buffers) can be stored/played from different locations (dynamically partitioned) in the buffer memory bank. Using 16 channels of 4 bytes per frame allows 30ms data storage for each channels as required by VoIP applications.

34.4.5 Action memory

Action memory is the heart of the state machine sending and recovering data from TDM bus. Each word contains the information of what to do with the relative Timeslot (byte). Action memory consists of 512 words of 32 bits each. Word0 informs about timeslot0 and similarly word 511 informs about timeslot 512.

The options that can be chosen for the timeslot include switching Timeslot x of DIN to Timeslot y of DOUT, bufferizing DIN and/or playing a buffer on DOUT or doing nothing.

The TDM behavior is based on three memories:

- The action memory for information,
- The switching memory for switching (target is small PABx),
- The buffer memory for bufferization (target is VoIP voice buffers).

According to the content of the action memory, an incoming sample may or may not be stored. Destination can be either the switching memory, or the buffer memory, or both.

In the reverse direction, the TDM output can be in low impedance or high impedance on a timeslot basis. When in low impedance, DOUT can play the data from the switching memory or from the buffer memory according to the information in the action memory.

Switching memory is only accessible by the TDM state machine on the TDM side, both in read and write mode. The aim of the switching memory is to switch an input timeslot of a frame to an output timeslot during the next frame. Please read *Section 34.4.6: Switching memory* for full description.

Buffer memory is organized in two banks (16KB each). One bank can be accessed by the TDM state machine while the other bank is available for the AHB interface. Each bank is split into locations for N channels (with 1=< N =<16). The bank that is available for the TDM side will play data from the all the channels to pre-defined timeslots at each frame and store the incoming data of the same timeslot in the same place. Each time the banks are switched, an interrupt request can be initialized by the RAS. Please read *Section 34.4.7: Buffer memory* for full description.

34.4.6 Switching memory

Switching feature uses total memory of 1024 locations, each 8 bit wide. Switching means that all the input timeslots of a frame received on the TDM are stored in the memory and at the next frame, this data will be sent on the timeslot that is required to be switched.

Consider an example of switching: TS3 is switched with TS511. It means that during TS3, the TS3 content received on the DIN pin is stored at the 3rd byte of the storage part of the switching memory while TS511 of the previous frame (511st byte of the reading part of the

switching memory) is played on the DOUT pin. During TS511, TS511 received on the DIN pin is stored at the 511st byte of the storage part of the switching memory while TS3 of the previous frame (3rd byte of the reading part of the switching memory) is played on the TDM output, allowing a phone call to be established between the caller connected on TS3 and the one connected on TS511.

Switching introduces a delay of one frame between the reception of a data on the TDM input and its transmission on the TDM output.

The 1024x8 memory is split in two banks of 512 8 bit wide locations. The address MSB bit informs which bank is used to store and to play data.

The processor can not access this memory. Anyway, to inform the processor about the bank position, the MSB bit can be read in the *Table 718: TDM_SYNC_GEN register (Offset 0x40)*.

When the last timeslot of a frame must be switched with the first timeslot of the next one, (TS511 <->TS0 for instance), there is no time to store the data before reading it for playing. A special bit in the action memory content will inform the state machine about this. Keeping LTSS bit as '1' will directly send the last sample to the output shift register. The sample will anyway be stored in the memory. For any timeslot, the action memory informs whether the sample must be stored in the switching memory, the buffer memory, both or none.

Figure 87. Illustration for TDM switching

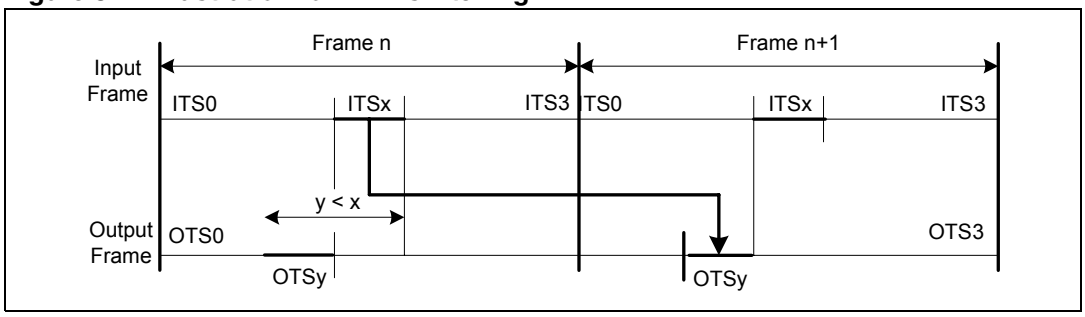


Figure 88. Storage in memory during an odd switched frame

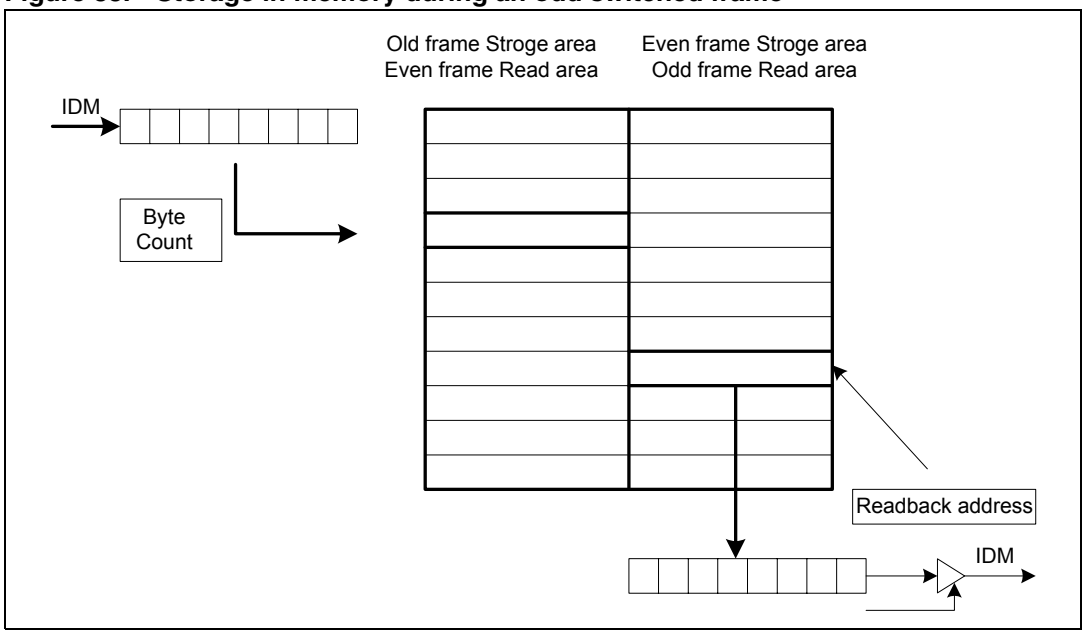
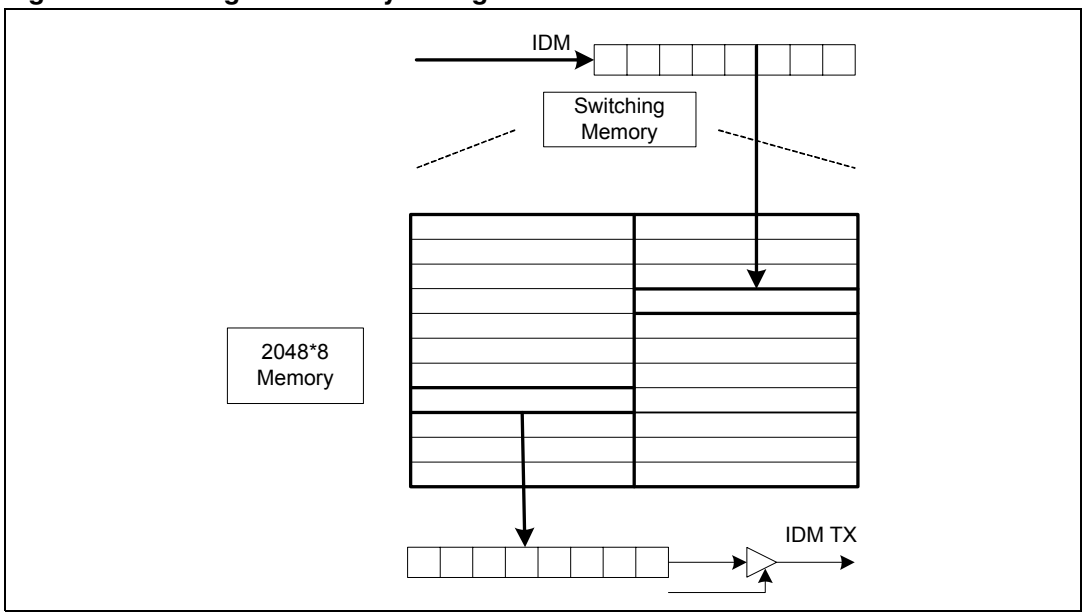


Figure 89. Storage in memory during an even switched frame



34.4.7 Buffer memory

Buffer memory allows bufferization of upto 16 channels. It contains two banks:

- one available for the TDM state machine,
- the other one for the AHB.

Each bank has a capacity of 16KB (total memory size is 32KB).

Timeslots that must be stored/played are declared in the action memory. This is done separately for the input and the output. For instance, during DTMF dialing, you don't need to play anything.

Buffers banks, unlike switching memory, do not switch at the frame rate but at the packet rate (usually 10ms, 20ms, or 30ms rate). A packet consists of a number of frames given by the *Table 717: TDM_Frame_NBR register (Offset 0x3C)*.

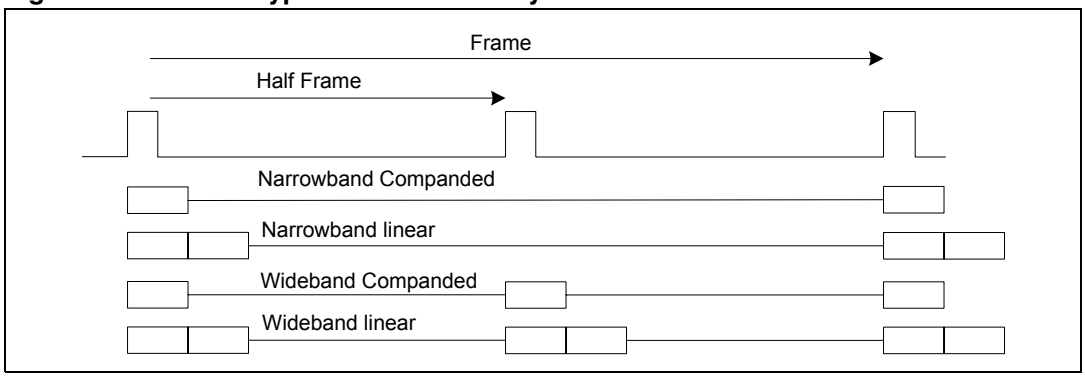
Number of frames per packet must be same for all channels. Hence, if some channels require 20ms packets size and others 30ms packet size, 10ms packet size will have to be selected, and the longest packet size requirement will be fulfilled with 3 transfers to the

external memory. However, it is possible to manage packets greater than 30 ms by reducing the number of channels. The buffer bank becomes full after the last timeslot of the last frame has been written in the buffer bank. Banks are switched and an interrupt is raised

Inside a buffer bank, memory pointer increases differently for each frame, according to the kind of channel:

1. *For Narrowband (8 kHz), Companded (8 bit) samples* - The pointer will progress by 1 byte at each frame.
2. *For Narrowband, Linear (16 bits) samples* - The pointer will progress by 2 bytes at each frame. The two bytes will be in consecutive time slots.
3. *For Wideband companded samples* - The pointer will also progress by 2 bytes at each frame. Anyway, the first sample will be in the first half of the frame and the second sample in the second half.
4. *For Wideband linear samples* - The pointer will progress by 4 bytes at each frame. Two bytes are located in the first half of the frame, two others in the second half.

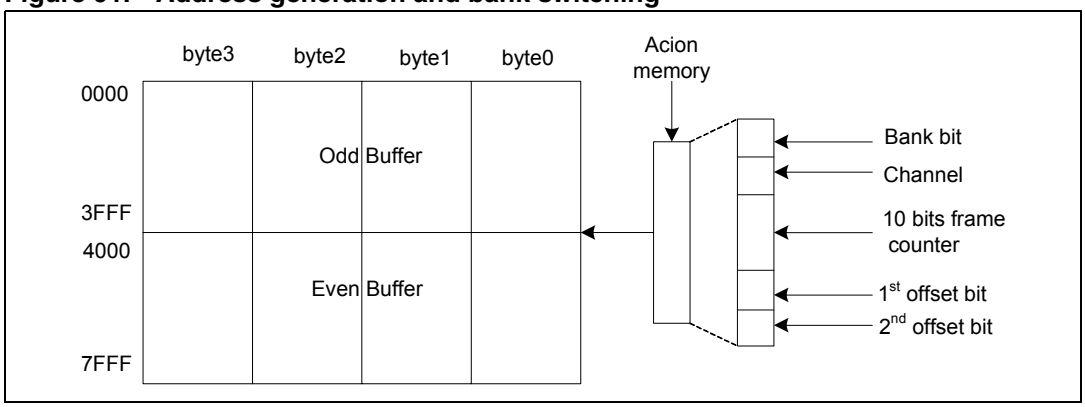
Figure 90. Various type of data carried by the TDM bus



To manage any data type using a single counter, a frame counter is implemented. From the above diagram, the location where the sample will has to be stored inside the current buffer area is given by:

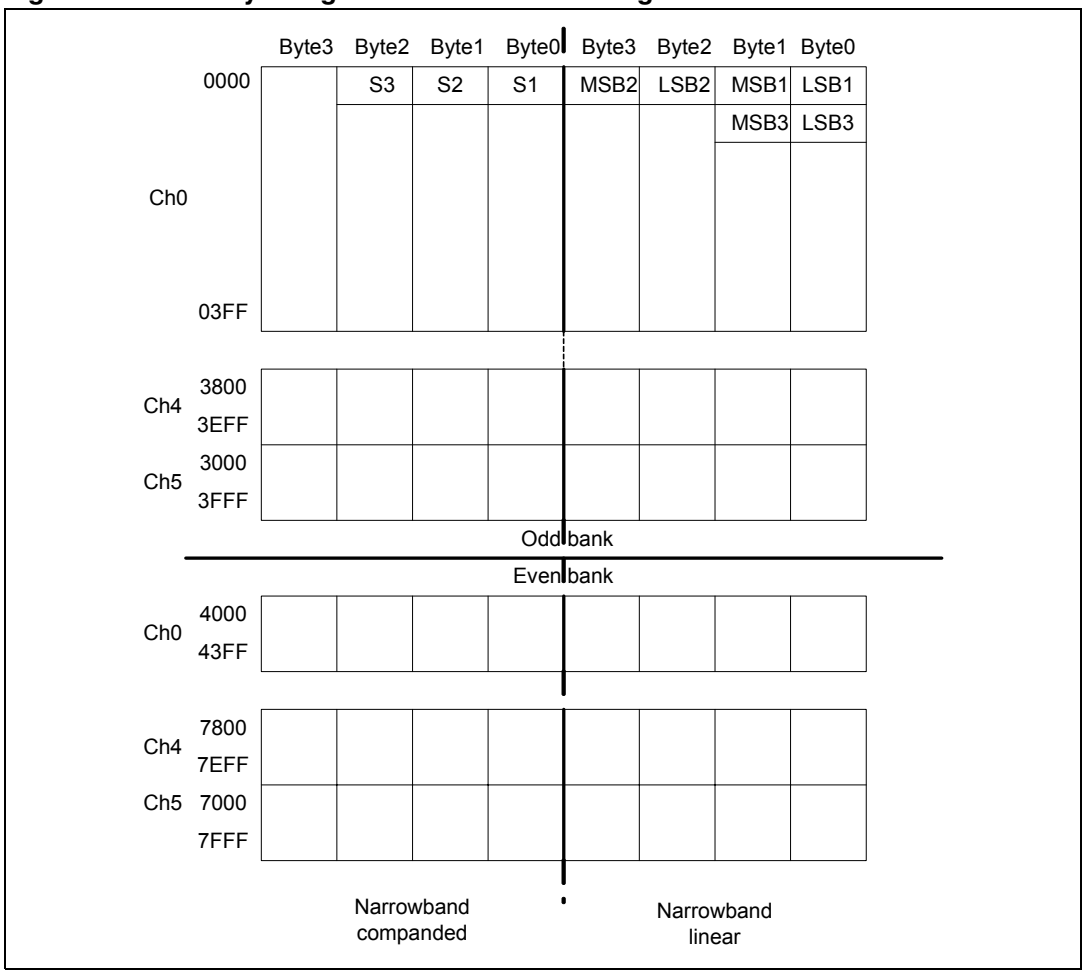
- Frame counter for the first configuration
- Frame counter and one LSB (offset bits inside the *action memory*) for the second and third configuration
- Frame counter and two LSB (offset bits the *action memory*) for the fourth configuration
- The bank information
- The channel number

Figure 91. Address generation and bank switching



Shown below is an example of storing 3 successive frames of the same channel (channel 0), considering the two narrowband cases with odd memory bank used for buffering samples.

Figure 92. Memory filling after 3 frames according narrowband cases



Shown below is an example of storing 3 successive frames of the same channel (channel 0), considering the two wideband cases with odd memory bank used for buffering samples.

Figure 93. Memory filling after 3 consecutive frames for two wideband cases

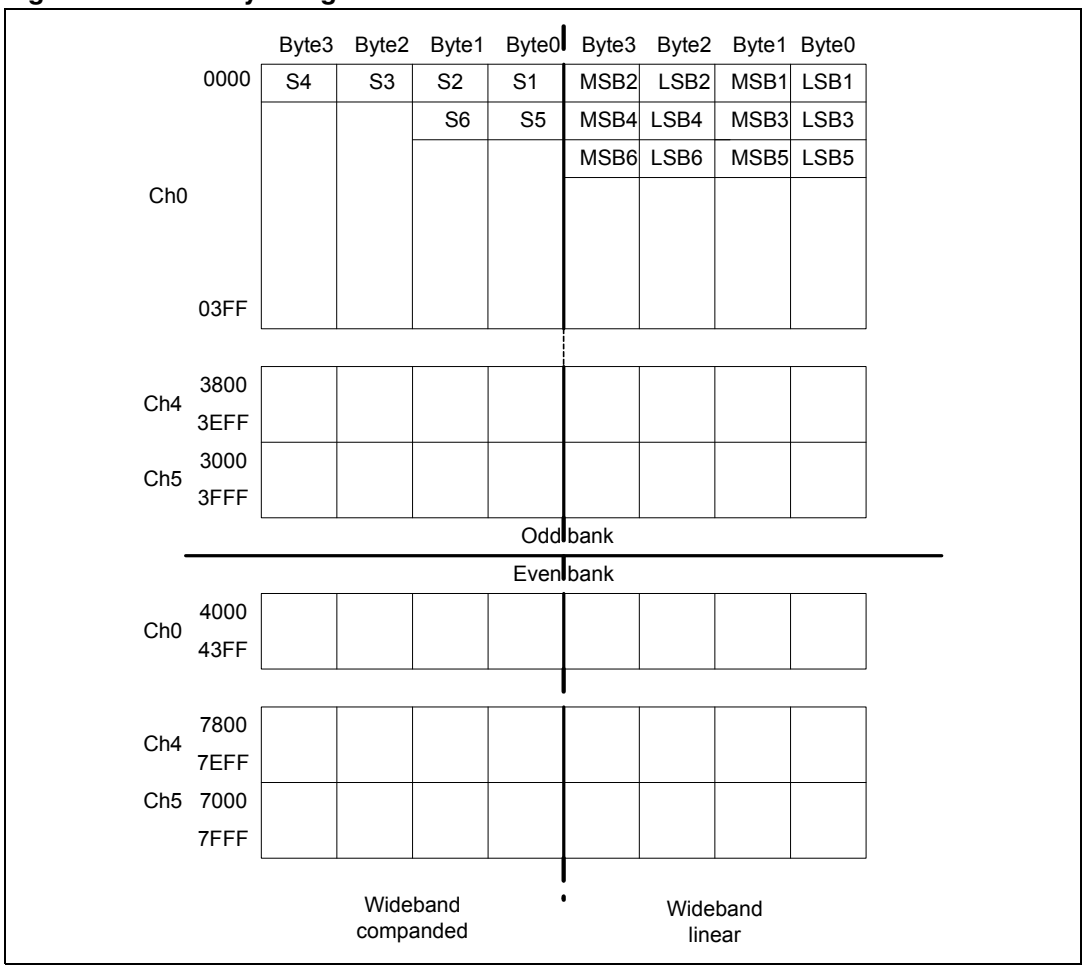


Figure 94. Sample management on odd frame

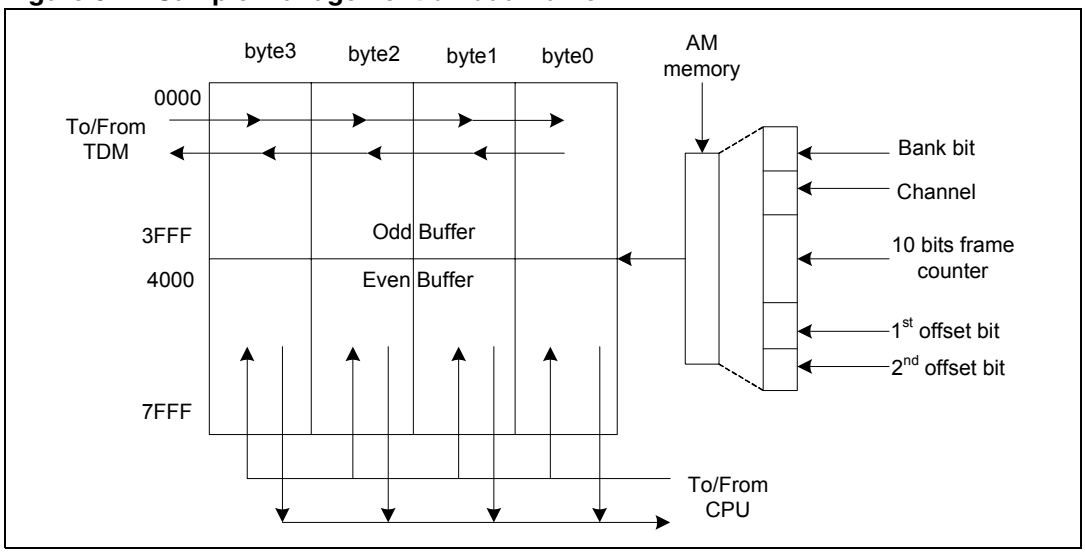
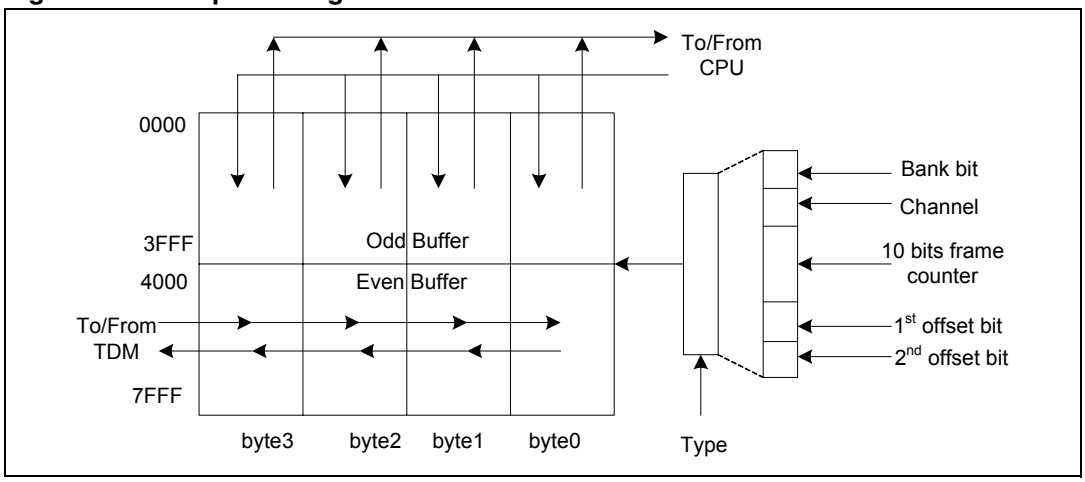
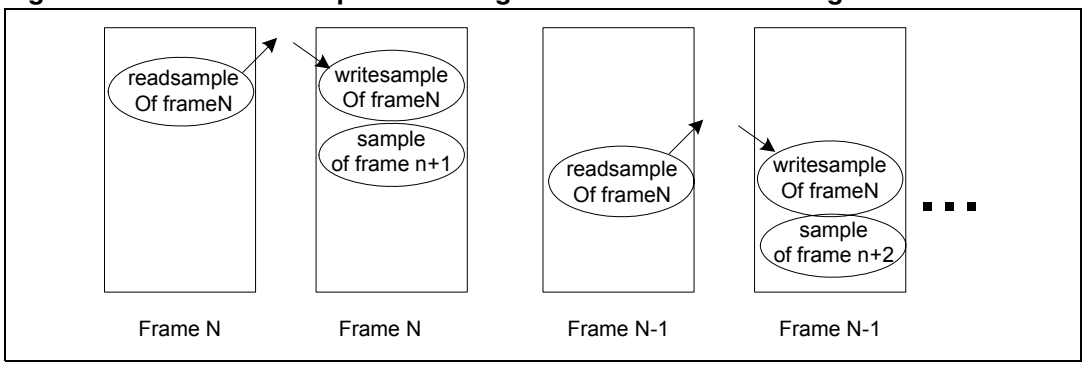


Figure 95. Sample management on even frame



In case of buffering mode, on the TDM side, for a given channel, during *frame n*, first the sample of *frame n* (provided by CPU) is read on the DOUT line and then the input sample of *frame n* from DIN line is stored at the same place.

Figure 96. Read/write sequence during frame N of a buffer for a given channel

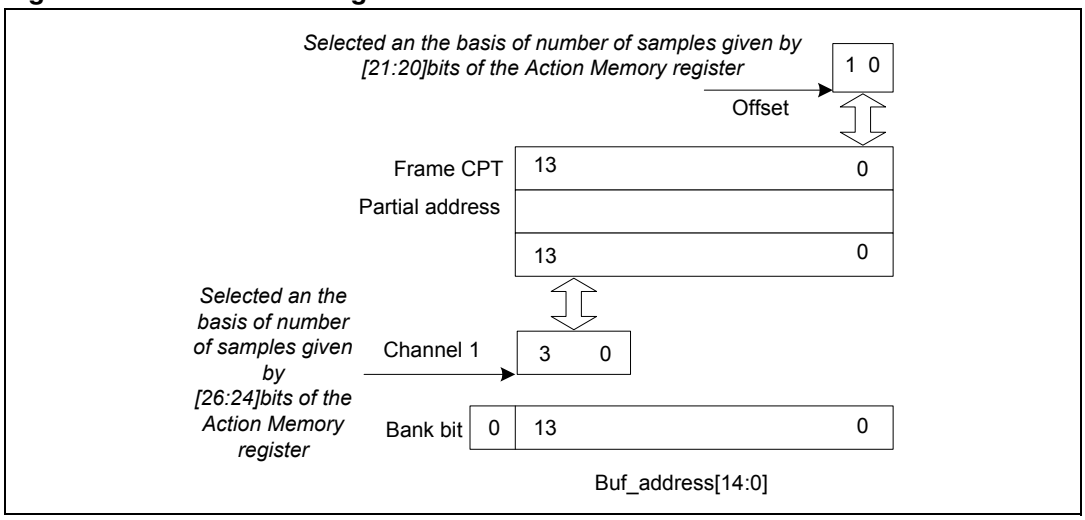


The address generation mechanism allows buffer extension when less than 16 channels are used.

The address is computed with four parameters

1. *The bank bit*
2. *The channel number* (programmed in [19:16] bits of *Table 727: Action memory* register and selected on the basis of number of channels given by [26:24] bits of the same *Table 727: Action memory*register)
3. *The frame counter*
4. *The offset bits* (programmed in [23:22] of *Table 727: Action memory* register and selected on the basis of number of samples given by [21:20] bits of the same *Table 727: Action memory* register)

Figure 97. Buffer address generation



34.4.8 I2S block

I2S interface is composed of 4 signals:

Table 699. I2S interface pins

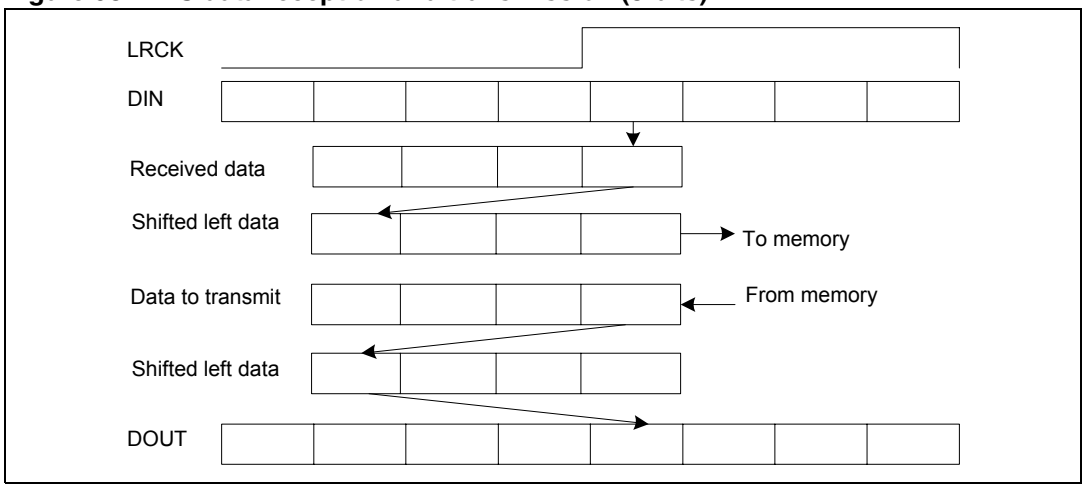
I2S_LRCK	Left and right channels synchronization (Master/slave)
I2S_CLK	I2S clock (Master/slave)
I2S_DIN	I2S input
I2S DOUT	I2S output (trim-state)

- I2S block is very similar to TDM block, but the frame sync can only be compliant with the Philips I2S definition.
- I2S block can be master or slave for the clock and sync signals (please refer to *Table 721: I2S_CONF register (Offset 0x4C)*
- Buffering is limited to 1024 samples (512 left and 512 right samples representing 64ms of voice). Data is stored always on 32 bits. Left and right channels are stored in two different buffers.
- In master mode, LRCK can be adjusted for 8, 16 or 32 bits width.
- Data width can be less than LRCK width. Input (received on I2S_DIN) and output (transmitted on DOUT) can be 8, 16 or 32 bits. The DOUT line can be high impedance when out of samples. Data is always stored in 32 bit format in the buffer. A shift left operation is possible to left align the data.
- In master mode, the CLK signal can be generated from different sources:
  - ClkR_Osci1: MCLK clock from external MCLK crystal
  - ClkR_Gpio4: external oscillator from PL_GPIO4 pin.
  - ClkR_Synt_2: From frequency synthesizer (source is AHB frequency),
  - TDM_CLK: I2S and TDM interfaces use the same clock.

The first three signals can be divided by the I2S_CLK block to adjust to the correct frequency. In slave mode, the clock will be received on I2S_CLK pin.

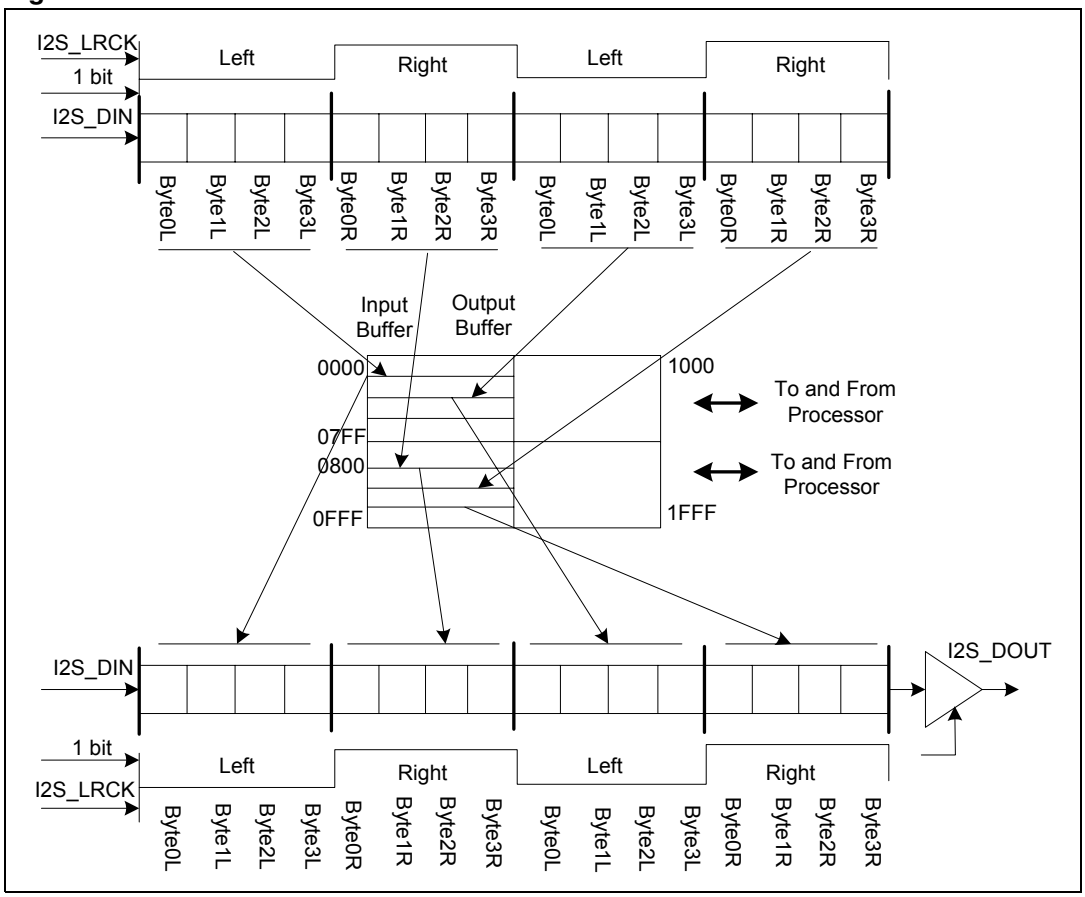
- Two banks are used to exchange the samples with the processor. The number of sample stored in a buffer is programmable (please refer to *Table 722: I2S_CONF2 register (Offset 0x6C)*). When the I2S interface reads and stores the data in one bank, the processor is owner of the other bank, allowing it to read the received data before writing a buffer to be played.
- When the two banks are switched, this can generate an interrupt or DMA transfer. When this event occurs, if the processer has not finished to process the previous input buffer and store the new output buffer, the processing is out of real time. It is the responsibility of the software to check that operations are done in real time.

Figure 98. I2S data reception and transmission (8 bits)



In order to avoid synchronization problems, the MSB of the buffer address is managed by the device itself to allow the processor to always access the correct buffer automatically. The processor always accesses the right bank seen between 0000 and 0FFF.

Figure 99. I2S data flow on 2*32 bit data



34.4.9 DAC block

DAC block implements a noise shapter of order 2 based on the TDM hardware. DAC cell is designed to work when not using the TDM. Its over sampling can be set between 32 and 256. DAC block uses samples placed in the buffer memory. Action memory informs if a new sample need to be sent to the DAC during the next byte. Input data must be 32 bit wide, either in 2's complement or binary representation.

When used in conjunction with the TDM, a bufferization channel has to be reserved for the DAC. In this case, the input sampling frequency must be either 8 kHz (standard TDM) or 16 kHz (when connecting wideband CODECs for instance). The number of bits in a frame must be fixed between 32 and 256, leading to an over sampling factor of 32 to 256. For example, an 8 kHz input and a 256 bit frame will generate a 204 kHz output.

Figure 100. DAC application

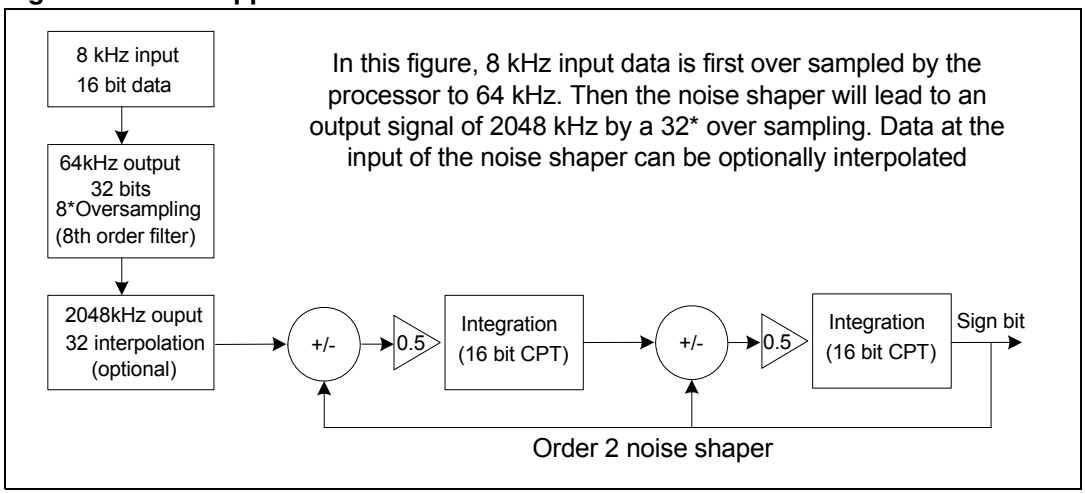
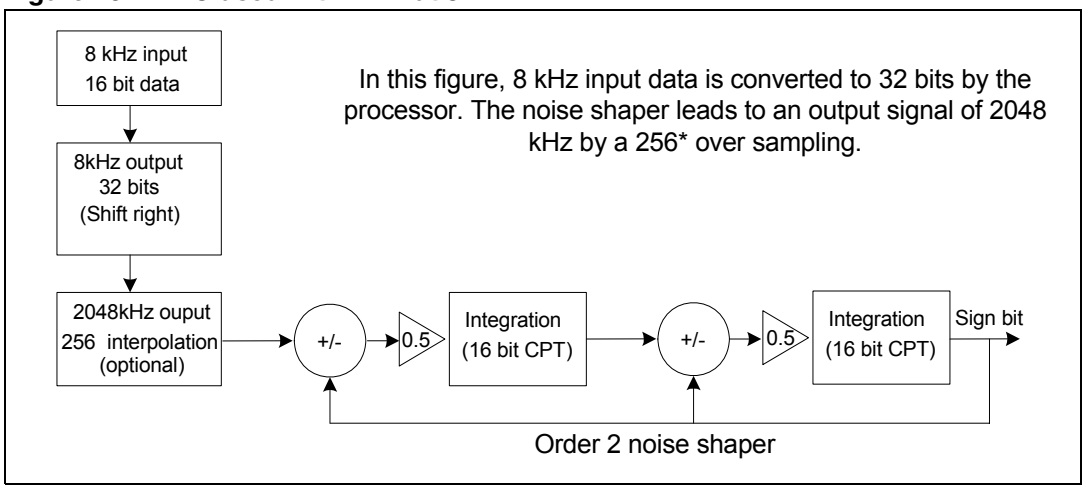


Figure 101. DAC used with TDM at 8 kHz



34.4.10 Camera interface

Camera Interface which is intended to get the data from a sensor in parallel mode (8 to 14 bits) and then store a full line in memory. Main features of the Camera Interface are:-

- It is an AHB slave port.
- 8 bit/10/12/14 bit parallel interface
- Embedded/External Synchronization for Line and Frame
- Crop Feature

Figure 102. Camera interface block diagram

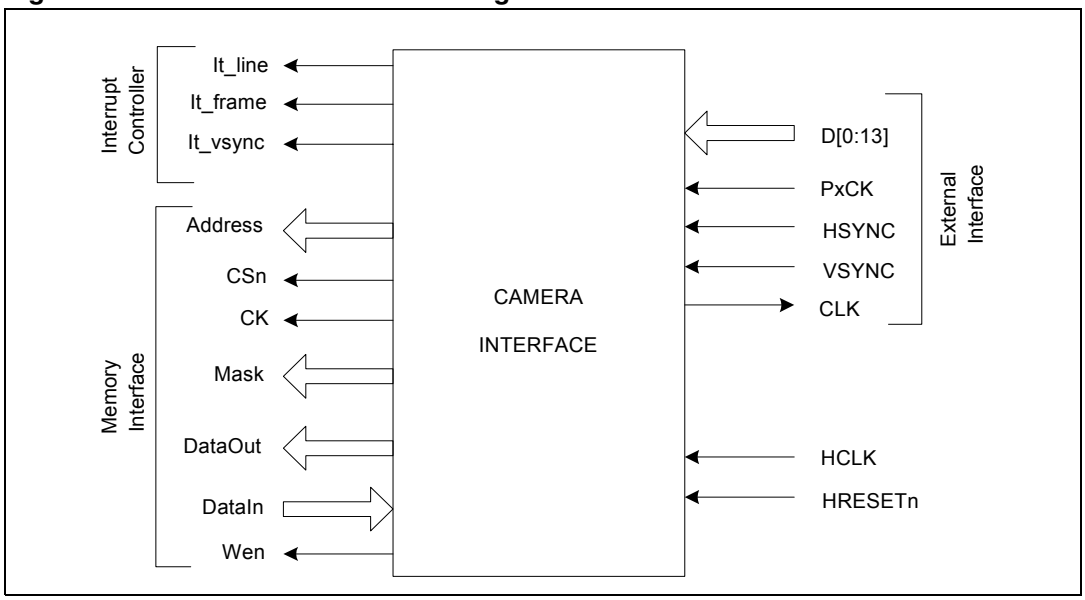


Table 700. Camera interface signal

Signal	Direction	Description
D[13:0]	In	Data Input
PxCK	In	Clock Input
HSYNC	In	Line Synchronization signal
VSYNC	In	Frame Synchronization signal
CLK	Out	Camera clock output (Refer to CAM_Control Register bits (15:13)
it line	Out	Line end interrupt
it frame	Out	Frame end interrupt
it vsync	Out	New frame interrupt
[Memory Interface]	-	Stores line in memory

34.4.11 SPI-I2C block

- SPI-I2C block allows upto 8 SPI and I2C devices or codecs to share one I2C or SPI interface in fixed part of SPEAr300.
- SPI is a chip select bus and only one chip select signal is available. This block allows switching this signal to multiple devices.
- Similarly, I2C being an addressed bus, it is not possible to control all devices at the same device address. The block sends I2C_SCL signal only to the desired device.
- Each of the 8 available outputs of this block can be programmed to function as SPI chip select, I2C clock or simple general purpose output pins (please refer to *Table 723: I2S_CLK_CONF register (Offset 0x50)* and *Table 721: I2S_CONF register (Offset 0x4C)* registers).

34.4.12 General purpose GPIOs G8 and G10

- G8(7:0) and G10(9:0) are a set of eighteen general purpose GPIO provided primarily for SLIC management.
- G10(7:0) have an added capability of generating an interrupt on change of value or persistence of change similar to GPIO_IT bus.

Selection of source of interrupt generation between G10(7:0) and IT(7:0) is explained in programmer's model (please refer to Table 711: IT_GEN register (Offset 0x24))

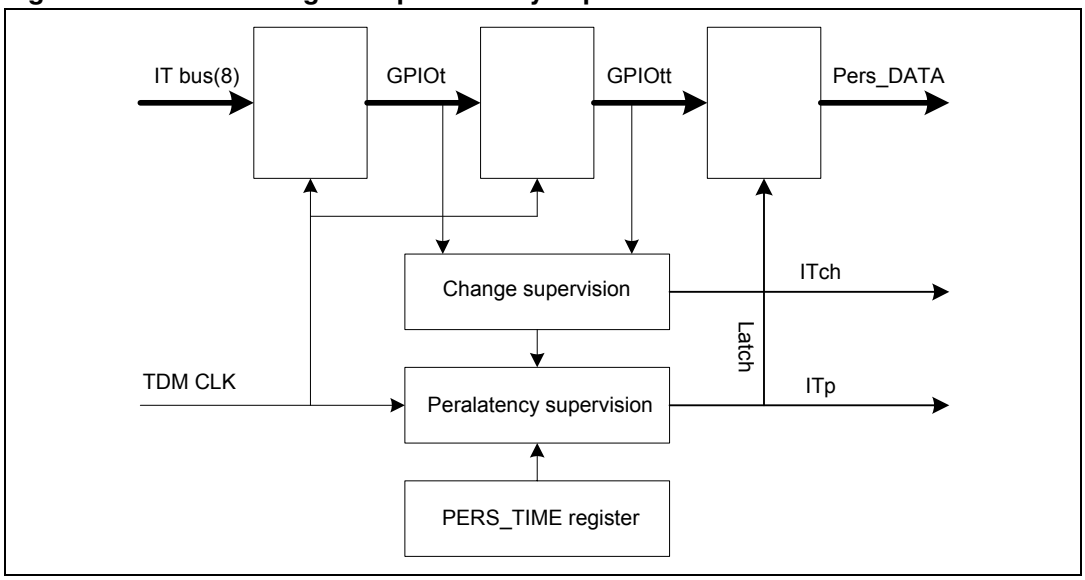
34.4.13 IT bus

- IT bus is an 8 bit input bus that is supervised.
- When a change is detected on this bus it can generate the ITch interrupt.
- It can be important to check that this change persists for more than a determined persistency time (as programmed in *Table 714: PERS_time register (Offset 0x30)*) before generating an interrupt. The IT interrupt is then generated.

The purpose of such an interface is to supervise the hook detection of up to 8 SLICs. Alternatively, it can be used to debounce a switch or simply generate an interrupt when a signal toggles.

Persistency time is based on TDM clock. The number of clocks to be used for persistency is programmable. The signal is latched two times and the two latched signals compared. If they are different the ITch interrupt can be generated if programmed so. It is also possible to read both the latch register. If a line is programmed to generate an interrupt on a persistent change, the data is latched after the persistency time and interrupt ITp is generated. This latched data can also be read.

Figure 103. IT bus change and persistency supervision



34.5 Programmer's model

This section describes the programmer's model.

34.5.1 Address map

Table 701. Telecom address map

Start Address	End Address	Name
0x5000_0000	0x5000_FFFF	Register Set
0x5001_0000	0x5001_0FFF	Action Memory
0x5003_0000	0x5003_7FFF	Buffer Memory
0x5004_0000	0x5004_0FFF	Sync Memory
0x5005_0000	0x5005_0FFF	I2S Bank 1
0x5005_1000	0x5005_1FFF	I2S Bank 2

34.5.2 Register set

The base address of telecom registers is **0x5000_0000.**

Summary of registers

Table 702. Telecom registers

Offset	Type	Name
0x00	RO	BOOT
0x04	RW	TDM_CONF
0x08	RW	GPIO8_DIR
0x0C	RW	GPIO10_DIR
0x10	RW	GPIO8_OUT
0x14	RW	GPIO10_OUT
0x18	RO	GPIO8_IN
0x1C	RO	GPIO10_IN
0x20	Reserved	
0x24	RW	IT_GEN
0x28	RO	GPIOt
0x2C	RO	GPIOtt
0x30	RW	PERS_TIME
0x34	RW	PERS_DATA
0x38	RW	TDM_TSLOT_NBR
0x3C	RW	TDM_FRAME_NBR
0x40	RW	TDM_SYNC_GEN
0x44	RW	SPI_I2C_usage
0x48	RW	SPI_I2C_active
0x4C	RW	I2S_CONF

Table 702. Telecom registers (continued)

Offset	Type	Name
0x50	RW	I2S_CLK_CONF
0x54	RW	Interrupt mask
0x58	RO	Interrupt status
0x6C	RW	I2S_CONF2

34.6 Description of registers

This section describes the registers of telecom block.

34.6.1 Boot register

Boot register is dedicated to the boot ROM software. The bits [15:00] are a Magic number 0x1C8C". Bits 27 to 20 (H7 to H0) informs the boot software about the hardware. Specially, BOOTP and TFTP can be run using a dedicated MAC address and a dedicated file name. Bits 19 to 16 (B3 to B0) informs the hardware about the peripheral used for booting: USB, Ethernet or memories.

Table 703. Boot register (Offset 0x00)

Bits	Name	Comments
[31:28]	Reserved	
[27:20]	H	Height bits from setting pins H7 to H0. This represents a number between 0 and 255. Boot ROM will use this information running BOOTP by taking 00:80:E1:00:xx:01 as MAC address and requesting xloader.bin file in TFTP process. "xx" is the hexadecimal value of the binary number H[7:0].
[19:16]	B	Informs the hardware about the peripheral used for booting: USB, Ethernet or memories.
[15:00]	Magic Number	0x1C8C

34.6.2 TDM_conf register

This register is used by the CLK_GEN module. It allows configuration of the internal clock that will be used as reference for the TDM.

RESET: all ‘0’

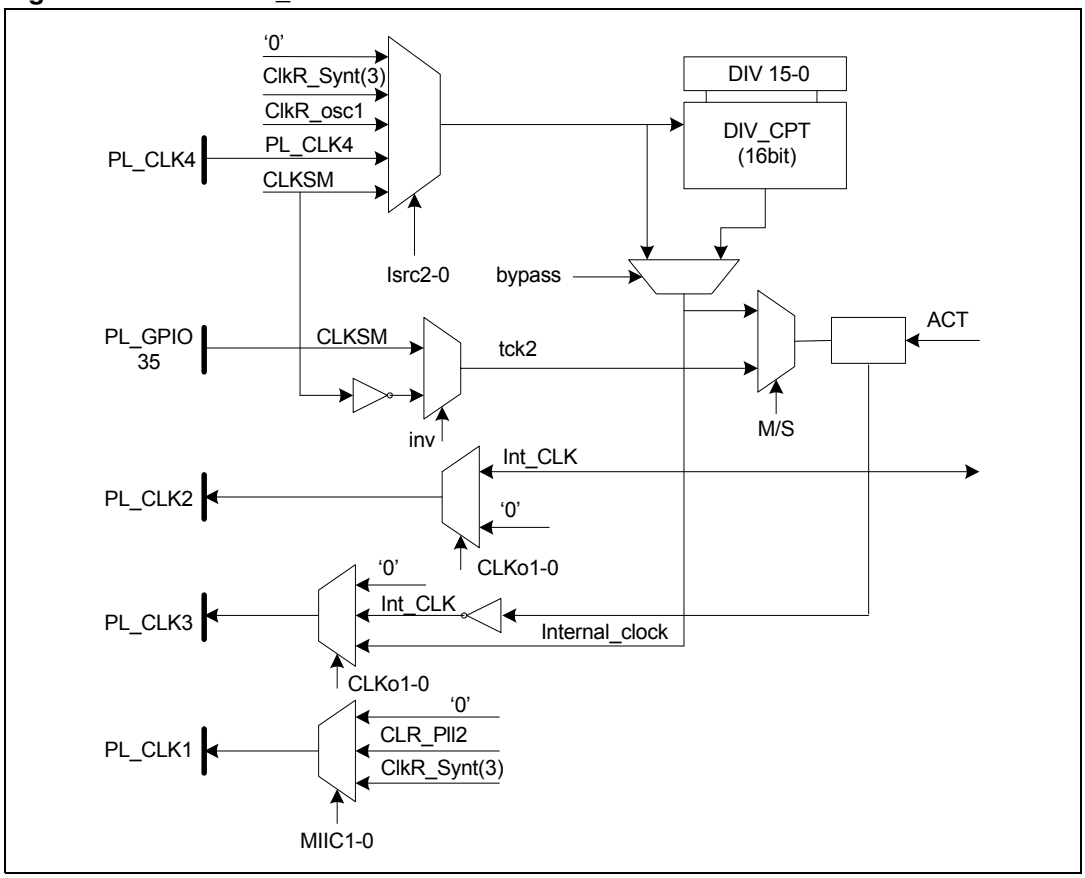
Table 704. TDM_conf register (Offset 0x04)

Bits	Name	Comments		
[31]	LBio	loopback from DIN to DOUT (external) *when LBio = ‘0’* no loopback is implemented *when LBio = ‘1’* DOUT plays back the DIN value		
[30]	LBoi	loopback from DOUT to DIN *when LBio = ‘0’* no loopback is implemented *when LBio = ‘1’* DIN receives the DOUT value		
[29]	LBshio	loopback from shift_in register to shift_out register (external) *when LBshio =’0’* no loopback is implemented *when LBShio = ‘1’* DOUT plays back the DIN value delayed by one byte.		
[28]	LBSD	loopback from selected data (sw_data_out or buf_data_out to store_in register *when LBSD = ‘0’* no loopback is implemented *when LBSD = ‘1’* the selected data is sent to the memories (sw or buf) through store_in.		
[27]	LBPD	loopback from parallel data (sw_data_out to sw_data_in and buf_data_out to buf_data_in) *When LBPD = ‘0’* no loopback is implemented *When LBPD = ‘1’* the loopback is set from incoming data from memory to outgoing data to memory, both for sw and buf channels. sw_data_out is sent to sw_data_in and buf_data_out is sent to buf_data_in.		
[26]	ACT	activates the int_CLK clock that is sent to all other blocks. This initialization is mandatory when using any telecom function *ACT = ‘0’:* int_CLK is always 0. *ACT = ‘1’:* int_CLK is either the generated internal clock or the slave external clock.		
[25:24]	CLK	these bits select what is output on the pins according to the following table:		
		CLKo1-0	PL_Clk2	PL_Clk3
		00	0	0
		01	Int_CLK	0
		10	Int_CLK	/int_CLK
		11	Int_CLK	Internal_clock
[23:22]	MIIC	selects the clock to be output on PL_CLK1 according to the following table:		
		MIIC	PL_Clk1	
		10	ClkR_Pll2	
		11	ClkR_Synt(3)	

Table 704. TDM_conf register (Offset 0x04) (continued)

Bits	Name	Comments	
[21:19]	Isrc	Others	reserved
		selection of the input source for the 7 bits divider	
		3'b000	0
		3'b001	CLKSM (pin TDM_CLK)
		3'b010	ClkR_oscl
		3'b011	ClkR_Synt(3)
		3'b100	PL_Clk4
		others	reserved
[18:03]	Div	this value will be compared to the divider counter value. When if it matches, the generated signal will toggle. The output of the divider stage is then the input frequency divided by {2 * (D[15-0] + 1)}	
[02]	Bypass	*when bypass = '1',* the osrc output is used as internal clock for the TDM logic (Int_CLK). *when bypass = '0',* the divider output is used for the TDM logic.	
[01]	Inv	*when inv = '0',* the CLKSM pin signal is not internally inverted (normally used in slave mode) *when inv= '1',* the CLKSM signal is inverted to generate the internal clock Int_CLK in slave mode.	
[00]	M/S	when M/S = 0, the device is in slave mode. The CLK pin is an input.	

Figure 104. TDM CLK_GEN bits



34.6.3 GPIO8_DIR register

GPIO8_DIR informs about the direction of the GPIO8 register pins.

RESET: all '1'

Table 705. GPIO8_DIR register (Offset 0x08)

Bits	Name	Comments
[31:08]	Reserved	When Dirx=0, the relevant GPIOx pin is set as output When Dirx=1, the relevant GPIOx pin is set as input
[07]	Dir7	
[06]	Dir6	
[05]	Dir5	
[04]	Dir4	
[03]	Dir3	
[02]	Dir2	
[01]	Dir1	
[00]	Dir0	

34.6.4 GPIO10_DIR register

GPIO10_DIR informs about the direction of the GPIO10 register pins.

RESET: all '1'

Table 706. GPIO10_DIR register (Offset 0x0C)

Bits	Name	Comments
[31:10]	Reserved	When Dirx=0, the relevant GPIOx pin is set as output When Dirx=1, the relevant GPIOx pin is set as input
[09]	Dir9	
[08]	Dir8	
[07]	Dir7	
[06]	Dir6	
[05]	Dir5	
[04]	Dir4	
[03]	Dir3	
[02]	Dir2	
[01]	Dir1	
[00]	Dir0	

34.6.5 GPIO8_out register

This register contains the value that will be out on the pins for GPIO8 block when they are in output mode

RESET: all '0'

Table 707. GPIO8_out register (Offset 0x10)

Bits	Name	Comments
[31:08]	Reserved	The bits to be out on the respective pins if they are in output mode
[07]	Val7	
[06]	Val6	
[05]	Val5	
[04]	Val4	
[03]	Val3	
[02]	Val2	
[01]	Val1	
[00]	Val0	

34.6.6 GPIO10_out register

This register contains the value that will be out on the pins for GPIO10 block when they are in output mode.

RESET: all ‘0’

Table 708. GPIO10_out register (Offset 0x14)

Bits	Name	Comments
[31:10]	Reserved	The bits to be out on the respective pins if they are in output mode.
[09]	Val9	
[08]	Val8	
[07]	Val7	
[06]	Val6	
[05]	Val5	
[04]	Val4	
[03]	Val3	
[02]	Val2	
[01]	Val1	
[00]	Val0	

34.6.7 GPIO8_in

This register contains the value from the pins connected to GPIO8 block.

RESET: all ‘0’

Table 709. GPIO8_in (Offset 0x18)

Bits	Name	Comments
[31:08]	Reserved	Latched value from the respective pins.
[07]	In7	
[06]	In6	
[05]	In5	
[04]	In4	
[03]	In3	
[02]	In2	
[01]	In1	
[00]	In0	

34.6.8 GPIO10_in register

This register contains the value from the pins connected to GPIO10 block.

RESET: all '0'

Table 710. GPIO10_in register (Offset 0x1C)

Bits	Name	Comments
[31:10]	Reserved	Latched value from the respective pins.
[09]	In9	
[08]	In8	
[07]	In7	
[06]	In6	
[05]	In5	
[04]	In4	
[03]	In3	
[02]	In2	
[01]	In1	
[00]	In0	

34.6.9 IT-GEN register

This register manages pin IT[7:0] or GPIO10_in [7-0]. It allows generating an interrupt when either a change appears on one pin, or when a change is stable for a persistency time given by PERS_time register. The supervision is done on the global Byte and not bit per bit (stability must be encountered on the 8 pins).

RESET: all '0'

Table 711. IT_GEN register (Offset 0x24)

Bits	Name	Comments
[31:24]	Reserved	
[23]	GPIO7	GPIOx=1: GPIO10_in[x] pin is taken into account to generate the interrupts GPIOx=0: IT[x] pin is tkane into account to generate the interrupts
[22]	GPIO6	
[21]	GPIO5	
[20]	GPIO4	
[19]	GPIO3	
[18]	GPIO2	
[17]	GPIO1	
[16]	GPIO0	
[15]	Ch7	1 - Interrupt on change on pin 7 0 - No interrupt on change on pin 7
[14]	P7	1 - Interrupt when 8 bits are stable for pers_time after last change on P7 0 - No interrupt on stability of change
[13]	Ch6	1 - Interrupt on change on pin 6 0 - No interrupt on change on pin 6

Table 711. IT_GEN register (Offset 0x24) (continued)

Bits	Name	Comments
[12]	P6	1 - Interrupt when 8 bits are stable for pers_time after last change on P6 0 - No interrupt on stability of change
[11]	Ch5	1 - Interrupt on change on pin 5 0 - No interrupt on change on pin 5
[10]	P5	1 - Interrupt when 8 bits are stable for pers_time after last change on P5 0 - No interrupt on stability of change
[09]	Ch4	1 - Interrupt on change on pin 4 0 - No interrupt on change on pin 4
[08]	P4	1 - Interrupt when 8 bits are stable for pers_time after last change on P4 0 - No interrupt on stability of change
[07]	Ch3	1 - Interrupt on change on pin 3 0 - No interrupt on change on pin 3
[06]	P3	1 - Interrupt when 8 bits are stable for pers_time after last change on P3 0 - No interrupt on stability of change
[05]	Ch2	1 - Interrupt on change on pin 2 0 - No interrupt on change on pin 2
[04]	P2	1 - Interrupt when 8 bits are stable for pers_time after last change on P2 0 - No interrupt on stability of change
[03]	Ch1	1 - Interrupt on change on pin 1 0 - No interrupt on change on pin 1
[02]	P1	1 - Interrupt when 8 bits are stable for pers_time after last change on P1 0 - No interrupt on stability of change
[01]	Ch0	1 - Interrupt on change on pin 0 0 - No interrupt on change on pin 0
[00]	P0	1 - Interrupt when 8 bits are stable for pers_time after last change on P0 0 - No interrupt on stability of change

34.6.10 GPIOt register

In this register the processor can read the value of the IT pins latched by the int_CLK clock.

RESET: all '0'

Table 712. GPIOt register (Offset 0x28)

Bits	Name	Comments
[31:08]	Reserved	
[07:00]	IT	Value of the IT pins latched by int_CLK

34.6.11 GPIOtt register

In this register the processor can read the value of the GPIOt pins latched by the int_CLK clock (IT pins latched two times).

RESET: all '0'

Table 713. GPIOtt register (Offset 0x2C)

Bits	Name	Comments
[31:08]	Reserved	
[07:00]	IT	Value of the **GPIOt Register (7:0)** latched by int_CLK

34.6.12 PERS_time register

This register will set the time for which the stability is observed before generating an Interrupt on pins change. The time set is the number of clocks minus one for which the signal must be stable

PERS_time value must not be 1. Zero does not generate any interrupt

RESET: all '0'

Table 714. PERS_time register (Offset 0x30)

Bits	Name	Comments
[31:08]	Reserved	
[07:00]	PT	Persistency time counter value - persistency time will be **(PT[7-0] + 1) * T(int_CLK)**

34.6.13 PERS_data register

When persistency time is met on a new 8 bit input, GPIOtt is latched on this register. This is the new value that was stable for more than pers_time.

RESET: all '0'

Table 715. PERS_data register (Offset 0x34)

Bits	Name	Comments
[31:08]	Reserved	
[07:00]	PV	The latched value of GPIOtt registers.

34.6.14 TDM_timeslot_NBR register

This register informs about the number of timeslots contained in the frame. It must be informed also in slave mode if other sync generation is required.

RESET: all '0'

Table 716. TDM_timeselot_NBR register (Offset 0x38)

Bits	Name	Comments
[31:11]	Reserved	
[10:00]	TSN	The number of timeslots in a frame. 0 x 200 = 512 timeslots.

34.6.15 TDM_frame_NBR register

This register informs about the number of samples that must be compiled in buffering mode before switching the banks and interrupting the processor.

RESET: all '0'

Table 717. TDM_Frame_NBR register (Offset 0x3C)

Bits	Name	Comments
[31:15]	Reserved	
[14:00]	FRN	Number of frames in buffer for active channels in buffering mode. Channels are opened in accordance with the contents of Action Memory.

34.6.16 TDM_SYNC_GEN register

SYNC_GEN register defines the parameters for the generation of first four (SYNC0 to SYNC3) synchronization signals.

Table 718. TDM_SYNC_GEN register (Offset 0x40)

Bits	Name	Comments
[31]	Nsh	When Nsh = 1'b1, the BUF memory is entirely kept by the RAS and can not be accessed by the AHB.
[30]	BB	Informs about the buffer bank in order to synchronize the state machine and the processor. *When BB = 0*,the processor must read and write the second bank (0x50034000-0x50037FFF); *When BB = 1*, the processor must read and write the first bank (0x50030000-0x50033FFF).
[29]	SB	Informs about the switching bank in order to synchronize the state machine and the processor. *When SB = 0*, the processor must read in the second bank (0x50020400-0x500207FF) and write in the first bank (0x50020000-0x500203FF); *When SB=1*, the processor must read in the first bank (0x50020000-0x500203FF) and write in the second bank (0x50020400-0x500207FF).

Table 718. TDM_SYNC_GEN register (Offset 0x40) (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Comments</th></tr>
<tr><td>[28]</td><td>ABBM</td><td>Automatic buffer bank management:
When ABBM = 0, the AHB delivers the full address for the buffer memory access. The processor then need to know which bank is available for it. When it accesses the wrong bank a bad value is returned.
When ABBM = 1, the MSB of the address will be overwritten by the RAS. The AHB will then only access to the part that is available for it.
Anyway, care must be taken during a short time when the memory is reserved for the RAS. In this case the AHB access if any will be corrupted. To avoid this, there should be no AHB access while the Nsh (not_shared_memory) bit is "1". NSh will generate an interrupt to inform that the memory can be accessed.</td></tr>
<tr><td>[27]</td><td>BBVal</td><td>This bit denotes the buffer bank in case the two buffer banks cannot toggle (BBfrz = 1).</td></tr>
<tr><td>[26]</td><td>BBfrz</td><td>buffer banks frozen
When BBfrz =0, the two buffer banks can toggle.
When BBfrz= 1, the two banks are frozen. The used bank is given by BBVal.</td></tr>
<tr><td>[25]</td><td>nAHB_L</td><td>The value to force for the lower buffer memory for the AHB
nAHB_L = 0: the AHB is able to read/write the BUF memory in the lower bank.
nAHB_L = 1: RAS logic can access the lower BUF memory if BB = 0, AHB can access the lower BUF memory if BB = 1.</td></tr>
<tr><td>[24]</td><td>nAHB_H</td><td>The value to force for the upper buffer memory for the AHB
nAHB_H = 0: the AHB is able to read/write the BUF memory in the upper bank.
nAHB_H = 1: RAS logic can access the upper BUF memory if BB = 1, AHB can access the upper memory if BB = 0.</td></tr>
<tr><td>[23]</td><td>S47</td><td>Informs if one of the SYNC4 to SYNC7 signals are used externally.
S47 = 0: no SYNC7-4 signal is used. It is not useful to read the SYNC memory.
S47 = 1: at least one of the SYNC7-4 signals is used.</td></tr>
<tr><td>[22:21]</td><td>Bdel3</td><td>Delay in byte between the SYNC3 and one of the previous channels.
<table>
<tr><th>Bdel3[1]</th><th>Bdel3[0]</th><th>Delay</th></tr>
<tr><td>0</td><td>0</td><td>No delay</td></tr>
<tr><td>0</td><td>1</td><td>1 byte</td></tr>
<tr><td>1</td><td>0</td><td>2 bytes</td></tr>
<tr><td>1</td><td>1</td><td>4 bytes</td></tr>
</table></td></tr>
</table>

Table 718. TDM_SYNC_GEN register (Offset 0x40) (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Comments</th></tr>
<tr><td>[20]</td><td>Bdel3</td><td>Informs if the frame synch (SYNC3) is for a delayed frame of non delayed frame
When bdel3 = 0, the frame sync signal is aligned with the first bit of the timeslot
When bdel3 =1, the frame sync is present one bit before the first bit of the first timeslot.</td></tr>
<tr><td>[19]</td><td>Wb3</td><td>This bit informs if the PCM synchro (SYNC3) must be in wideband type (one additional pulse at the middle of the frame)
When Wb3 = 0, the PCM carries narrowband data. Only one pulse is required for the framesync (generally every 125µS).
When Wb3 = 1, the PCM carries wideband data. Two pulses are required for the framesync (generally every 62.5µs)</td></tr>
<tr><td>[18:17]</td><td>Use3</td><td>These bits are used with Wb3 according to the following table:
<table>
<tr><td>Wb3</td><td>Use3[1]</td><td>Use3[0]</td><td>Synchro type</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>I2S</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCM/DSP-SFS</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>LFS</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCM wideband (SFS)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>LFS</td></tr>
</table></td></tr>
<tr><td>[16:15]</td><td>Bdel2</td><td>Delay in byte between the SYNC2 and one of the previous channel
<table>
<tr><td></td><td>Bdel2[1]</td><td>Bdel2[0]</td><td>Delay</td></tr>
<tr><td></td><td>0</td><td>0</td><td>No delay</td></tr>
<tr><td></td><td>0</td><td>1</td><td>1 byte</td></tr>
<tr><td></td><td>1</td><td>0</td><td>2 bytes</td></tr>
<tr><td></td><td>1</td><td>1</td><td>4 bytes</td></tr>
</table></td></tr>
<tr><td>[14]</td><td>bdel2</td><td>Informs if the frame synch (SYNC2) is for a delayed frame of non delayed frame
When bdel2 = 0, the frame sync signal is aligned with the first bit of the timeslot
When bdal2 = 1, the frame sync is present one bit before the first bit of the first timeslot.</td></tr>
</table>

Table 718. TDM_SYNC_GEN register (Offset 0x40) (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Comments</th></tr>
<tr><td>[13]</td><td>Wb2</td><td>This bit informs if the PCM synchro (SYNC2) must be in wideband type (one additional pulse at the middle of the frame)
When Wb2 = 0, the PCM carries narrowband data. Only one pulse is required for the framesync (generally every 125µS).
When Wb2 = 1, the PCM carries wideband data. Two pulses are required for the framesync (generally every 62.5µs)</td></tr>
<tr><td>[12:11]</td><td>Use2</td><td>These bits are used with Wb2 according to the following table
<table>
<tr><td>WB2</td><td>Use2[1]</td><td>Use2[0]</td><td>**Synchro type**</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>I2S</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCM/DSP-SFS</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>LFS</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCM wideband (SFS)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>LFS</td></tr>
</table></td></tr>
<tr><td>[10:09]</td><td>Bdel1</td><td>Delay in byte between the SYNC1 and one of the previous channel.
<table>
<tr><td></td><td>Bdel1[1]</td><td>Bdel1[0]</td><td>**Delay**</td></tr>
<tr><td></td><td>0</td><td>0</td><td>No delay</td></tr>
<tr><td></td><td>0</td><td>1</td><td>1 byte</td></tr>
<tr><td></td><td>1</td><td>0</td><td>2 bytes</td></tr>
<tr><td></td><td>1</td><td>1</td><td>4 bytes</td></tr>
</table></td></tr>
<tr><td>[08]</td><td>bdel1</td><td>Informs if the frame synch *(SYNC1)* is for a delayed frame of non delayed frame
When bdel1 = 0, the frame sync signal is aligned with the first bit of the timeslot
When bdel1 = 1, the frame sync is present one bit before the first bit of the first timeslot.</td></tr>
<tr><td>[07]</td><td>Wb1</td><td>This bit informs if the PCM synchro *(SYNC1)* must be in wideband type (one additional pulse at the middle of the frame)
When Wb1 = 0, the PCM carries narrowband data. Only one pulse is required for the framesync (generally every 125µS).
When Wb1 = 1, the PCM carries wideband data. Two pulses are required for the framesync (generally every 62.5µs)</td></tr>
</table>

Table 718. TDM_SYNC_GEN register (Offset 0x40) (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Comments</th></tr>
<tr><td>[06:05]</td><td>Use1</td><td>These bits are used with Wb1 according to the following table
<table>
<tr><td>Wb1</td><td>Use1[1]</td><td>Use1[0]</td><td>Synchro type</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>I2S</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCM/DSP-SFS</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>LFS</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCM wideband (SFS)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>LFS</td></tr>
</table></td></tr>
<tr><td>[04]</td><td>bdel0</td><td>Informs if the frame synch (SYNC0) is for a delayed frame of non delayed frame
When bdel0 = 0, the frame sync signal is aligned with the first bit of the timeslot
When bdal0 = 1, the frame sync is present one bit before the first bit of the first timeslot.</td></tr>
<tr><td>[03]</td><td>Wb0</td><td>This bit informs if the PCM synchro (SYNC0) must be in wideband type (one additional pulse at the middle of the frame)
When Wb0 = 0, the PCM carries narrowband data. Only one pulse is required for the framesync (generally every 125µS).
When Wb0 = 1, the PCM carries wideband data. Two pulses are required for the framesync (generally every 62.5µs)</td></tr>
<tr><td>[02:01]</td><td>Use0</td><td>These bits are used with Wb0 according to the following table:
<table>
<tr><td>Wb0</td><td>Use0[1]</td><td>Use1[0]</td><td>Synchro type</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>I2S</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCM/DSP-SFS</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>LFS</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCM wideband (SFS)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>LFS</td></tr>
</table></td></tr>
<tr><td>[00]</td><td>M/S</td><td>When M/S = 0, the device is slave for the SYNC0
When M/S = 1, the device is master for SYNC0.
If the device is slave, it must be slave for the clock.</td></tr>
</table>

34.6.17 SPI_I2C_usage register

RESET: all '0'

Table 719. SPI_I2C_usage register (Offset 0x44)

Bits	Name	Comments
[31:08]	Reserved	when Ux=0, the switched signal is I2C_SCL when Ux=1, the switched signal is SS0_SS
[07]	U7	
[06]	U6	
[05]	U5	
[04]	U4	
[03]	U3	
[02]	U2	
[01]	U1	
[00]	U0	

34.6.18 SPI_I2C_active

SPI_I2C active informs if the switched signal has to be out or not. When not used, the pin will out the value of SPI_I2C_usage. This register can then be used as extra GPIOs in output mode.

RESET: all '0'

Table 720. SPI_I2C_active (Offset 0x48)

Bits	Name	Comments
[31:08]	Reserved	when Ax=0, no signal is switched, Ux signal are out instead, when Ax=1, the switched signal is out.
[07]	A7	
[06]	A6	
[05]	A5	
[04]	A4	
[03]	A3	
[02]	A2	
[01]	A1	
[00]	A0	

34.6.19 I2S_CONF register

I2S_ conf register is used by the I2S module. It informs about the I2S frame topology.

RESET: all '0'

Table 721. I2S_CONF register (Offset 0x4C)

<table>
<tr><th>Bits</th><th>Name</th><th>Comments</th></tr>
<tr><td>[31]</td><td>I2S_IT</td><td>When I2S_IT =1, an I2S interrupt is in progress (after bank switching)</td></tr>
<tr><td>[30:28]</td><td>Reserved</td><td></td></tr>
<tr><td>[27]</td><td>BANK</td><td>Read only bit. Indicates the I2S memory bank being accessed by the I2S block to allow the processor to be synchronized with the samples memory.</td></tr>
<tr><td>[26]</td><td>FRC_BK</td><td>Force bank.
When FRC_BK = 0, the processor will access the two memory banks using address12-0.
When FRC_BK =1, the processor will access only the memory bank containing the samples of the previous buffer. Only processor address bits 11 to 0 are taken into account. Bit 12 is inserted by the I2S state machine.</td></tr>
<tr><td>[25:24]</td><td>DTo</td><td>Output transfer size. The goal is to left align the data received from the 32 bit transmit memory if necessary. According to the data size, the not used bits are filled with zero.
<table>
<tr><th>DTo1-DTo0</th><th>Transfer</th><th>Output Data</th></tr>
<tr><td>00</td><td>8->32</td><td>8 bits</td></tr>
<tr><td>01</td><td>16->32</td><td>16 bits</td></tr>
<tr><td>10</td><td>24->32</td><td>24 bits</td></tr>
<tr><td>11</td><td>32->32</td><td>32 bits</td></tr>
</table></td></tr>
<tr><td>[23]</td><td>invint</td><td>inversion of TDM internal sync when selected by intsel
when invint = 0, no action
when invint = 1, if TDM internal sync is selected by Intsel, it will be inverted before being used by the I2S block.</td></tr>
<tr><td>[22]</td><td>instel</td><td>select the TDM internal sync signal instead of the I2S generated one
when intsel =0, the generated I2S sync signal is used to the I2S block
when intsel = 1, the TDM internal sync signal is used instead the locally generated.</td></tr>
<tr><td>[21:12]</td><td>NS</td><td>Number of samples to recover before switching bank.</td></tr>
<tr><td>[11:10]</td><td>DTi</td><td>Input transfer size. The goal is to left align the data in the 32 bit receive memory if necessary (from din pin to memory). According to the data size, the not used bits are filled with zero.
<table>
<tr><th>DTi1-DTi0</th><th>Transfer</th><th>Input Data</th></tr>
<tr><td>00</td><td>8->32</td><td>8 bits</td></tr>
<tr><td>01</td><td>16->32</td><td>16 bits</td></tr>
<tr><td>10</td><td>24->32</td><td>24 bits</td></tr>
<tr><td>11</td><td>32->32</td><td>32 bits</td></tr>
</table></td></tr>
</table>

Table 721. I2S_CONF register (Offset 0x4C) (continued)

Bits	Name	Comments
[09:08]	Dw	Input data width. Only the valid bits will be shifted in the receive register and then right aligned. If this data needs to be left aligned in the memory, Tfs1-0 bits must be set. 00 8 bits 01 16 bits 10 24 bits 11 32 bits
[07:06]	Sw	Width of the SYNC signal low (SYNC signal high time is the same). 00 8 bits 01 16 bits 10 24 bits 11 32 bits
[05]	LBGxD	loopback at memory level *when GxD = 0,* there is no loopback, *when GxD = 1,* the data read from the memory (the one that is going to be sent) is copied in the receive register (the one that is going to be written in the memory). Then this loopback at the memory level will generate receive buffers identical to the transmit ones.
[04]	LBsoi	Switch output to input. *when Soi = 0,* no loopback is implemented *when Soi =1,* the output shift register is looped back inside the input register instead of the DIN pin. So, the receive sample is equal to the sent sample. This makes a loopback at the pad level.
[03]	LBsio	Switch input to output. *when Sio = 0,* no loopback is implemented *when Sio = 1,* the DIN pin is looped back to the DOUT pin instead of the data to be sent. This makes a loopback for the external entity.
[02]	ICLK	inverted clock. *When I_CLK is 0,* data is shifted out on the falling edge of the clock and shifted in the rising one. Synchronization signal will toggle on a falling edge. *When I_CLK is 1,* data is shifted out on the rising edge of the clock and shifted in the falling one. Synchronization signal will toggle on a rising edge.

Table 721. I2S_CONF register (Offset 0x4C) (continued)

Bits	Name	Comments
[01]	M/S	When M/S = '0' the device is slave and the pin I2S_CLK is an input.
[00]	ACT	*When ACT = 0*, the I2S cell is not used. No sample will be written/read in the memory that is available for the processor. *When ACT =1,* the I2S cell is active.

34.6.20 I2S_CONF2 register

I2S_ conf2 register is used by the I2S_interface module. It is a extension to I2S_conf register. It allows 8 and 16 bits data transfer, as well as DOUT impedance type management, additional interrupts management and specify the number of buffer memories allowed for the two banks.

Reset: all '0'

Table 722. I2S_CONF2 register (Offset 0x6C)

Bits	Name	Comments
[31:25]	Reserved	
[24]	ST_mode	Informs about the data width storage. Data width is know through DW1-0 bits of I2S_CONF register. *ST_Mode = 0*: data is stored/read always on 32 bits whatever the useful data size. *ST_Mode = 1:* data is stored/read according its size in the I2S memory.
[23]	MMdel	informs if the external LRCK generated in master mode must be delayed by one bit. *MMdel = 0:* LRCK is not delayed (Philips timing) *MMdel = 1:* LRCK is delayed by one bit (aligned delay)
[22]	Vndat	This bit informs about the data to be played on output pin, when there is no sample to play and that LZndat bit is at '1'. *Vndat = 0:* a '0' is played during non data time, *Vndat = 1:* a '1' is played during non data time
[21]	LZndat	This bit informs if the output pin must be low impedance or high impedance during non data time. *LZndat = 0:* pin is high impedance as long as non data is played. *LZndat =1:* pin is low impedance during non data is played and the its value is initialized in the Vndat bit.
[20]	HZdat	This bit informs if the output pin must be low impedance or high impedance during data is played. *HZdat = 0:* the pin is low impedance. *HZdat =1:* the pin is high impedance during a '1' and low impedance during a zero.
[19:08]	A	Address that will generate the IT_addr interrupt.
[07:04]	T	informs about which bit will generate the IT_tog interrupt. Allowed bit position is from 0 to 11.

Table 722. I2S_CONF2 register (Offset 0x6C) (continued)

Bits	Name	Comments
[03]	IT_tog	informs if an interrupt has to be generated when an address bit toggles from 0 to 1. Purpose of this bit is to generate an interrupt every N samples (N=2x), even if banks are greater.
[02]	IT_addr	informs if an interrupt has to be generated when a data is written at the A[11:0] address. *IT_addr = 0:* No interrupt is generated on address comparison *IT_addr = 1:* sample write address is compared to A[11:0]. When a sample is written at address A11-0, an interrupt is generated.
[01]	nIT_BK	informs if the bank switching (set by the I2S_CONF register) interrupt is masked. The interrupt is generated after those banks are switched. *nIT_BK = 0:* bank switching generates an interrupt *nIT_BK = 1:* bank switching is masked
[00]	MEM	Informs about the number of 1024*32 memories available for data buffering. *MEM=0:* 2 memories are available *MEM=1:* 1 memory is available

34.6.21 I2S_CLK_CONF register

This register allows generation of the internal clock that will be used as the reference for the I2S interface (I2S CLK).

Reset: all '0"

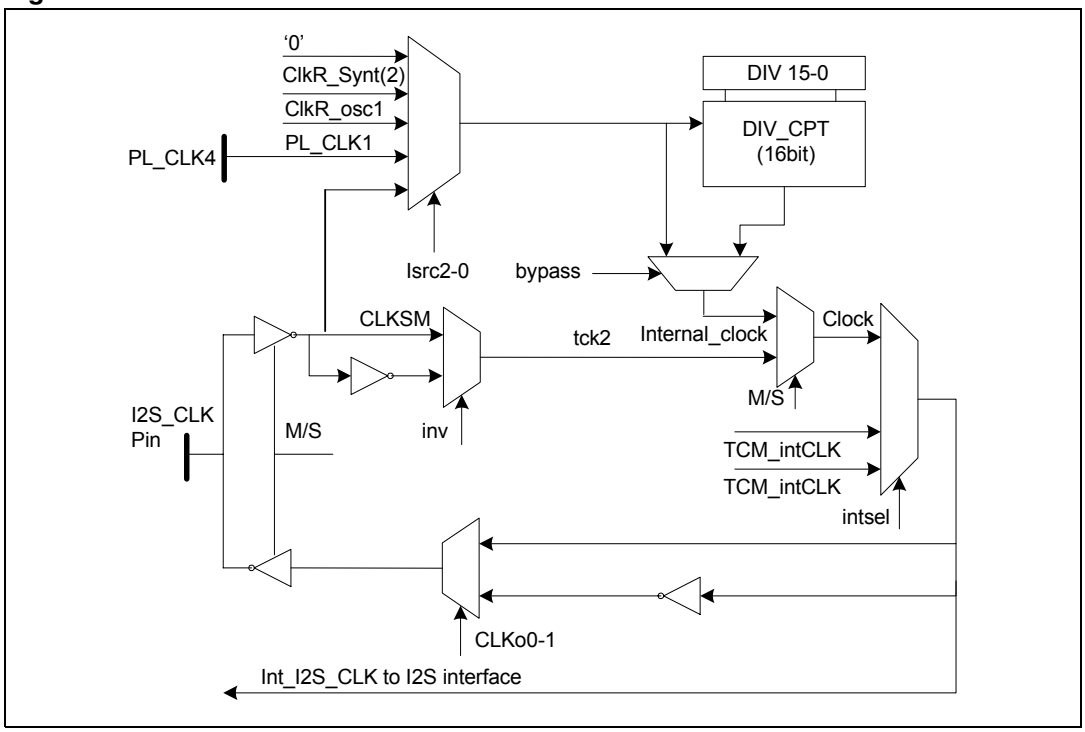
Table 723. I2S_CLK_CONF register (Offset 0x50)

Bits	Name	Comments	
[31:27]	Reserved		
[26]	ACT	activates the internal I2S_CLK clock that is sent to all other blocks. This initialization is mandatory when using any telecom function *ACT = 0*: internal I2S_CLk is always 0 for I2S_interface blocs *ACT = 1*: internal I2S_CLK is either the generated internal clock or the slave external clock, and is sent to the I2S_interface block.	
[25:24]	CLKo	Clock out select bits -	
		Clko	I2S_CLK
		00	0
		01	Clock
		10	/Clock
		11	0

Table 723. I2S_CLK_CONF register (Offset 0x50) (continued)

Bits	Name	Comments
[23]	Invint	inversion of TDM_CLK when selected by intsel *when invint =0,* no action *when invint =1*, if TDM_CLK is selected by Intsel, it will be inverted before being sent to the I2S block.
[22]	Intsel	select the TDM clock instead of the I2S clock *when intsel = 0*, the generated I2S_CLK is sent to the I2S bloc *when intsel = 1*, the TDM_CLK is sent instead
[21:19]	Isrc	selection of the input source for the 7 bits divider 000 → 0 001 → CLKSM (pin I2S_CLK) 010 → ClkR_Synt(2) 011 → ClkR_osc1 100 → PL_Clk4 others → 0
[18:03]	Div	this value will be compared to the divider counter value. When if will match, the generated signal will toggle. The output of the divider stage is then the input frequency divided by (2*[D[15:0]+1]).
[02]	Bypass	To bypass divider for clock in master mode. *Bypass =1*, the selected input clock is directly used as "clock" for master mode. *Bypass =0*, the divider output is used as "clock" in master mode
[01]	Inv	To invert CLKSM for int_CLK generation. *Inv =0,* the CLKSM pin signal is directly used as int_CLK (normally used in slave mode) *Inv =1*, the CLKSM signal is inverted to generate the internal clock Int_CLK (in slave mode).
[00]	M/S	*M/S =0*, the device is in slave mode. The I2S_CLK pin is an input. *M/S =1*, the device is master and I2S_CLK is out on I2S_CLK pin.

Figure 105. I2S clock tree



34.6.22 Interrupt mask register

RESET: all '0'

Table 724. Interrupt mask register (Offset 0x54)

Bits	Name	Comments
[31:16]	Reserved	
[15:14]	Reserved	Always write 1 (mandatory).
[13:09]	Reserved	Always write 0 (mandatory).
[08]	IT_GPIO	Mask for interrupt from GPIO pins.
[07]	IT_KB	Mask for interrupt from keyboard.
[06]	Reserved	
[05]	Reserved	
[04]	Reserved	
[03]	ITtdm	Mask for interrupt fromTDM module.
[02]	ITi2s	Mask for interrupt from I2S module.
[01]	ITch	Mask for interrupt on change detected on IT bus (for SLIC management).
[00]	ITp	Mask for interrupt from IT bus if change detected is persistent for the time as programmed in PERS_time Register.

Bit is 0: Interrupt is masked

Bit is 1: Interrupt is unmasked

The request will be cleared by a dummy access (read or write) of a byte at the following addresses:

Table 725. Dummy access address

Interrupt	Dummy Access Address (byte)
ITp	0x5006_0000
ITch	0x5006_0001
ITi2S	0x5006_0002
ITtdm	0x5006_0003

- IT_KB is cleared by reading the pressed keyboard key.
- IT_GPIO is cleared by reading the GPIO register.

34.6.23 Interrupt status register

The interrupt status register shows the status of the raw interrupt request and the interrupt filtered by the mask.

RESET: all '0'

Table 726. Interrupt status register (Offset 0x58)

Bits	Name	Comments
[31]	Reserved	
[30]	IT	Interrupt sent to the interrupt controller.
[29:21]	Reserved	
[20]	IT_GPIO_raw	Raw interrupt request from GPIO pins.
[19]	IT_KB_raw	Raw interrupt request from keyboard.
[18]	Reserved	
[17]	Reserved	
[16]	Reserved	
[15]	ITtdm_raw	Raw interrupt request from TDM module.
[14]	ITi2s_raw	Raw interrupt request from I2S module.
[13]	ITch_raw	Raw interrupt request on change detected on IT bus (for SLIC management).
[12]	ITp_raw	Raw interrupt request from IT bus if change detected is persistent for the time as programmed in PERS_time Register.

Table 726. Interrupt status register (Offset 0x58) (continued)

Bits	Name	Comments
[11:09]	Reserved	Interrupt requests from IPs after filtering through interrupt mask register.
[08]	IT_GPIO	
[07]	IT_KB	
[06]	Reserved	
[05]	Reserved	
[04]	Reserved	
[03]	ITtdm	
[02]	ITi2s	
[01]	ITch	
[00]	Itp	

When a source generates an interrupt request it is the processor that must clear it in the interrupt handler.

The above interrupts are ORed to generate interrupt on RAS_INT_out(0) - IRQ28.

34.7 Action memory content description

The action memory contains for any timeslot the necessary parameters to use this timeslot, both for input and for output.

The action memory is defined in the address range - 0x5001_0000 to 0x5001_0FFF.

Each word (corresponding to a specific time slot) of the memory is described as -

RESET: all '0'

Table 727. Action memory

Bits	Name	Comments
[31:29]	Reserved	
[28]	LSBin	informs if the DIN data must be latched first bit in MSB (shift left) or first bit in LSB (shift right). LSBin = 0**:** data is received MSB first (usual in voice) LSBin = 1**:** data is received LSB first.
[27]	LSBout	informs if the DOUT data must be MSB in first bit (shift left) or LSB in first bit (shift right). LSBout = 0**:** data is sent MSB first (usual in voice) LSBout = 1**:** data is sent LSB first.

Table 727. Action memory (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Comments</th></tr>
<tr><td rowspan="10">[26:24]</td><td rowspan="10">Chn</td><td>Indicates how many channels are present. This further implies how many valid bits of Ch[3:0] are used to indicate the channel number in the action memory. It should be the same for all the active channels.
<table>
<tr><th>Chn2</th><th>Chn1</th><th>Chn0</th><th>Valid Bits of Ch (3:0)</th><th>Number of channels</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>-</td><td>-</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>-</td><td>Upto 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>Upto 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3:0</td><td>Upto 16</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1:0</td><td>Upto 4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>3:0</td><td>Upto 16</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>3:0</td><td>Upto 16</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>2:0</td><td>Upto 8</td></tr>
</table></td></tr>
<tr><td>[23:22]</td><td>Off</td><td>These bits informs about the offset in bytes, at which data has to be stored.
They are used only for buffered channels.
For narrow band (8 bit), these bits are not relevant.
For wideband (16 bits), the value can be 0, 1, 2 or 3 (Off0 and Off1 are used).
In other cases the value can be 0 or 1 (Only off0 is used).</td></tr>
<tr><td>[21:20]</td><td>Ns</td><td>These bits inform about the number of bytes to be stored in a frame for buffered channels. The address will then be generated by Off1-0. NS must be the same for all the descriptions of bytes relative to a specific channel.So if a buffer channel is associated to more than one TS, then value of NS should be same for all associated TS.
<table>
<tr><th></th><th></th><th>Ns1</th><th>Ns0</th><th>Number of bytes per frame</th></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>1 bytes</td></tr>
<tr><td></td><td></td><td>1</td><td>0</td><td>2 bytes</td></tr>
<tr><td></td><td></td><td>1</td><td>1</td><td>4 bytes</td></tr>
</table></td></tr>
<tr><td>[19:16]</td><td>Ch</td><td>Channel number for the considered timeslot. It can be from 0 to 15.</td></tr>
</table>

Table 727. Action memory (continued)

Bits	Name	Comments
[15]	LTSS	This bit is used to inform that the last timeslot is switched on timeslot 0. As there is no time to store it in the memory and read back, the sample will go directly from the shift-in register to the shift-out register. It will anyway be stored in the switching memory, but not used. It is valid only for switched timeslot. When LTSS = 0, the event has not occurred. Input data follows the normal flow. When LTSS = 1, the event occurred.
[14:05]	Sts	Denotes the timeslot number of the previous frame to be played out during this timeslot.
[04]	Bin	The data has to be buffered in the channel (denoted by bits **Ch[3:0]**) when Bin = 1.
[03]	Sin	The data has to be stored in the switching memory when Sin = 1.
[02]	Bout	If this bit is set, then the data played will come from the buffered channel.
[01]	Sout	This bit if set tells that the data played will come from the switched channel. If both Bout and Sout are set, the data will come from the buffered channel.
[00]	LowZ	If LowZ = 1'b0, the timeslot will be high impedance on the DOUT pin. Otherwise normal operation.

34.7.1 Int block

This block collects the interrupt request of several IPs and then merges them on the interrupt 0 line of the interrupt controller. It is the responsibility of the interrupt handler to determine the root of the interrupt.

The interrupt 0 line is shared by events occurring on the following IPs:

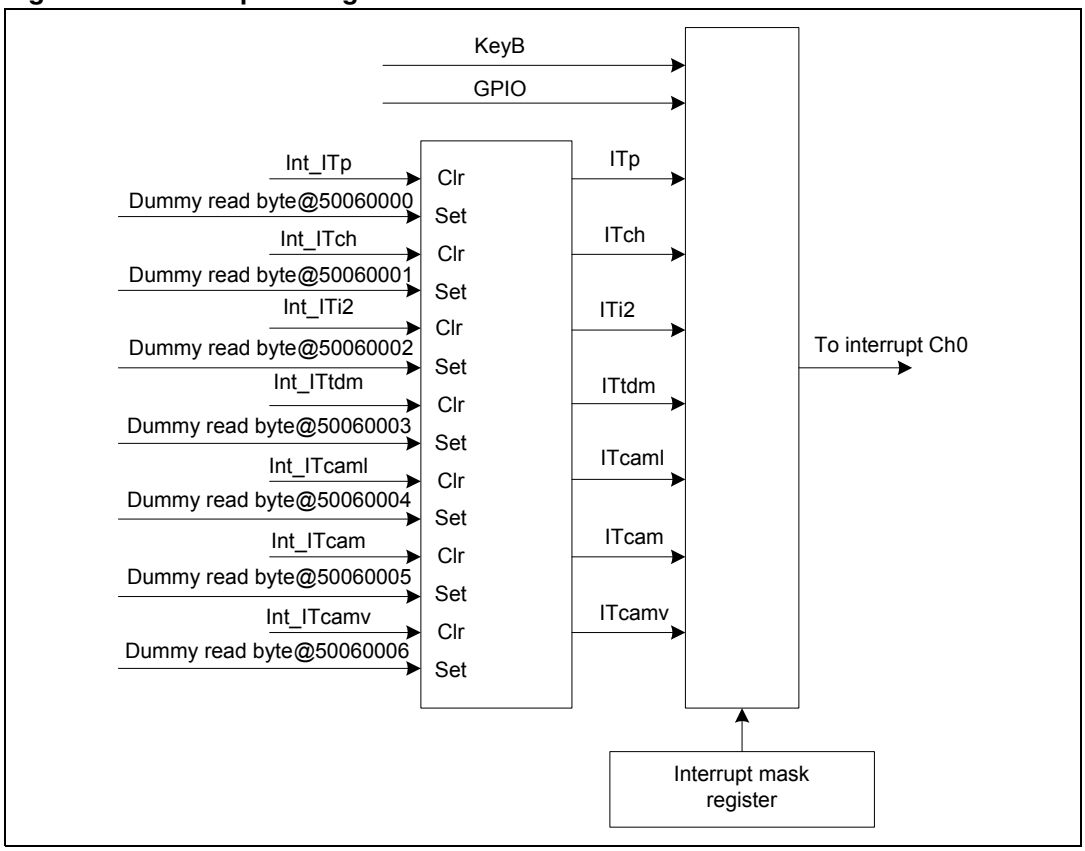
- Keyboard
- GPIOs
- IT bus
  - Change on these pins
  - Persistence of change on these pins
- I2S (buffer bank switching)
- TDM (buffer bank switching)

The Keyboard and GPIOs interrupts are cleared by writing to keyboard or GPIO registers. All interrupt requests can be cleared by a dummy read/write access to register 0X5006_000x as shown in figure 21.

The interrupts can be masked through interrupt mask register at 0x5000_0054.

Interrupts are ORed to generate interrupt on RAS_INT_out(0) - IRQ28

Figure 106. Interrupt management



35 RS_Keyboard controller

35.1 Overview

Within its Reconfigurable Array Subsystem, SPEAr300 provides a GPIO/Keyboard block which is a two-mode input and output port.

In summary, it provides:

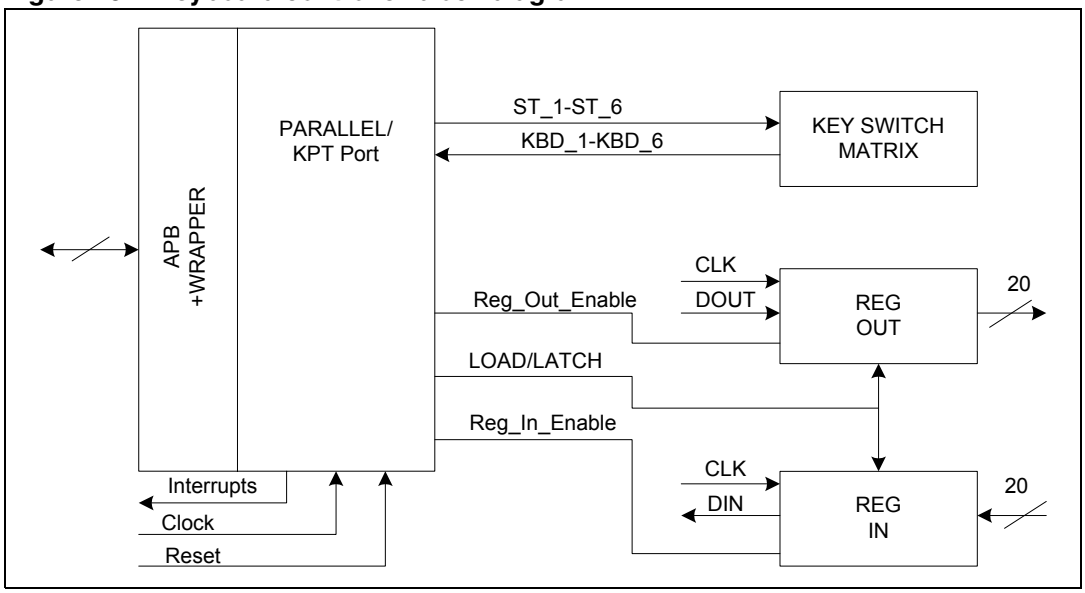
- 18 bit general-purpose parallel port with input or output single pin programmability.
- 81 key Keyboard (9x9 matrix).

The selection between the two modes is an APB Bus programmable bit.

35.2 Functional description

This section describes the functioning of the keyboard controller.

Figure 107. Keyboard controller block diagram



35.2.1 General purpose input output interface

When GPIO mode is selected, it is possible to program through APB Bus each of the 18 signals available, as output or not (they are always inputs). Once programmed, each pin maintains its identity as an input or output. The default on power up is that all pins are set to inputs.

The Mode Control bits in the Mode Control Register must be set to [01] to enable the GPIO mode.

The ARM may read or write to the Port register at any time. Reading this register shall provide the status/values on all of the pins, inputs and outputs.

35.2.2 Keyboard interface

When Keyboard mode is selected, it is possible to read from APB Bus the value of externally connected keyboard, scanned at programmed rate.

The keyboard may contain up to 81 keys. 18 port pins provide a 9x9 scanning matrix. 9 of the pins are strobes and nine (9) of the pins are inputs.

The circuitry shall scan the keys at a rate of 10, 20, 40 or 80 msecs, controlled by the software. Two successive cycles are needed to validate a key. Only one key will be allowed down in a scan cycle. Every valid key condition will cause the value of the key to be written to a register and an interrupt shall be set. The key value is coded on eight bit; the lower nibble refers to the column number (0, 1,2…8) while the higher nibble gives the row number (0,1,2…8) of the key-pressed.

Control Register bits b3 and b2 determine keyboard scanning rate. Each time the timer expires, the keyboard is scanned. If only one key down is detected and it is the same key as on the previous scan, a bit is set in the Status register indicating New Key Data. The code for the key is written to the Keyboard Value register. Key release is signaled only once.

The keypad encoder initialization is made one time when the application starts (prescaler load value, keyboard enable, scan rate, keyboard operation mode), and then the software handles the interrupt line in order to process Keyboard interrupt.

35.3 Programming model

35.3.1 External signals

Table 728. External signals

Port pin	GPIO	Keyboard
ROW0	GPIO 0	output kbd(row)0
ROW1	GPIO 1	output kbd(row)1
ROW2	GPIO 2	output kbd(row)2
ROW3	GPIO 3	output kbd(row)3
ROW4	GPIO 4	output kbd(row)4
ROW5	GPIO 5	output kbd(row)5
ROW6	GPIO 6	output kbd(row)6
ROW7	GPIO 7	output kbd(row)7
ROW8	GPIO 8	output kbd(row)8
COLUMN0	GPIO 9	input kbd(column)0
COLUMN1	GPIO 10	input kbd(column)1
COLUMN2	GPIO 11	input kbd(column)2
COLUMN3	GPIO 12	input kbd(column)3
COLUMN4	GPIO 13	input kbd(column)4
COLUMN5	GPIO 14	input kbd(column)5

Table 728. External signals (continued)

Port pin	GPIO	Keyboard
COLUMN6	GPIO 15	input kbd(column)6
COLUMN7	GPIO 16	input kbd(column)7
COLUMN8	GPIO 17	input kbd(column)8

35.3.2 Register map

The GPIO/Keyboard block can be fully configured by programming its registers which can be accessed at the base address 0xA000_0000. *Table 729* shows the address map of them.

Table 729. Register map

Name	Offset	Type	Size in bit	Description
MDCTRLREG	0x00	RW	16	Mode Control Register
GPIODIRREG	0x04	WO	18	Direction Register for GPIO outputs
GPIODATAREG	0x08	RW	18	Data Register for GPIO input/outputs
STATUSREG	0x0C	RW	2	Status Register
KBREG	0x10	RO	8	Keyboard Value Register

35.3.3 Registers description

35.3.3.1 MDCTRLREG register

Table 730. MDCTRLREG register bit assignments

Bit	Name	Reset value	Type	Description
[31:16]	Not used			
[15:09]	PCLKFREQ	7'h00	RW	Prescaler programmable value. The value is the APB frequency (in MHz) minus 1.
[08]	KBSCANFLG	1'h0	RW	When set to '1' enable keyboard scanning.
[07:04]	Reserved	-	-	Reserved. Read undefined. Should be written 0.
[03:02]	SCANRATE	2'h0	RW	Keyboard scan rate: – 2'b00 - 10 ms – 2'b01 - 20 ms – 2'b10 - 40 ms – 2'b11 - 80 ms
[01:00]	MODECTRL	2'h0	RW	Mode selection: – 2'b00 - inactive (reset value) – 2'b01 - GPIO mode – 2'b10 - Keyboard mode – 2'b11 - Not used

35.3.3.2 GPIODIRREG register

When GPIO Mode is enabled, the value in this register specifies whether the particular I/O pin is an output or not. Writing a logic1 to the register bit sets the port pin to output mode; otherwise the related pin is considered as only input.

Table 731. GPIODIRREG register bit assignments

Bit	Name	Reset value	Type	Description
[31:18]	not used			
[17:00]	GPIODIR(n)	18'h0	WO	When set to '1', related pin is activated as output. The output pin value is determined by Data Register content.

35.3.3.3 GPIODATAREG register

This is a read/write (RW) register. Writing to the register will set the output pin to the value of this register, only for pins that are programmed as outputs. Input pins are unaffected. Reading this register transfers port values, including both the input and output values.

Table 732. .GPIODATAREG register bit assignments

Bit	Name	Reset value	Type	Description
[31:18]	Not used			
[17:00]	GPIODATA(n)	18'h0	RW	WRITE: When set to '1' and if activated as output, related pin goes to high level, otherwise is at low level. If not activated as output, is meaningless for related pin. READ: all pin values whether input or output.

35.3.3.4 STATUSREG register

In Status Register only bit [1] is meaningful; it is set to '1', by logic, every time new data is available from keyboard. This bit should be written to '0', once the data has been read from KBREG. For interrupt handling this write operation is necessary to reset interrupt line.

Status register is a two bit register.

Table 733. STATUSREG register bit assignments

Bit	Name	Reset value	Type	Description
[31:02]	Not used			

Table 733. STATUSREG register bit assignments (continued)

Bit	Name	Reset value	Type	Description
[01]	KBNEWDATA	1'h0	RW	This bit is set to '1' when a new keyboard value is available in KBREG. Once data is read from KBREG, this bit should be reset to '0'.
[00]	-	-	-	Reserved. Read undefined. Should be written 0.

35.3.3.5 KBREG

It is an eight bit register and its content is the keyboard key-code value.KBREG bit assignments are given in *Table 734*. The Key-code values are given in *Table 735*.

Table 734. KBREG register bit assignments

Bit	Name	Reset value	Type	Description
[31:08]	Not used			
[07:00]	KBPRDATA	8'hFF	RO	Key-code value

Table 735. Key-code table (hex values)

	COL(0)	COL(1)	COL(2)	COL(3)	COL(4)	COL(5)	COL(6)	COL(7)	COL(8)
ROW(0)	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08
ROW(1)	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18
ROW(2)	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	0x28
ROW(3)	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37	0x38
ROW(4)	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47	0x48
ROW(5)	0x50	0x51	0x52	0x53	0x54	0x55	0x56	0x57	0x58
ROW(6)	0x60	0x61	0x62	0x63	0x64	0x65	0x66	0x67	0x68
ROW(7)	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77	0x78
ROW(8)	0x80	0x81	0x82	0x83	0x84	0x85	0x86	0x87	0x88

36 RS_General Purpose Input Output (GPIO)

Please refer to *Chapter 18: BS_General purpose input/output (GPIO)* for general details of this IP.

36.1 Application Notes

1. Eight individually programmable input/output pins are available (instead of only 6 as in the fixed part GPIO block described in Chapter 17). Hence, width of GPIO Data Direction Register, Interrupt Control Registers and Mode Control Register is 8bit (7:0).
2. GPIODATA Register appears at 256 locations. Hence offset of this register ranges from 0x000 to 0x3FC.
3. No GPIOs in this block are dedicated for any SPI functionality.
4. IT_GPIO(Bit 8) of Telecom Interrupt Mask Register(0x50000054) should be set to generate an interrupt for RAS GPIO.

37 Power and clock management

37.1 Overview

Power consumption is an important design aspect of any modern system. Power management techniques allows to reduce power consumption ensuring requested performance by utilization.

37.1.1 Power management techniques

System Control State Machine is a device feature designed to support reduction of power consumption controlling clock inputs to the CPU. It presents four state:

- SLEEP
- DOZE (reset state)
- SLOW
- NORMAL

All transactions between states are software controllable except for SLEEP to DOZE that is activated only by a hardware event.

The following items describe the power management techniques:

- **Dynamic Frequency Scaling** applicable in NORMAL state

This technique uses dynamic selection of the optimal frequency to allow a task to be performed in the required amount of time.

As described in *Chapter 11: Clock & reset system* chapter PLL1 (Sys) provide frequency to the system. By default also DRAM is driven by PLL1, this mode is called **synchronous DRAM.** It is possible to use PLL2 to drive DRAM, this mode is called **asynchronous DRAM.**

In asynchronous DRAM mode it is possible to change system frequency (PLL1) without affecting DRAM works, for this reason to apply Dynamic Frequency Scaling System has to work in asynchronous DRAM mode.

Frequency changes are applied with small delay, due to PLL lock time (see formula below to calculate this delay)

- **Dynamic Clock Switching**

It is possible to dynamically switch off, or on, the clock to modulate group in accordance with their use.

This operation is performed without delay.

- **Combining Frequency Scalling and Clock techniques**

In NORMAL state the best active power saving is obtained by combining the power management techniques previously described.

These techniques is aimed at reducing power consumption of a device allowing a fast response to critical task (that can be performed always at maximum frequency, if needed).

- **Statically Frequency Selection and Clock Switching OFF**

In a well known system it is possible to define statically, based on performances defined in the manufacturing process, the operating frequency and the group that do not need clock.

This technique is easier from a designer prospect for software development and offer a well known consumption. It is recommended that when performance required is without critical task and it is sufficient to guarantee an average power computation.

37.2 System control state machine

System control state machine is used to select the input frequency to apply to the system.

Mainly three selections are available:

- MAIN Oscillator: directly 24 MHz or its ratio (1:2, 1:4, 1:16 or 1:32)
- RTC Oscillator, if present its 32.768 kHz
- PLL1 Frequency, generated from MAIN Oscillator

The following sections describe operative System Control State (note that only operating states are described here, for all intermediate states referred to *Chapter 14: BS_System controller*)

See register *Section 14.4.3: SCCTRL register* to change System states.

Figure 108. Operative system control states

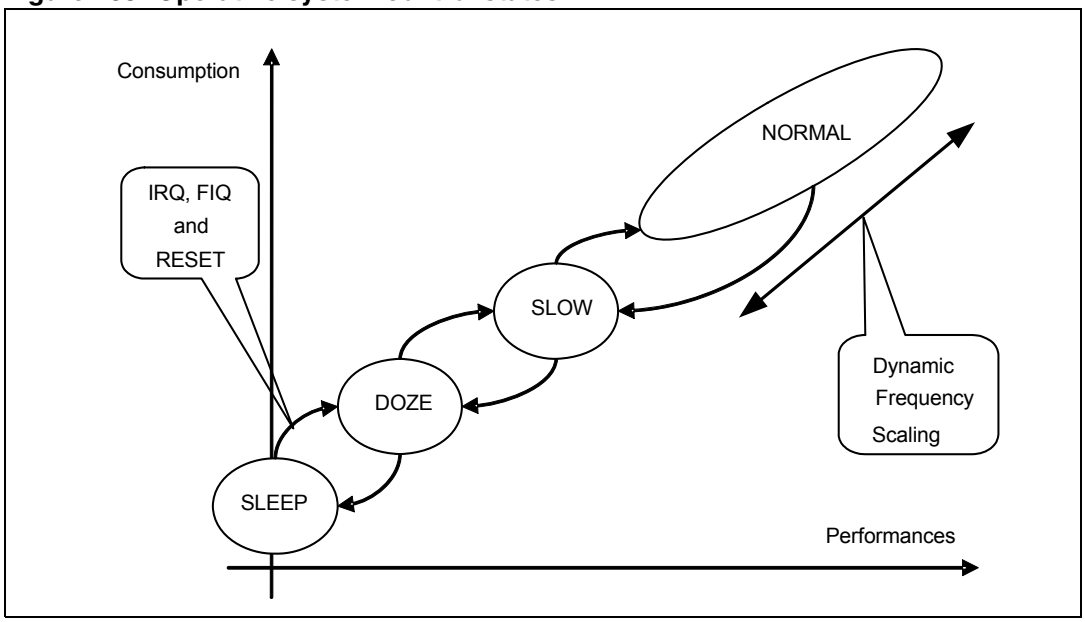


Table 736. Power state for synchronous DRAM system (DRAM clocked by PLL1)

State	ARM	ARM clock	DRAM	Possible code execution memory
SLEEP	Hibernate	Off	Self refresh	None
DOZE	Running Running	RTC Osc MAIN Osc(PLL off)	Self refresh Self refresh	Internal memory Internal memory
SLOW	Running	MAIN Osc.(PLLoff)	Self refresh	Internal memory
NORMAL	Running	PLL1 (Up to 333 MHz)	Active	Internal memory and external DRAM

Table 737. Power state for asynchronous DRAM system (DRAM clocked by PLL2)

State	ARM	ARM clock	DRAM	Possible code execution memory
SLEEP	Hibernate Hibernate	off off	Self refresh Active	None None
DOZE	Running	RTC Osc.	Self refresh	Internal memory
	Running	RTC Osc	Active	Internal memory and external DRAM
	Running	MAIN Osc. (PLL off)	Self refresh	Internal memory
	Running	MAIN Osc.(PLL off)	Active	Internal memory and external DRAM
SLOW	Running	MAIN Osc. (PLL off)	Self refresh	Internal memory
		MAIN Osc.(PLL off)	Active	Internal memory and external DRAM
NORMAL	Running	PLL1 (Up to 333 MHz)	Active	Internal memory and exernal DRAM

37.2.1 SLEEP

In SLEEP state clock is not provided to CPU. This state maximize power saving.

System Controller clock is driven by the last selected source in DOZE mode, it could be RTC or MAIN oscillator.

On interrupt request, normal (IRQ) or fast (FIQ), CPU wake-up and go in DOZE. Few clock cycles (less than five) are required for this transition.

It is recommended, that to reduce power consumption, switch off all clocks to modules which are not used for wake-up purpose (see *Section 37.6: Statiscally frequency selection and clock switching OFF*).

Interrupts enabling wake-up from SLEEP state are:

- Ethernet MAC. In this case it is possible to disable clock to Ethernet MAC (PERIP1_CLK_ENB.gmac_clkenb) using external clock provided by PHY MAC.
- USB device. In this case clock to USB device cannot be switched off (PERIP1_CLK_ENB.usbdev_clkenb) and AHB since the resume interrupt is registered by HCLK.
- RTC. All clocks to internal modules can be switched off (except for RTC).
- GPIO. All clocks to internal modules can be switched off (except for GPIO).
- TIMER. All timers, if timer clock is not switched off (see *Section 13.4.12: PERIP1_CLK_ENB register* for timer clock sources)

Note: Sleep state is only activated if SCCTRL Mode Ctrl is set to zero and processor is in Wait-for-Interrupt state.

37.2.2 DOZE (reset state)

DOZE is the first state activated after reset.

In this state CPU is running with RTC or MAIN Oscillator, according to bit set in PRPH_CLK_CFG.rtc_disable. After reset MAIN Oscillator is selected (that allows to have systems without RTC Oscillator).

It is possible to select a division of MAIN Oscillator frequency through CORE_CLK_CFG.osci24_div_en to enable and CORE_CLK_CFG.osci24_div_ratio bits to select ratio.

Better results, to reduce power consumption, are achieved with RTC Oscillator.

Code execution

In DOZE state code has to run from internal memory (Boot ROM, internal RAM, Cache and TCM if present) or from external memory Serial Flash or DRAM. To run from DRAM it is necessary to use PLL2 for the SDRAM controller (asynchronous DRAM mode) with a frequency higher than minimum supported by used DRAM.

It is also recommended, if wake-up response allows it, to put DDR in self refresh mode.

Transition to other state is software controlled through SCCTRL ModeCtrl bits. It allows program directly in the NORMAL state, in this case the hardware will execute transitioning as several steps. Less than five clock cycles are required to change from one state to the other.

37.2.3 SLOW

In SLOW state MAIN Oscillator is used to clock CPU.

It is possible to select a division of MAIN Oscillator frequency through CORE_CLK_CFG.osci24_div_en to enable and CORE_CLK_CFG.osci24_div_ratio bits to select ratio.

In SLOW state there are the same constrains of DOZE state for code execution.

Transition to other state is software controlled through SCCTRL Mode Ctrl bits.

It is allowed to program DOZE state and nothing else is required

To go in NORMAL state, it is necessary to program PLL1 at the desired frequency. PLL has to be stable before switch in NORMAL, two ways are possible:

- Controlled by software, verifying PLL1_CTR.pll_lock bit, applicable if Dither is disable (PLL1_CTR.pll_control1.DitherMode).
- Controlled by Hardware. An intermediate hardware state waits for PLL stabilization using a pre programmed delay, use SCPLLCTRL.PllTime for time delay with SCPLLCTRL PllOver disabled. See formula in *Section 37.3: Dynamic frequency scaling* to calculate the proper delay.

37.2.4 NORMAL

In NORMAL state it is possible to apply the power management techniques described in following sections.

Table 738. Techniques applicable in NORMAL state

Technique	Synchronous DRAM	Asynchronous DRAM
Dynamic Frequency Scalling (DFS)	Denied	Allowed
Dynamic Clock Switching (DCS)	Allowed	Allowed
Combining DFC+DCS	Denied	Allowed
Statically Frequency Selection and Clock Switching OFF	Allowed	Allowed

37.3 Dynamic frequency scaling

Dynamic Frequency Scaling (DFS) is generally used when the work-load is not CPU-bound. It reduces processor's instructions in a given amount of time, thus reducing performance but also consumption. It is very efficient to run briefly at peak speed and at a reduced clock rate for a long time.

It is possible to change PLL frequency in NORMAL state, but it generates undesirable frequency overshoot/undershoot. To avoid this it is better to switch in Slow state, change PLL frequency, disable Dithering (if enabled), wait for PLLlock signal, switch again in Normal mode and enable again Dithering (if it was originally enabled).

With the following formula it is possible to calculate delay time introduced by PLL for frequency changes.

Lock time = 4ms/(decimal equivalent of PLL Charge Pump bit setting+ 1)

PLL Charge Pump bits are PLL1_CTR.CP

So, for example with CP = 01110 = 14 (decimal)

Lock time = 4ms/15 = 267 us

There are two way to wait PLL stabilization, software and hardware, look at section *Section 37.2.3: SLOW* for details.

37.4 Dynamic clock switching

Like DFS, Dynamic Clock Switching (DCS) is a power-management technique aimed at reducing active power consumption of a device, whereas DFS change frequency of all modules using CLK_Pll1 signal. DCS could switch OFF completely the clock of an unused modules and quickly switch ON when its use is required.

With this technique the processor, or system, could run at maximum frequency maintained highest performances.

To have maximum flexibility DCS is fully software controlled through *Section 13.4.12: PERIP1_CLK_ENB register* in *Chapter 13: Miscellaneous registers (Misc)*

DCS is useful when a real-time application is waiting for an event. The system can switch OFF clock of modules not used and enable them, with a low latency, when needed.

Modules that support this feature are shown in *Table 739*

Table 739. Modules supporting DCS technique

Module	Module	Module
C3	GPIO	SPI
SDRAM	RTC	UART
USB 2.0 host	ADC	ARM subsystem
USB 2.0 device	Timer 2	ARM
Ethernet	Timer 3	
Flash serial (SMI)	IrDA	PLL1
Internal ROM	JPEG codes	PLL2
DMA	12C	PLL3

37.5 Combining frequency scaling and clock switching techniques

The best active power saving is obtained by combining power-management techniques previously described; there are no limitation to do that.

37.6 Statiscally frequency selection and clock switching OFF

With this technique it is possible statically (based on performance points predefined in the manufacturing process of a given device) define the frequency and activate only modules requested by the application.

This technique is to apply when a constant consumption is requested by the system.

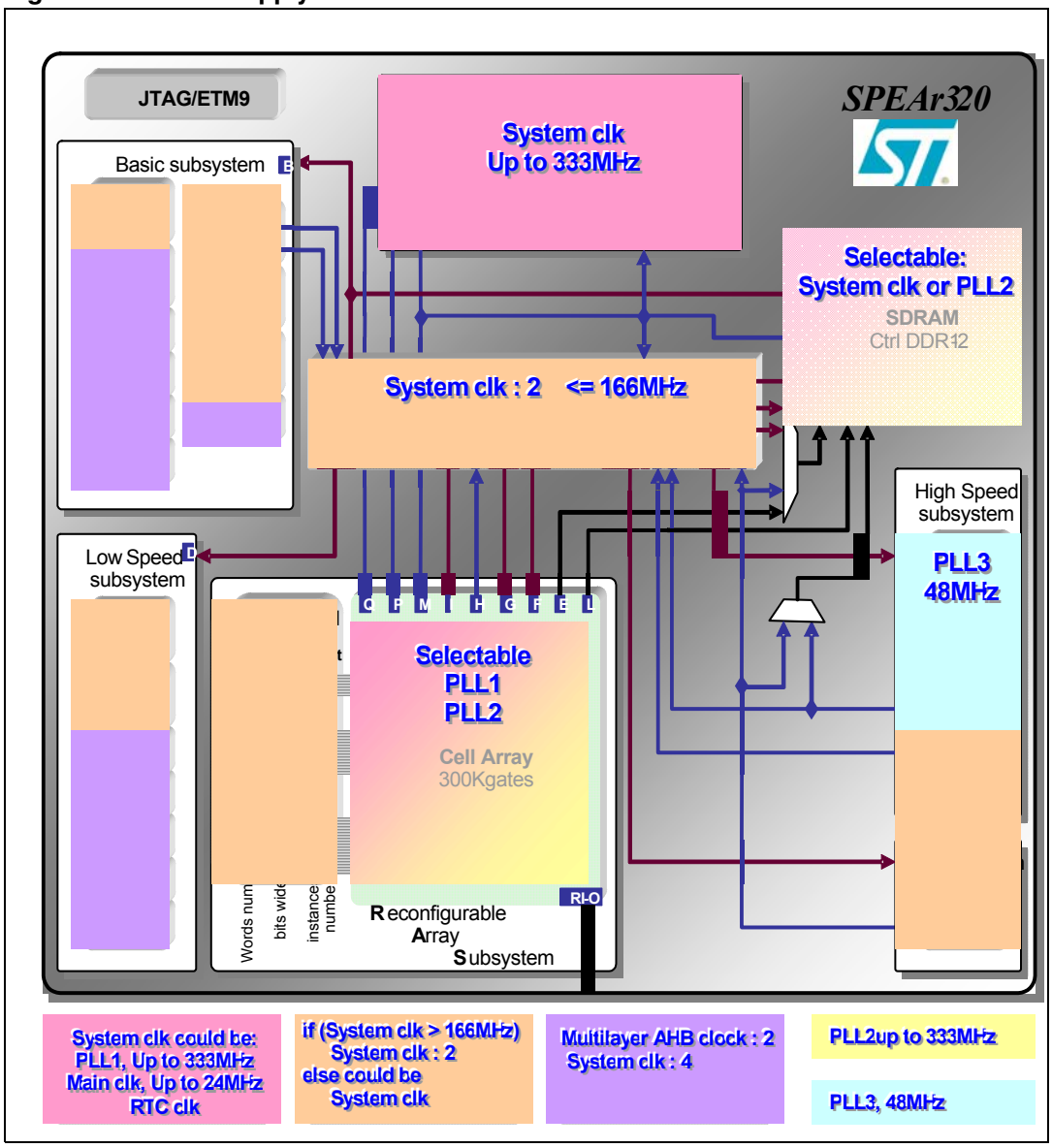
It is useful when USB ports are not requested switch of USB PLL (PLL3) and peripheral clock could be attached to PLL1, with the right prescaler.

Also PLL2 should be OFF in synchronous mode.

37.7 PLLs usage

The following diagram show frequency distribution in the system, with the clock domains generated by the different PLLs.

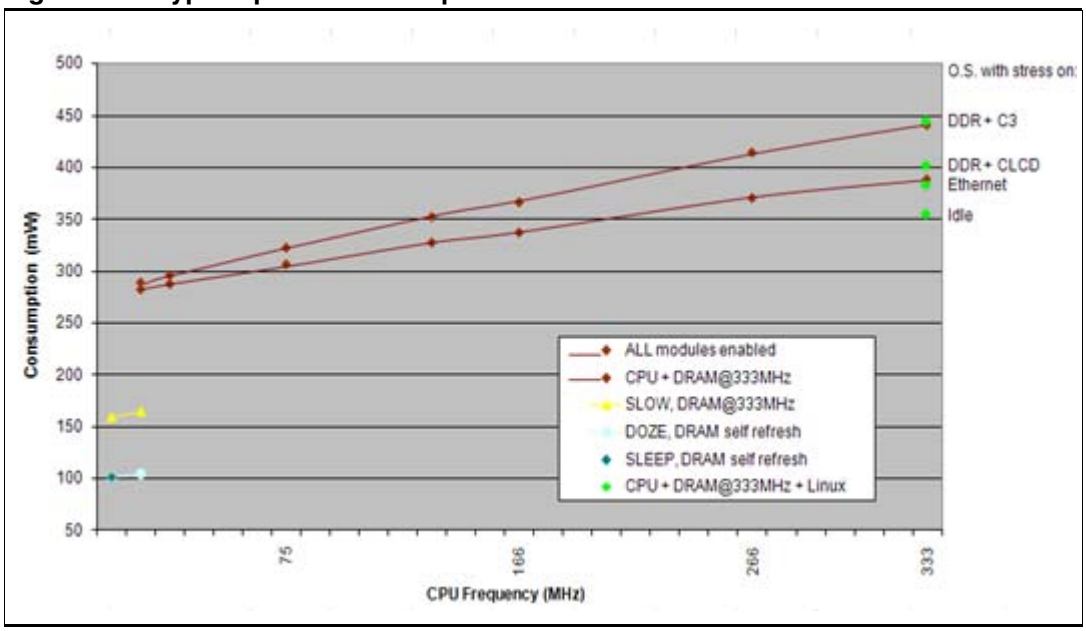
Figure 109. Clock supply



37.8 Power consumption

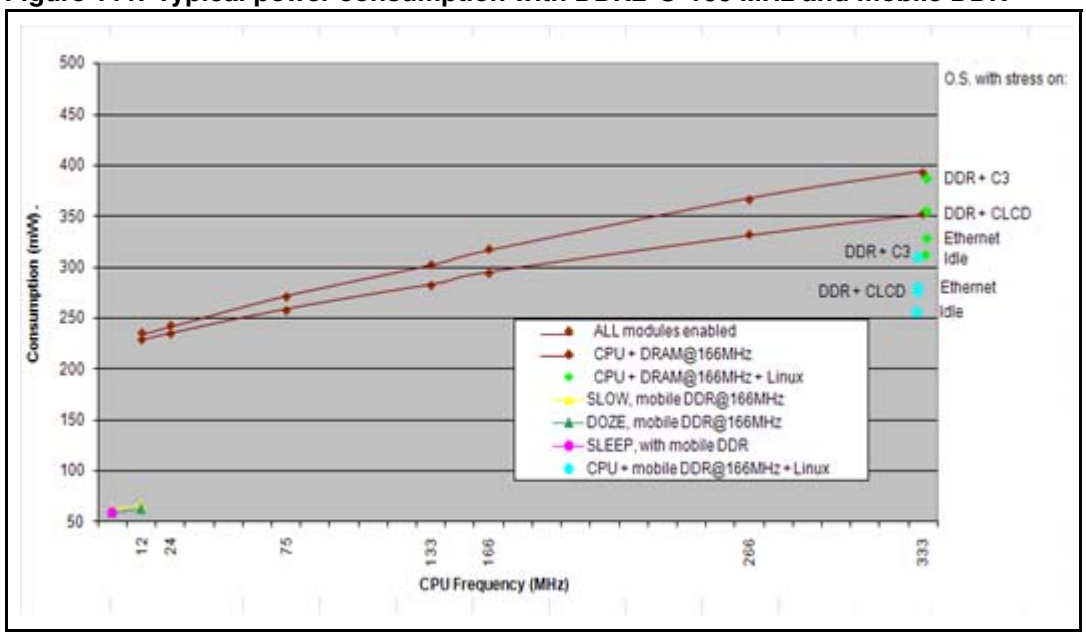
The following diagrams show different states (SLOW, DOZE and SLEEP) and, for NORMAL results of Dynamic Frequency Scaling and Statically Frequency Selection, Dynamic Clock Switching and Statically Clock off techniques. When DRAM is in self refresh, code is executed from internal RAM.

Figure 110. Typical power consumption with DDR2 @ 333 MHz



Note: Slow, Doze and Sleep mode values are obtained with USB ports in suspend mode and then power off USB PLL (PLL3) (setting register USB2_PHY_CFG.PLL-pwdn to 1)

Figure 111. Typical power consumption with DDR2 @ 166 MHz and mobile DDR



Note: Typical current and power values listed in this chapter are not guaranteed. These values are dependent on many factors including the type of application running, use of internal functional capability, external interface usage, case temperature and power supply voltages.

The following information provides details about the condition under which values could be obtained:

- Data based on characterization, results tested at nominal VDD
- Several STD selected SPEAr revision B, tested on the device demoboard with external power supply dedicated to SPEAr
- For NORMAL state DDR2 clocked by PLL2 @ 333 MHz (166 MHz below)
- UBOOT code running (except for Linux measures)
- Temperature ambience +25°C

Note: 1 Linux all modules clocked

2 Mobile DDR benefits power consumption reduction cross 1.8V power line. It is in the range of 15 to 35mA vs 50 to 90mA of a DDR2@166 MHz.

37.8.1 Modules power consumption

In the following table are reported power consumption values for different modules, there are also columns that show current consumption on different input voltages.

Table 740. Power and current consumption for modules

Module	Power mW CPU@333 MHz	1V2 current mA	1V8 current mA	2V5 current mA	3V3 current mA
ARM SDRAM	379	164	54	25.0	6.4
ARM+SDRAM+All USB ports	434	181	62	32.3	7.3
ARM+SDRAM+Ethernet	394	169	59	25.2	6.4
ARM+SDRAM+CLCD Ctrl	419	164	63	25.3	13.6
ARM+SDRAM+C3 Ctrl	449	195	72	25.2	6.4
ARM+SDRAM+All modules	531	212	84	32.3	13.6

Note: 3.3V current is primarily dependent on the capacitive loading, frequency and utilization of the external buses.

Table 741. Delta power consumption for modules

Module	Power mW CPU@12 MHz	Power mW CPU@24 MHz	Power mW CPU@ 75 MHz	Power mW CPU@ 133 MHz	Power mW CPU@ 166 MHz	Power mW CPU@ 266 MHz	Power mW CPU@ 332 MHz
ARM+SDRAM	283	289	305	325	335	362	379
All USB ports	16.8	16.9	28.6	37.5	42.3	535	55.7
Ethernet	1.7	2.0	2.6	4.6	7.8	14.4	15.0
CLCD Ctrl	15.4	15.6	35.8	36.4	36.5	40.4	40.4
C3	1.1	2.4	6.2	9.8	38.0	57.5	70.3
ARM and all IPs	359	362	371	417	434	483	531

Note: 1 Values reported are related to a system in NORMAL state setting in asynchronous mode, DRAM clocked by PLL2 at 333 MHz

*2 Absolute value in **bold** delta values in normal*

37.8.2 IPs power

All IPs are connected to Vcore (1.2Volt) to power the I/F logic with the internal buses, some IPs are also connected to other voltages. The following table describes it.

Table 742. IP voltage usage

Modules	Vcore1.2Volts	1.8Volts	2.5Volts	3.3Volts	Vbat1.5Volts
PLLs	ctrl		Osc		
ADC			Yes		
USB	ctrl		phy	phy	
DRAM	ctrl	padDDR2			
I/O pads				Yes	
RTC					Yes
All other logic (VCORE)	Yes				

38 BootROM

BootROM is a small piece of code that starts its execution just after the SoC exits from reset.

The following are the features supported by SPEAr300 BootROM:

- Boot from NOR serial Flash
- Boot from NAND Flash
- Boot from NOR parallel Flash
- Boot / Upgrade from USB
- Boot from UART
- Boot from Ethernet

First three are the normal ways of booting the software and requires to have a second-level boot software (Xloader) in NOR/NAND Flash.

Last three types (USB/UART/Ethernet) are meant to boot without Flash memories. USB boot is used to upgrade 1st to 3rd level boot in Flash memories.

Table 743. Booting types

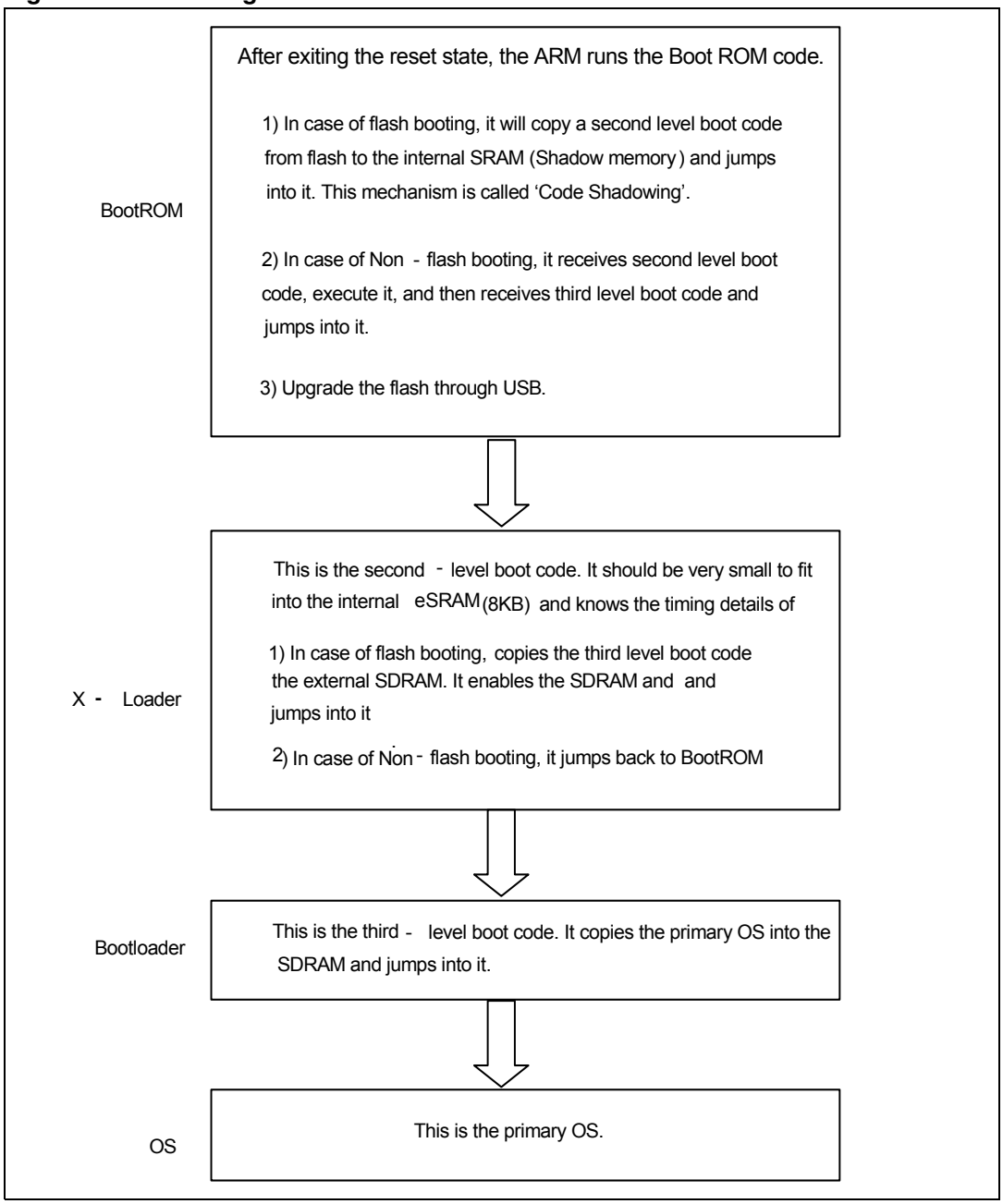
Booting using Flash memories	Booting without using Flash memories	Upgrading the Flash memories
Serial NOR	UART	USB
Parallel NOR	Ethernet	
Parallel NAND		

38.1 Boot Stages

Figure 112 describes boot stages on SPEAr SOC in more details. There are 4 stages of booting:-

- Boot stage 1 (BootROM)
- Boot stage 2 (Xloader)
- Boot stage 3 (Bootloader e.g. U-Boot)
- OS

Figure 112. Boot stages

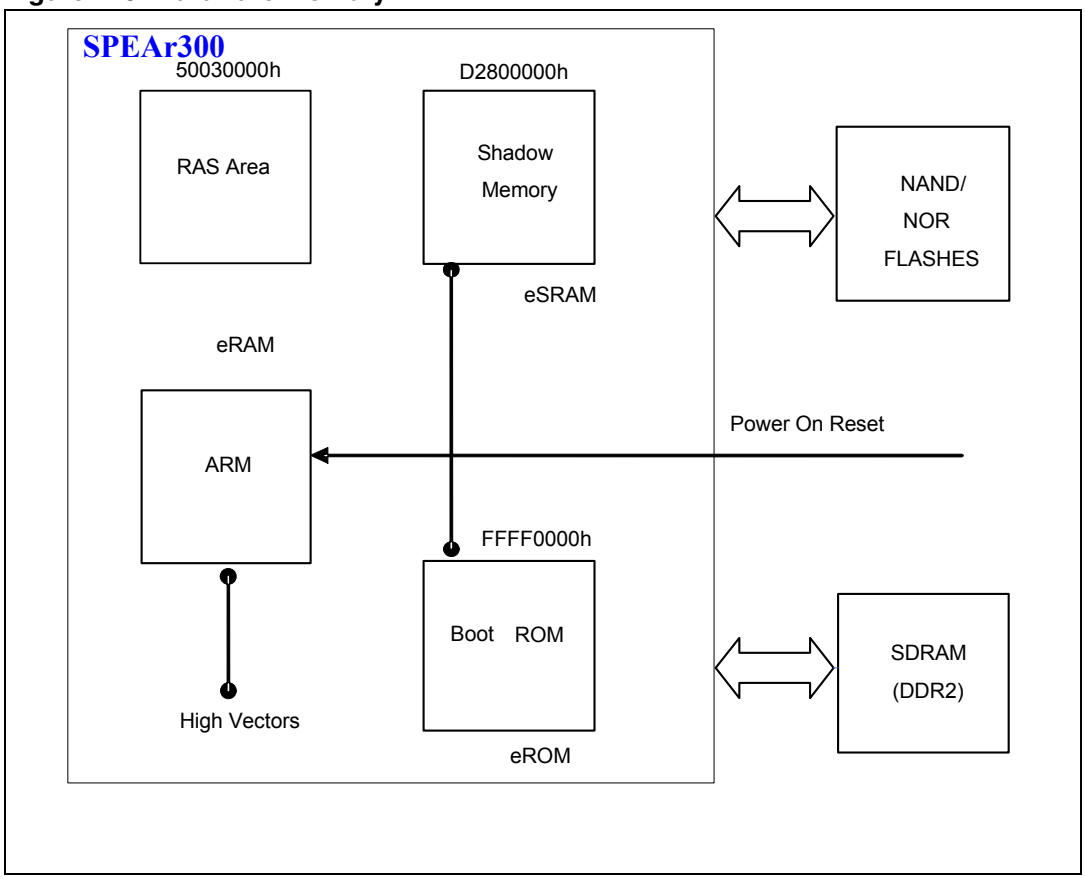


38.2 Booting Pins

Please look at section '*Section 34.6.1: Boot register*' of "*Chapter 34: RS_Telecom IP*

38.3 Hardware overview

Figure 113. Hardware memory



38.3.1 eROM (Embedded ROM)

eROM is the 32KB of area starting from 0xFFFF_0000. The ARM processor is mapped to HIGH vectors and starts executing instructions from 0xFFFF_0000.

38.3.2 Shadow memory

Shadow memory is that area, where BootROM copies the X-Loader after reading/receiving from any of the booting processes. Address where X-Loader is copied in the shadow memory is specified in the X-Loader's header.

38.3.3 System controller

The system controller is used to program/control the system clock mode and frequency. After reset, the system is set to DOZE mode.

BootROM configures System in different modes for different booting type:-

c) For Serial NOR, Parallel NOR, 8/16 bit NAND and BootROM bypass booting: SLOW mode (CPU 24 MHz - AHB 24 MHz)

d) USB booting: NORMAL mode (CPU 333 MHz - AHB 166 MHz)

38.4 Software overview

This section describes the BootROM software for SPEAr300.

38.4.1 ARM processor modes

SPEAr BootROM runs in supervisor mode during entire execution.

38.4.2 SoC peripheral interrupts

SPEAr BootROM runs with all interrupts disabled, except in the particular case of boot/upgrade through USB, in which case it enable the USB interrupt.

38.4.3 Memory Overview

BootROM in SPEAr is located at 0xFFFF_0000.

Code section starts from 0xFFFF_0000 and ends at 0xFFFF_7FFF. This section contains all the routines required to boot on SPEAr300.

38.4.4 X-Loader and U-boot Header

```
typedef struct image_header
{
uint32_t ih_magic; /* Image Header Magic Number */
uint32_t ih_hcrc; /* Image Header CRC Checksum */
uint32_t ih_time; /* Image Creation Timestamp */
uint32_t ih_size; /* Image Data Size */
uint32_t ih_load; /* Data Load Address */
uint32_t ih_ep; /* Entry Point Address */
uint32_t ih_dcrc; /* Image Data CRC Checksum */
uint8_t ih_os; /* Operating System */
uint8_t ih_arch; /* CPU architecture */
uint8_t ih_type; /* Image Type */
uint8_t ih_comp; /* Compression Type */
uint8_t ih_name[IH_NMLEN]; /* Image Name */
} image_header_t;
```

38.4.5 X-Loader and U-boot authentication

During next section of Boot Flows, X-Loader and U-boot authentication is done by the following three steps:-

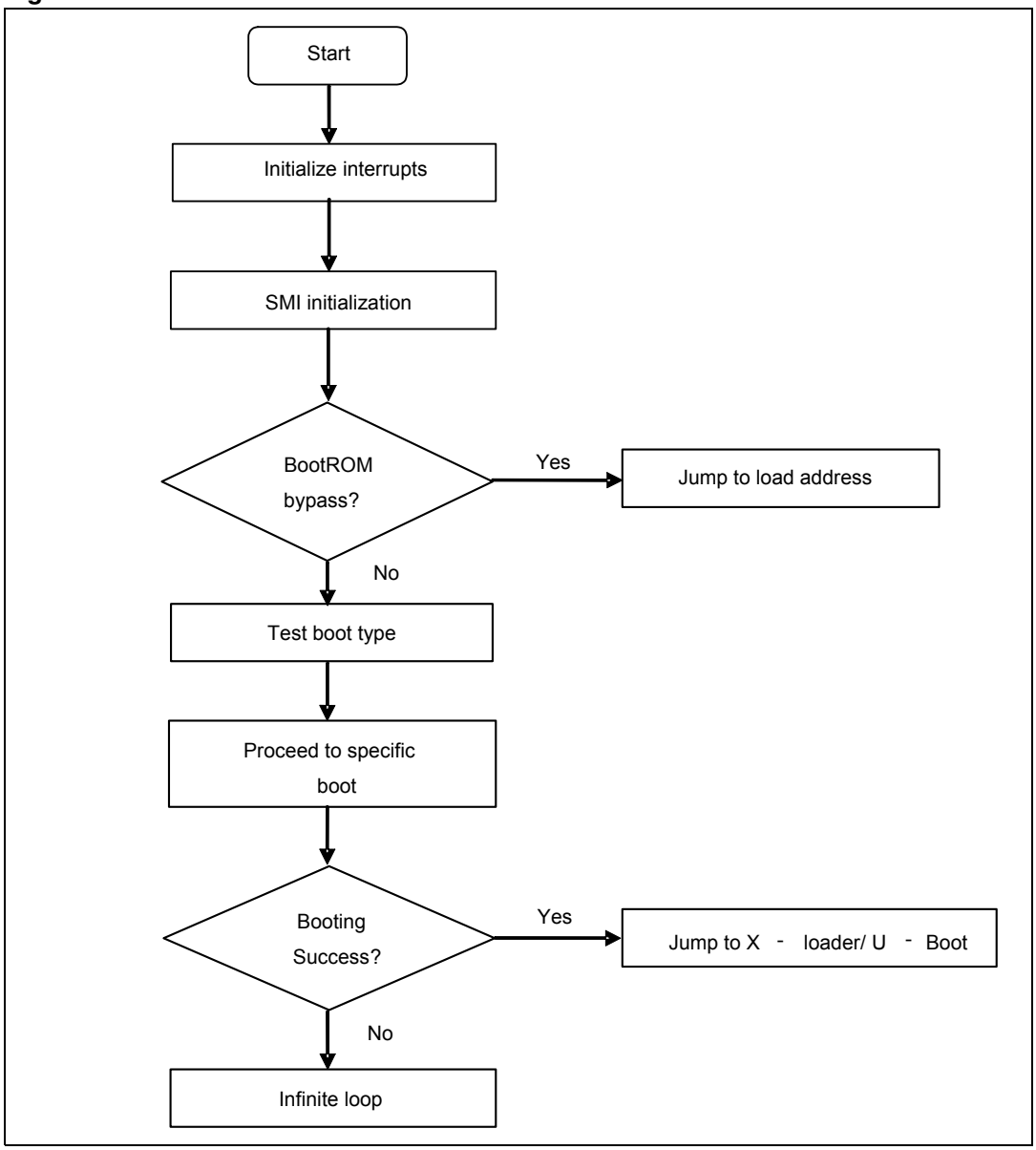
1. Compare the Image Name in received header with:
   – XLOADER in case of X-Loader
   – UBOOT in case of U-boot
2. Compare the magic number in received header with 0x2705_1956.
3. Checking the CRC of the image header and image received with the CRCs present in the header.

38.5 Boot flows

An overview diagrammatical representation of boot flow is given below. A detailed explanation of each boot flow is explained in further section.

- Serial NOR Flash
- Parallel NAND Flash
- USB upgrade
- Parallel NOR Flash
- UART boot
- Ethernet boot

Figure 114. Boot flows

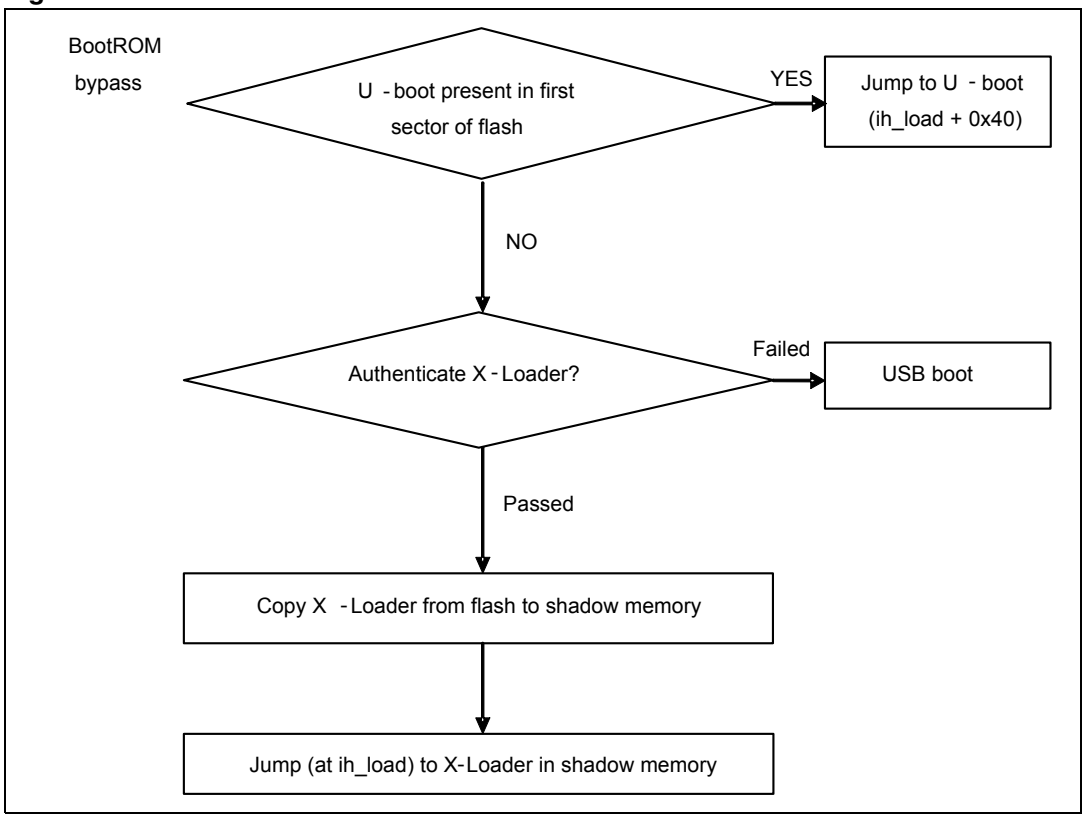


38.5.1 Serial NOR Flash boot

In NOR Boot, BootROM tries to perform BootROM bypass i.e. BootROM tries to authenticate U-boot in the first sector of Flash, if U-boot is found then BootROM jumps to the U-boot (ih_load location plus 0x40), otherwise if U-boot authentication fails then BootROM tries to authenticate X-Loader, if X-loader is found, then BootROM copies X-Loader from Flash to 'shadow memory' area and jumps to X-Loader in 'shadow memory'. The load address of X-Loader is also specified in 64 bytes X-Loader header (ih_load).

If X-Loader authentication fails, or any other error is occurs during serial NOR boot, BootROM shifts to USB boot.

Figure 115. Serial NOR Flash boot



38.5.2 NAND Flash boot

To detect NAND devices, initialize the FSMC controller with the relaxed timing values.

- Thiz = 0x01;
- Thold = 0x04;
- Twait = 0x06;
- Tset = 0x00;

Boot ROM code reads the device ID code and looks for it in a table which contains all supported devices code. Then, it fills a device description structure for page_size, block_size, memory size & spare command according to this ID code.

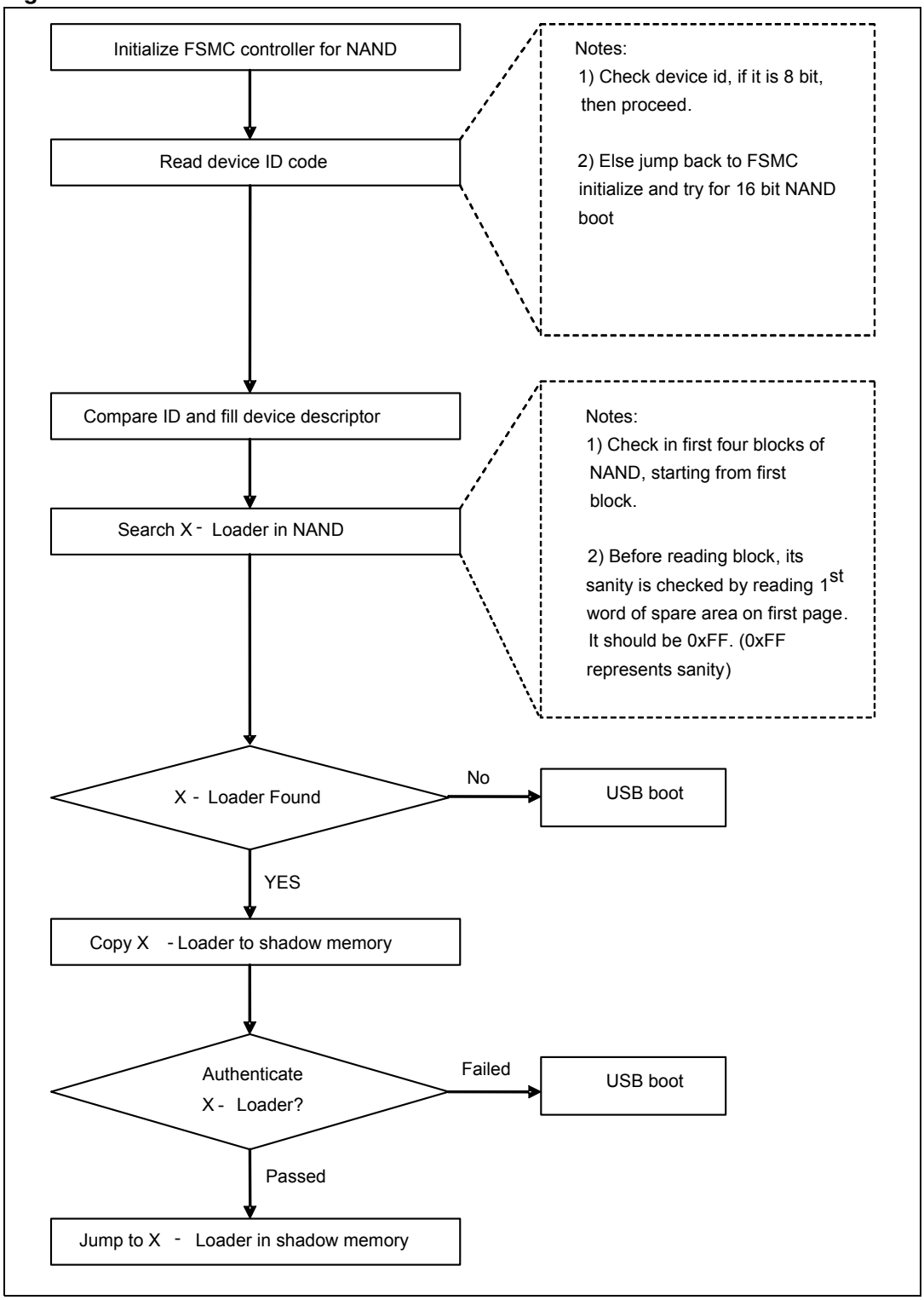
Then it searches for X-Loader in the first page of every block of the Flash starting from 1st block.

It checks the sanity of the block before reading the block by reading the 1st word of spare area of every block. It should be 0xFF. If bad then skip to next block.

Read the whole page in a buffer (buffer size equals page size) & search for X-Loader. If found, return start address. Find the transfer size from the header. Calculate the number of pages & start reading the pages from the Flash & copy them into the shadow memory. Next step is to authenticate X-Loader. If authenticated then jump to the start address, else jump to USB boot.

While reading check ECC for every page, Read ECC from the NAND memory spare area and from the FSMC controller, if error of 1 bit, fix it else return Boot failed.

Figure 116. NAND Flash boot



38.5.3 USB boot

USB Boot refers upgrading of Flash memories (NAND and NOR) via USB. In USB boot, BootROM programs PLL1 to 333 MHz and system to Normal mode i.e. ARM frequency at 333 MHz, initializes UDC Controller and initializes USB state machine to GET_CMD phase.

UDC supports 2 modes of operation i.e. Slave mode and DMA mode, in SPEAr300 BootROM UDC is configured in slave mode.

After initializing, BootROM waits for a 12 byte command on BULK Out End point 2 from the USB host, the format for 12 byte command is as follows.

Table 744. Command format

Byte 0	Type of Data
Byte 1 - 3	RESERVED
Byte 4 - 7	Size of Data
Byte 8 - 11	Load Address in RAM

After receiving 12 bytes BootROM decodes 12 byte command, changes the USB state machine to GET_DATA phase and then waits for expected number of bytes from Host. BootROM receives the data and stores it into load address specified in the command, once all the data is received, BootROM changes the USB state machine to EXEC phase and decodes the type of data, if the received data is DDR Driver, then BootROM jumps to loadaddress, executes the DDR driver and jumps back to BootROM. Now that the DDR is initialized, BootROM changes the USB state machine again to GET_CMD phase. Now same process is repeated again, but this time type of data received is FIRMWARE, the FIRMWARE is capable of receiving data from Host, Flash upgrade capable etc. After receiving the FIRMWARE, BootROM jumps to it in DDR.

The current version of BootROM uses the slave mode, because of limitation of SPEAr architecture i.e. there is no Path from UDC DMA to access descriptors present in eRAM.

USB descriptors

As a part of enumeration process, several descriptors are exchanged. Details of these descriptors are as follows: -

1. **Device Descriptors**: - Each gadget has one device descriptor

Table 745. Device descriptors

Offset	Field	Size	Value
0	bLength	Byte	0x12
1	bDescriptorType	Byte	0x01
2	bcdUSB	Word	0x200
4	bDeviceClass	Byte	0x00
5	bDeviceSubClass	Byte	0x00
6	bDeviceProtocol	Byte	0x00
7	wMaxPacketSize0	Byte	0x40
8	idVendor	Word	0x0483
10	iProduct	Word	0x3801
12	bcdDevice	Word	0x100
14	iManufacturer	Byte	0x01
15	iProduct	Byte	0x02
16	iSerial Number	Byte	0x03
17	bNumConfigurations	Byte	0x01

2. **Configuration descriptors**:- Each device has one default configuration descriptor which supports at least one interface.

Table 746. Configuration registers

Offset	Field	Size	Value
0	bLength	Byte	0x09
1	bDescriptorType	Byte	0x02
2	bTotalLength	Word	0x0020
4	bNumInterfaces	Byte	0x01
5	bConfigurationValue	Byte	0x01
6	iConfiguration	Byte	0x00
7	bmAttributes	Byte	0xE0
8	MaxPower	Byte	0x00

3. **Interface descriptors**:- Each device has a single data interface with no possible alternates.

Table 747. Interface registers

Offset	Field	Size	Value
0	bLength	Byte	0x09
1	bDescriptorType	Byte	0x04
2	bInterfaceNumber	Byte	0x00
3	bAlternateSetting	Byte	0x00
4	bNumEndpoints	Byte	0x02
5	bInterfaceClass	Byte	0x00
6	interfaceSubClass	Byte	0x00
7	bInterfaceProtocol	Byte	0x02
8	iInterface	Byte	0x01

4. **Endpoint descriptors**:- A device supports the following endpoints.
   a) **Bulk OUT endpoint**:- Used for transfer of data from host to device.

Table 748. Bulk OUT endpoint

Offset	Field	Size	Value
0	bLength	Byte	0x07
1	bDescriptorType	Byte	0x05
2	bEndpointAddress	Byte	0x02
3	bmAttributes	Byte	0x02
4	wMaxPacketSize	Word	0x040
6	bInterval	Byte	0x00

   b) **Bulk IN endpoint**:- Used for transfer of data from device to host.

Table 749. Bulk In endpoint

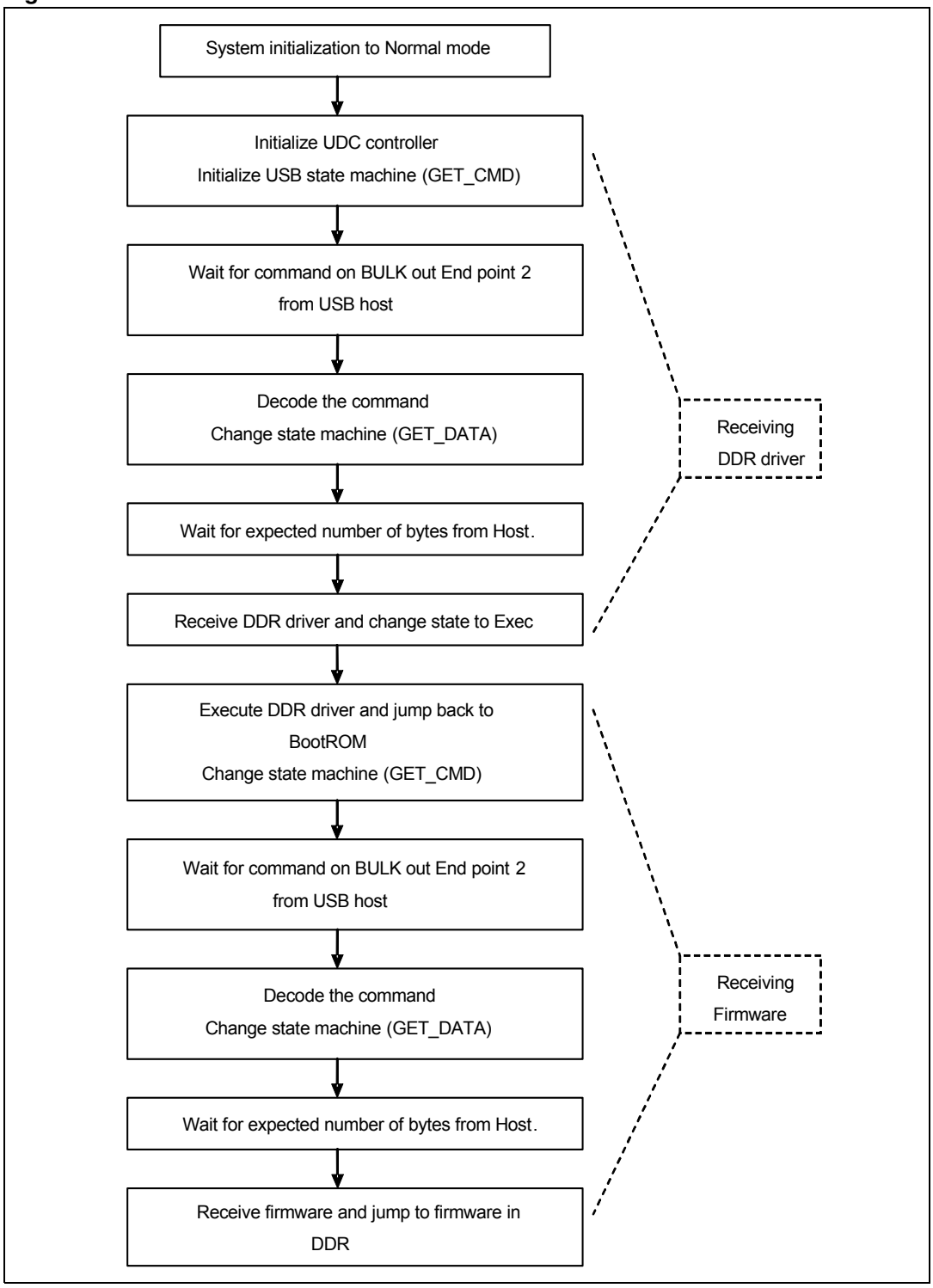
Offset	Field	Size	Value
0	bLength	Byte	0x07
1	bDescriptorType	Byte	0x05
2	bEndpointAddress	Byte	0x81
3	bmAttributes	Byte	0x02
4	wMaxPacketSize	Word	0x040
6	bInterval	Byte	0x00

5. String descriptors

Table 750. String descriptors

Offset	Field	Size	Value
0	bLength	Byte	0x04
1	bDescriptorType	Byte	0x03
2	bEndpointAddress	Byte	0x0409

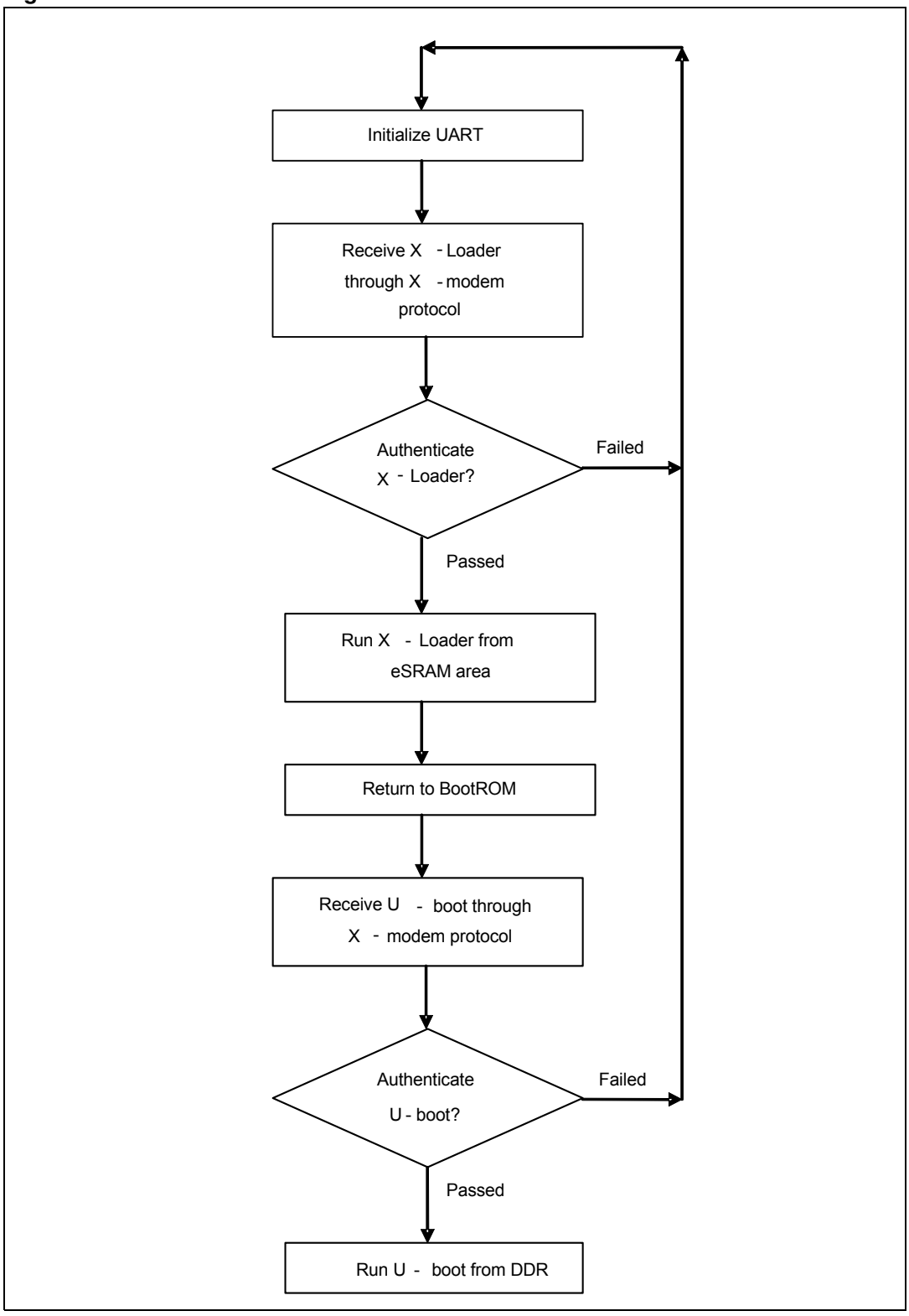
Figure 117. USB boot



38.5.4 Serial (UART) Boot

Serial boot initializes the UART IP and uses the X-modem protocol to receive the X-Loader image at a fixed baud rate of 11250 bits per second (bps). It then runs the X-Loader image which initializes the PLLs and DDR and then returns back the control to Boot code. After this, X-modem protocol is again used to download the U-boot image into DDR and then the image is run from there. To be noted that this booting mode does not require any non-volatile memory to be present on the board.

Figure 118. Serial boot



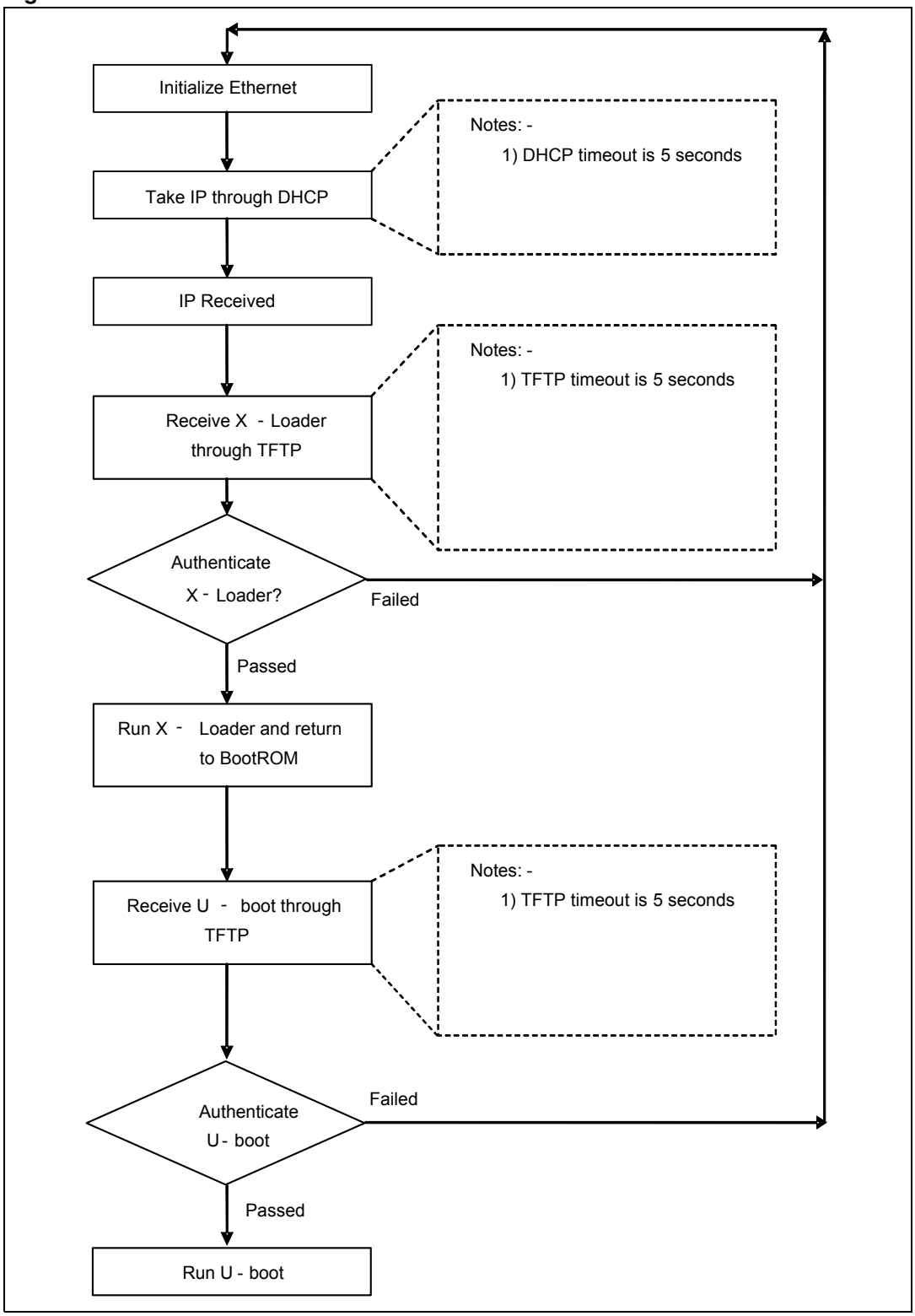
38.5.5 Ethernet boot

The Ethernet boot is specifically used to boot the devices which do not have a storage capacity. There are two phases in this kind of booting. For any device to communicate in an IP network, it needs to have an IP of its own. This can be a fixed IP which is stored in a non-volatile memory within the device or a new IP can be obtained through BOOTP server. BOOTP is a protocol which facilitates obtaining the IP from a server. This is essentially the first phase of booting in the case of Ethernet boot.

The second stage of booting starts after the device has its own IP and the device can communicate with the TFTP server to download images. This stage consists of downloading an executable image from the TFTP server via the TFTP protocol and executing that image. The first image which is downloaded is X-Loader and it is responsible for initializing the PLLs and DDR memory. The U-boot image is downloaded and run after X-Loader has completely run.

Please note that this booting also does not require any non-volatile memory to be present on the board.

Figure 119. Ethernet boot



39 Document revision history

Table 751. Document revision history

Date	Revision	Changes
29-Apr-2011	1	Initial Release

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com