SIGMATEL®

MIXED-SIGNAL MULTIMEDIA SEMICONDUCTORS

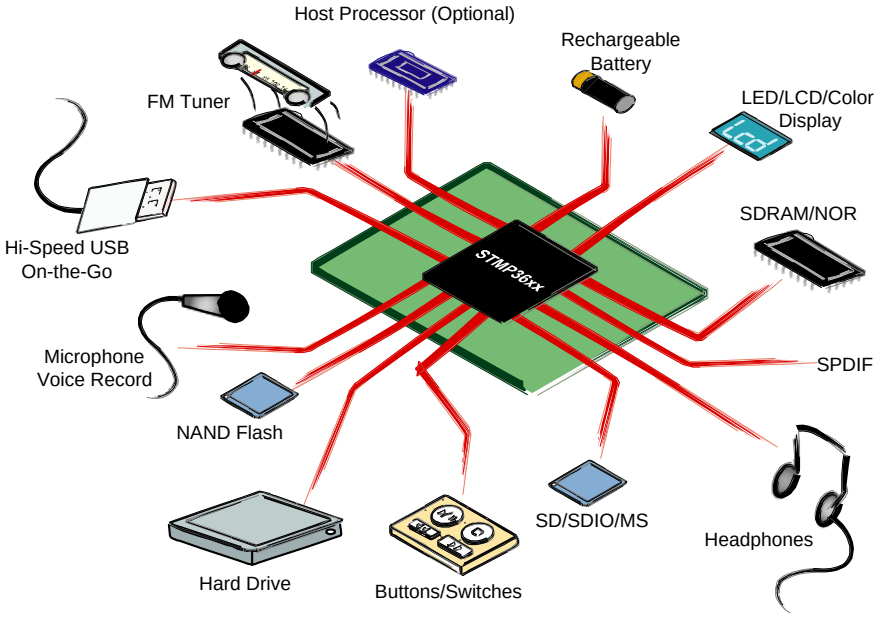
PRODUCT DATA SHEET

STMP36xx

Audio System on Chip
with USB OTG, LCD, Hard Drive, and Battery Charger

Fourth-Generation Audio Decoder

Version 1.02 May 3, 2006



ISO9001:2000 Certified
IEC QC 080000:2005
(IECQ HSPM) Certified

Windows Media™ Compatible

HI-SPEED CERTIFIED USB

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

5-36xx-D1-1.02-050306

Copyright © 2003-2006 SigmaTel, Inc.

All rights reserved.

SigmaTel, Inc. makes no warranty for the use of its products, assumes no responsibility for any errors which may appear in this document, and makes no commitment to update the information contained herein. SigmaTel reserves the right to change or discontinue this product at any time, without notice. There are no express or implied licenses granted hereunder to design or fabricate any integrated circuits based on information in this document.

The following are trademarks of SigmaTel, Inc., and may be used to identify SigmaTel products only: SigmaTel, the SigmaTel Logo, C Major, D Major and Go-Chip. Windows Media and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names contained herein may be trademarks of their respective owners.

CUSTOMER SUPPORT

Additional product and company information can be obtained by going to the SigmaTel website at:

http://www.sigmatel.com

Additional product and design information is available for authorized customers at:

http://extranet.sigmatel.com

The product shown in this data sheet is not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Any use or distribution of this product in such applications is at your own risk. SigmaTel, Inc. does not assume any liability arising out of the application or use of any product or circuit shown herein, and specifically disclaims any and all liability, including without limitation special, consequential, or incidental damages. Supply of this Implementation of AAC technology does not convey a license nor imply any right to use this Implementation in any finished end-user or ready-to-use final product. An independent license for such use is required.

CONTENTS

REVISION HISTORY **21**

1. PRODUCT OVERVIEW **23**
1.1. Hardware Features 23
1.2. Application Capability 24
1.3. Design Support 25
1.4. Additional Documentation 25
1.5. STMP36xx System Block Diagram 26
1.6. STMP36xx Product Features 27
1.6.1. ARM 926 Processor Core 27
1.6.2. On-Chip RAM and ROM 29
1.6.3. Interrupt Collector 30
1.6.4. Default First-Level Page Table 30
1.6.5. External Memory Interface (SDRAM/NOR Flash Controller) 30
1.6.6. DMA Controller 31
1.6.7. Clock Generation Subsystem 31
1.6.8. Power Management Unit 32
1.6.9. USB Interface 33
1.6.10. General-Purpose Media Interface (GPMI) 33
1.6.11. Hardware Acceleration for ECC for Robust External Storage 34
1.6.12. Memory Copy Unit 34
1.6.13. Mixed Signal Audio Subsystem 35
1.6.14. Master Digital Control Unit (DIGCTL) 35
1.6.15. Synchronous Serial Port (SSP) 35
1.6.16. I^2C Interface 35
1.6.17. General-Purpose Input/Output (GPIO) 36
1.6.18. LCD Controller 37
1.6.19. SPDIF Transmitter 37
1.6.20. Rotary Decoder 37
1.6.21. Dual UARTs 37
1.6.22. Infrared Interface 37
1.6.23. Low-Resolution ADC and Touch-Screen Interface 37
1.6.24. Pulse Width Modulator (PWM) Controller 37
1.6.25. Camera Interface 38

2. CHARACTERISTICS AND SPECIFICATIONS **39**
2.1. Absolute Maximum Ratings 39
2.2. Recommended Operating Conditions 40
2.2.1. Recommended Operating Conditions for Specific Clock Targets 41
2.3. DC Characteristics 42

3. ARM CPU COMPLEX **43**
3.1. ARM 926 Processor Core 43
3.2. JTAG Debugger 45
3.2.1. JTAG READ ID 45
3.2.2. JTAG Hardware Reset 45
3.2.3. JTAG Interaction with CPUCLK 45
3.3. Embedded Trace Macrocell (ETM) Interface 46

4. CLOCK GENERATION AND CONTROL **47**
4.1. Overview 47
4.2. Crystal Oscillators 47
4.3. Clock Domains 47
4.4. Power Saving Features of the Clock Architecture 49
4.5. Clock Dividers 49
4.5.1. Automatic HCLK Divider 49
4.6. Phase-Locked Loop (PLL) 51
4.6.1. Frequency Program 52
4.6.2. PLL Use in USB and SPDIF Modes 52
4.6.3. VCO and Phase Followers 53
4.6.4. PFD and Charge Pump 53

4.7. Integrated USB 2.0 PHY Initialization Flow Charts 54
4.8. Clocking During Reset 55
4.9. Programmable Registers 56
4.9.1. PLL Control Register 0 Description 56
4.9.2. PLL Control Register 1 Description 58
4.9.3. CPU Clock Control Register Description 58
4.9.4. AHB, APBH Bus Clock Control Register Description 59
4.9.5. APBX Clock Control Register Description 61
4.9.6. XTAL Clock Control Register Description 62
4.9.7. On-Chip SRAM Clock Control Register Description 63
4.9.8. UTMI Clock Control Register Description 63
4.9.9. Synchronous Serial Port Clock Control Register Description 64
4.9.10. General-Purpose Media Interface Clock Control Register Description 65
4.9.11. SPDIF Clock Control Register Description 66
4.9.12. EMI Clock Control Register Description 67
4.9.13. IR Clock Control Register Description 67

5. INTERRUPT COLLECTOR 69
5.1. Overview 69
5.2. Nesting of Multi-Level IRQ Interrupts 72
5.3. FIQ Generation 73
5.4. Interrupt Sources 75
5.5. CPU Wait-for-Interrupt Mode 77
5.6. Behavior During Reset 77
5.7. Programmable Registers 78
5.7.1. Interrupt Collector Interrupt Vector Address Register Description 78
5.7.2. Interrupt Collector Level Acknowledge Register Description 78
5.7.3. Interrupt Collector Control Register Description 79
5.7.4. Interrupt Collector Status Register Description 81
5.7.5. Interrupt Collector Raw Interrupt Input Register 0 Description 82
5.7.6. Interrupt Collector Raw Interrupt Input Register 1 Description 83
5.7.7. Interrupt Collector Priority Register 0 Description 83
5.7.8. Interrupt Collector Priority Register 1 Description 85
5.7.9. Interrupt Collector Priority Register 2 Description 86
5.7.10. Interrupt Collector Priority Register 3 Description 88
5.7.11. Interrupt Collector Priority Register 4 Description 90
5.7.12. Interrupt Collector Priority Register 5 Description 91
5.7.13. Interrupt Collector Priority Register 6 Description 93
5.7.14. Interrupt Collector Priority Register 7 Description 95
5.7.15. Interrupt Collector Priority Register 8 Description 96
5.7.16. Interrupt Collector Priority Register 9 Description 98
5.7.17. Interrupt Collector Priority Register 10 Description 100
5.7.18. Interrupt Collector Priority Register 11 Description 101
5.7.19. Interrupt Collector Priority Register 12 Description 103
5.7.20. Interrupt Collector Priority Register 13 Description 105
5.7.21. Interrupt Collector Priority Register 14 Description 106
5.7.22. Interrupt Collector Priority Register 15 Description 108
5.7.23. Interrupt Collector Interrupt Vector Base Address Register Description 110
5.7.24. Interrupt Collector Debug Register 0 Description 110
5.7.25. Interrupt Collector Debug Read Register 0 Description 112
5.7.26. Interrupt Collector Debug Read Register 1 Description 112
5.7.27. Interrupt Collector Debug Flag Register Description 113
5.7.28. Interrupt Collector Debug Read Request Register 0 Description 113
5.7.29. Interrupt Collector Debug Read Request Register 1 Description 114

6. DEFAULT FIRST-LEVEL PAGE TABLE FOR ARM926 MMU 115
6.1. Overview 115
6.2. 16-Megabyte Page-Mapped Virtual Memory (0xFFXXXXXX) 117
6.2.1. Default First-Level Page Table Entry 4095 118
6.2.2. Default First-Level Page Table Entries 4094–4080 119
6.2.3. Default First-Level Page Table PIO Register Map Entry 2048 120
6.2.4. Default First-Level Page Table Entry 0000 V==R SRAM Access 121

7. DIGITAL CONTROL AND ON-CHIP RAM 123
7.1. Overview 123
7.2. SRAM Controls 124
7.2.1. SRAM BIST Control 126
7.3. ROM Controls 127
7.4. Miscellaneous Controls 127
7.4.1. Performance Monitoring 127
7.4.2. High-Entropy PRN Seed 127
7.4.3. Write-Once Register 128
7.4.4. Microseconds Counter 128
7.5. Behavior During Reset 128
7.6. Programmable Registers 128
7.6.1. DIGCTL Control Register Description 128
7.6.2. DIGCTL Status Register Description 130
7.6.3. Free-Running HCLK Counter Register Description 131
7.6.4. On-Chip RAM Control Register Description 131
7.6.5. On-Chip RAM Repair Data 0 Register Description 133
7.6.6. On-Chip RAM Repair Data 1 Register Description 134
7.6.7. Software Write-Once Register Description 135
7.6.8. AHB Transfer Count Register Description 136
7.6.9. AHB Performance Metric for Stalled Bus Cycles Register Description 136
7.6.10. Entropy Register Description 137
7.6.11. Digital Control ROM Shield Read Enable Register Description 138
7.6.12. Digital Control Microseconds Counter Register Description 138
7.6.13. Digital Control Debug Read Test Register Description 139
7.6.14. Digital Control Debug Register Description 139
7.6.15. SRAM BIST Control and Status Register Description 140
7.6.16. SRAM BIST Repair Register 0 Description 140
7.6.17. SRAM BIST Repair Register 1 Description 141
7.6.18. SRAM Status Register 0 Description 142
7.6.19. SRAM Status Register 1 Description 142
7.6.20. SRAM Status Register 2 Description 143
7.6.21. SRAM Status Register 3 Description 143
7.6.22. SRAM Status Register 4 Description 144
7.6.23. SRAM Status Register 5 Description 144
7.6.24. SRAM Status Register 6 Description 145
7.6.25. SRAM Status Register 7 Description 145
7.6.26. SRAM Status Register 8 Description 146
7.6.27. SRAM Status Register 9 Description 146
7.6.28. SRAM Status Register 10 Description 147
7.6.29. SRAM Status Register 11 Description 147
7.6.30. SRAM Status Register 12 Description 148
7.6.31. SRAM Status Register 13 Description 149
7.6.32. Digital Control Scratch Register 0 Description 150
7.6.33. Digital Control Scratch Register 1 Description 150
7.6.34. Digital Control ARM Cache Register Description 151
7.6.35. SigmaTel Copyright Identifier Register Description 151
7.6.36. Digital Control Chip Revision Register Description 152

8. USB HIGH-SPEED ON-THE-GO (HOST/DEVICE) CONTROLLER 155
8.1. Overview 155
8.2. USB Controller Core 155
8.3. USB Programmed I/O (PIO) Target Interface 157
8.4. USB DMA Interface 157
8.5. USB UTMI Interface 157
8.5.1. Exporting the PHY 157
8.5.2. Digital/Analog Loopback Test Mode 157
8.6. USB Controller Flowcharts 158

9. INTEGRATED USB 2.0 PHY 161
9.1. Overview 161
9.2. External Signals 161
9.3. UTMI and Digital Circuits 162

9.3.1. UTMI Block 162
9.3.2. Digital Transmitter Block 162
9.3.3. Digital Receiver Block 162
9.3.4. Programmable Registers Block 162
9.4. Analog Transceiver 163
9.4.1. Analog Receiver 163
9.4.2. Analog Transmitter 164
9.5. Behavior During Reset 169
9.6. Programmable Registers 169
9.6.1. USB PHY Power-Down Register Description 169
9.6.2. USB PHY Transmitter Control Register Description 170
9.6.3. USB PHY Receiver Control Register Description 172
9.6.4. USB PHY General Control Register Description 173
9.6.5. USB PHY Status Register Description 175
9.6.6. USB PHY Debug Register Description 176
9.6.7. UTMI Debug Status Register 0 Description 177
9.6.8. UTMI Debug Status Register 1 Description 178
9.6.9. UTMI Debug Status Register 2 Description 179
9.6.10. UTMI Debug Status Register 3 Description 180
9.6.11. UTMI Debug Status Register 4 Description 181
9.6.12. UTMI Debug Status Register 5 Description 181
9.6.13. UTMI Debug Status Register 6 Description 182
9.6.14. UTMI Debug Status Register 7 Description 183
9.6.15. UTMI Debug Status Register 8 Description 184

10. AHB-TO-APBH BRIDGE WITH DMA 185
10.1. Overview 185
10.2. AHBH DMA 186
10.3. Implementation Examples 190
10.3.1. HWECC Example Command Chain 190
10.3.2. NAND Read Status Polling Example 191
10.3.3. APBH DMA and PIO Bus Implementation Example 193
10.4. Behavior During Reset 194
10.5. Programmable Registers 195
10.5.1. AHB-to-APBH Bridge Control and Status Register 0 Description 195
10.5.2. AHB-to-APBH Bridge Control and Status Register 1 Description 196
10.5.3. AHB-to-APBH DMA Device Assignment Register Description 198
10.5.4. APBH DMA Channel 0 Current Command Address Register Description 199
10.5.5. APBH DMA Channel 0 Next Command Address Register Description 199
10.5.6. APBH DMA Channel 0 Command Register Description 200
10.5.7. APBH DMA Channel 0 Buffer Address Register Description 202
10.5.8. APBH DMA Channel 0 Semaphore Register Description 202
10.5.9. AHB-to-APBH DMA Channel 0 Debug Register 1 Description 203
10.5.10. AHB-to-APBH DMA Channel 0 Debug Register 2 Description 205
10.5.11. APBH DMA Channel 1 Current Command Address Register Description 206
10.5.12. APBH DMA Channel 1 Next Command Address Register Description 207
10.5.13. APBH DMA Channel 1 Command Register Description 207
10.5.14. APBH DMA Channel 1 Buffer Address Register Description 209
10.5.15. APBH DMA Channel 1 Semaphore Register Description 210
10.5.16. AHB-to-APBH DMA Channel 1 Debug Register 1 Description 211
10.5.17. AHB-to-APBH DMA Channel 1 Debug Register 2 Description 212
10.5.18. APBH DMA Channel 2 Current Command Address Register Description 213
10.5.19. APBH DMA Channel 2 Next Command Address Register Description 214
10.5.20. APBH DMA Channel 2 Command Register Description 214
10.5.21. APBH DMA Channel 2 Buffer Address Register Description 216
10.5.22. APBH DMA Channel 2 Semaphore Register Description 217
10.5.23. AHB-to-APBH DMA Channel 2 Debug Register 1 Description 218
10.5.24. AHB-to-APBH DMA Channel 2 Debug Register 2 Description 219
10.5.25. APBH DMA Channel 3 Current Command Address Register Description 220
10.5.26. APBH DMA Channel 3 Next Command Address Register Description 221
10.5.27. APBH DMA Channel 3 Command Register Description 221
10.5.28. APBH DMA Channel 3 Buffer Address Register Description 223
10.5.29. APBH DMA Channel 3 Semaphore Register Description 224

10.5.30. AHB-to-APBH DMA Channel 3 Debug Register 1 Description 225
10.5.31. AHB-to-APBH DMA Channel 3 Debug Register 2 Description 226
10.5.32. APBH DMA Channel 4 Current Command Address Register Description 227
10.5.33. APBH DMA Channel 4 Next Command Address Register Description 228
10.5.34. APBH DMA Channel 4 Command Register Description 228
10.5.35. APBH DMA Channel 4 Buffer Address Register Description 230
10.5.36. APBH DMA Channel 4 Semaphore Register Description 231
10.5.37. AHB-to-APBH DMA Channel 4 Debug Register 1 Description 232
10.5.38. AHB-to-APBH DMA Channel 4 Debug Register 2 Description 233
10.5.39. APBH DMA Channel 5 Current Command Address Register Description 234
10.5.40. APBH DMA Channel 5 Next Command Address Register Description 235
10.5.41. APBH DMA Channel 5 Command Register Description 235
10.5.42. APBH DMA Channel 5 Buffer Address Register Description 237
10.5.43. APBH DMA Channel 5 Semaphore Register Description 238
10.5.44. AHB-to-APBH DMA Channel 5 Debug Register 1 Description 239
10.5.45. AHB-to-APBH DMA Channel 5 Debug Register 2 Description 240
10.5.46. APBH DMA Channel 6 Current Command Address Register Description 241
10.5.47. APBH DMA Channel 6 Next Command Address Register Description 242
10.5.48. APBH DMA Channel 6 Command Register Description 242
10.5.49. APBH DMA Channel 6 Buffer Address Register Description 244
10.5.50. APBH DMA Channel 6 Semaphore Register Description 245
10.5.51. AHB-to-APBH DMA Channel 6 Debug Register 1 Description 246
10.5.52. AHB-to-APBH DMA Channel 6 Debug Register 2 Description 247
10.5.53. APBH DMA Channel 7 Current Command Address Register Description 248
10.5.54. APBH DMA Channel 7 Next Command Address Register Description 249
10.5.55. APBH DMA Channel 7 Command Register Description 249
10.5.56. APBH DMA Channel 7 Buffer Address Register Description 251
10.5.57. APBH DMA Channel 7 Semaphore Register Description 252
10.5.58. AHB-to-APBH DMA Channel 7 Debug Register 1 Description 253
10.5.59. AHB-to-APBH DMA Channel 7 Debug Register 2 Description 254

11. AHB-TO-APBX BRIDGE WITH DMA 257
11.1. Overview 257
11.2. APBX DMA 258
11.3. DMA Chain Example 261
11.4. Behavior During Reset 262
11.5. Programmable Registers 263
11.5.1. AHB-to-APBX Bridge Control and Status Register 0 Description 263
11.5.2. AHB-to-APBX Bridge Control and Status Register 1 Description 264
11.5.3. AHB-to-APBX DMA Device Assignment Register Description 266
11.5.4. APBX DMA Channel 0 Current Command Address Register Description 267
11.5.5. APBX DMA Channel 0 Next Command Address Register Description 267
11.5.6. APBX DMA Channel 0 Command Register Description 268
11.5.7. APBX DMA Channel 0 Buffer Address Register Description 270
11.5.8. APBX DMA Channel 0 Semaphore Register Description 270
11.5.9. AHB-to-APBX DMA Channel 0 Debug Register 1 Description 271
11.5.10. AHB-to-APBX DMA Channel 0 Debug Register 2 Description 273
11.5.11. APBX DMA Channel 1 Current Command Address Register Description 274
11.5.12. APBX DMA Channel 1 Next Command Address Register Description 275
11.5.13. APBX DMA Channel 1 Command Register Description 275
11.5.14. APBX DMA Channel 1 Buffer Address Register Description 277
11.5.15. APBX DMA Channel 1 Semaphore Register Description 277
11.5.16. AHB-to-APBX DMA Channel 1 Debug Register 1 Description 278
11.5.17. AHB-to-APBX DMA Channel 1 Debug Register 2 Description 280
11.5.18. APBX DMA Channel 2 Current Command Address Register Description 281
11.5.19. APBX DMA Channel 2 Next Command Address Register Description 282
11.5.20. APBX DMA Channel 2 Command Register Description 282
11.5.21. APBX DMA Channel 2 Buffer Address Register Description 284
11.5.22. APBX DMA Channel 2 Semaphore Register Description 284
11.5.23. AHB-to-APBX DMA Channel 2 Debug Register 1 Description 285
11.5.24. AHB-to-APBX DMA Channel 2 Debug Register 2 Description 287
11.5.25. APBX DMA Channel 3 Current Command Address Register Description 288
11.5.26. APBX DMA Channel 3 Next Command Address Register Description 289

11.5.27. APBX DMA Channel 3 Command Register Description 289
11.5.28. APBX DMA Channel 3 Buffer Address Register Description 291
11.5.29. APBX DMA Channel 3 Semaphore Register Description 291
11.5.30. AHB-to-APBX DMA Channel 3 Debug Register 1 Description 292
11.5.31. AHB-to-APBX DMA Channel 3 Debug Register 2 Description 294
11.5.32. APBX DMA Channel 4 Current Command Address Register Description 295
11.5.33. APBX DMA Channel 4 Next Command Address Register Description 296
11.5.34. APBX DMA Channel 4 Command Register Description 296
11.5.35. APBX DMA Channel 4 Buffer Address Register Description 298
11.5.36. APBX DMA Channel 4 Semaphore Register Description 298
11.5.37. AHB-to-APBX DMA Channel 4 Debug Register 1 Description 299
11.5.38. AHB-to-APBX DMA Channel 4 Debug Register 2 Description 301
11.5.39. APBX DMA Channel 5 Current Command Address Register Description 302
11.5.40. APBX DMA Channel 5 Next Command Address Register Description 303
11.5.41. APBX DMA Channel 5 Command Register Description 303
11.5.42. APBX DMA Channel 5 Buffer Address Register Description 305
11.5.43. APBX DMA Channel 5 Semaphore Register Description 305
11.5.44. AHB-to-APBX DMA Channel 5 Debug Register 1 Description 306
11.5.45. AHB-to-APBX DMA Channel 5 Debug Register 2 Description 308
11.5.46. APBX DMA Channel 6 Current Command Address Register Description 309
11.5.47. APBX DMA Channel 6 Next Command Address Register Description 310
11.5.48. APBX DMA Channel 6 Command Register Description 310
11.5.49. APBX DMA Channel 6 Buffer Address Register Description 312
11.5.50. APBX DMA Channel 6 Semaphore Register Description 312
11.5.51. AHB-to-APBX DMA Channel 6 Debug Register 1 Description 313
11.5.52. AHB-to-APBX DMA Channel 6 Debug Register 2 Description 315
11.5.53. APBX DMA Channel 7 Current Command Address Register Description 316
11.5.54. APBX DMA Channel 7 Next Command Address Register Description 317
11.5.55. APBX DMA Channel 7 Command Register Description 317
11.5.56. APBX DMA Channel 7 Buffer Address Register Description 319
11.5.57. APBX DMA Channel 7 Semaphore Register Description 319
11.5.58. AHB-to-APBX DMA Channel 7 Debug Register 1 Description 320
11.5.59. AHB-to-APBX DMA Channel 7 Debug Register 2 Description 322

12. EXTERNAL MEMORY INTERFACE (EMI) 325
12.1. Overview 325
12.2. Dynamic Memory Controller 326
12.2.1. DRAM Timing 327
12.3. Static Memory Controller (SMC) 327
12.4. EMI Operation Example 328
12.5. Behavior During Reset 329
12.6. Programmable Registers 329
12.6.1. EMI Control Register Description 329
12.6.2. EMI Status Register Description 330
12.6.3. EMI Debug Register Description 331
12.6.4. EMI DRAM Status Register Description 332
12.6.5. EMI DRAM Control Register Description 333
12.6.6. EMI DRAM Address Configuration Register Description 334
12.6.7. EMI DRAM Mode Configuration Register Description 335
12.6.8. EMI DRAM Timing Control Register 1 Description 336
12.6.9. EMI DRAM Timing Control Register 2 Description 338
12.6.10. EMI Static Memory Control Register Description 338
12.6.11. EMI Static Memory Timing Control Register Description 339

13. GENERAL-PURPOSE MEDIA INTERFACE (GPMI) 341
13.1. Overview 341
13.2. GPMI ATA Mode 341
13.2.1. Basic ATA Operation 341
13.2.2. GPMI ATA Clocking and Timing 341
13.2.3. GPMI ATA Pin Sharing 342
13.2.4. ATA PIO Mode Timing 343
13.2.5. ATA UDMA Mode 343
13.2.6. UDMA Timings 344

13.2.7. ATA Command/IRQ/Check Status Example 344
13.3. GPMI NAND Mode 345
13.3.1. Multiple NAND Support 345
13.3.2. GPMI NAND Timing and Clocking 345
13.3.3. Basic NAND Timing 346
13.3.4. NAND Command and Address Timing Example 346
13.3.5. NAND Read Timing 346
13.4. Behavior During Reset 348
13.5. Programmable Registers 348
13.5.1. GPMI Control Register 0 Description 348
13.5.2. GPMI Compare Register Description 351
13.5.3. GPMI Control Register 1 Description 351
13.5.4. GPMI Timing Register 0 Description 353
13.5.5. GPMI Timing Register 1 Description 354
13.5.6. GPMI Timing Register 2 Description 355
13.5.7. GPMI DMA Data Transfer Register Description 356
13.5.8. GPMI Status Register Description 356
13.5.9. GPMI Debug Information Register Description 357

14. HARDWARE ECC ACCELERATOR (HWECC) 361
14.1. Overview 361
14.2. Reed-Solomon ECC Accelerator 362
14.2.1. Reed-Solomon Encoding 364
14.2.2. Reed-Solomon Decoding 366
14.2.3. Reed-Solomon Decoding Using PIO Debug Mode 371
14.3. Behavior During Reset 372
14.4. Programmable Registers 373
14.4.1. Hardware ECC Accelerator Control Register Description 373
14.4.2. Hardware ECC Accelerator Status Register Description 374
14.4.3. Hardware ECC Accelerator Debug Register 0 Description 375
14.4.4. Hardware ECC Accelerator Debug Register 1 Description 377
14.4.5. Hardware ECC Accelerator Debug Register 2 Description 378
14.4.6. Hardware ECC Accelerator Debug Register 3 Description 378
14.4.7. Hardware ECC Accelerator Debug Register 4 Description 379
14.4.8. Hardware ECC Accelerator Debug Register 5 Description 380
14.4.9. Hardware ECC Accelerator Debug Register 6 Description 380
14.4.10. Hardware ECC Accelerator DMA Read/Write Data Register Description 381

15. SYNCHRONOUS SERIAL PORT (SSP) 383
15.1. Overview 383
15.2. External Pins 384
15.3. Bit Rate Generation 384
15.4. Frame Format for SPI, SSI, and Microwire 384
15.5. Motorola SPI Mode 385
15.5.1. SPI DMA Mode 385
15.5.2. Motorola SPI Frame Format 385
15.5.3. Motorola SPI Format with Polarity=0, Phase=0 386
15.5.4. Motorola SPI Format with Polarity=0, Phase=1 387
15.5.5. Motorola SPI Format with Polarity=1, Phase=0 388
15.5.6. Motorola SPI Format with Polarity=1, Phase=1 389
15.6. Texas Instruments Synchronous Serial Interface (SSI) Mode 390
15.7. National Semiconductor Microwire Mode 391
15.8. SD/SDIO/MMC Mode 393
15.8.1. SD/MMC Command/Response Transfer 393
15.8.2. SD/MMC Data Block Transfer 394
15.8.3. SDIO Interrupts 397
15.8.4. SD/MMC Mode Error Handling 397
15.8.5. SD/MMC Clock Control 398
15.9. MS Mode 398
15.9.1. MS Mode I/O Pins 398
15.9.2. Basic MS Mode Protocol 398
15.9.3. MS Mode High-Level Operation 399
15.9.4. MS Mode Four-State Bus Protocol 399

15.9.5. Wait for Card IRQ 401
15.9.6. Checking Card Status 401
15.9.7. MS Mode Error Conditions 402
15.9.8. MS Mode Details 402
15.10. Behavior During Reset 402
15.11. Programmable Registers 403
15.11.1. SSP Control Register 0 Description 403
15.11.2. SD/MMC and MS Command Register 0 Description 405
15.11.3. SD/MMC Command Register 1 Description 407
15.11.4. SD/MMC and MS Compare Reference Register Description 407
15.11.5. SD/MMC and MS Compare Mask Register Description 408
15.11.6. SSP Timing Register Description 408
15.11.7. SSP Control Register 1 Description 409
15.11.8. SSP Data Register Description 412
15.11.9. SD/MMC Card Response Register 0 Description 413
15.11.10. SD/MMC Card Response Register 1 Description 413
15.11.11. SD/MMC Card Response Register 2 Description 414
15.11.12. SD/MMC Card Response Register 3 Description 414
15.11.13. SSP Status Register Description 414
15.11.14. SSP Debug Register Description 416

16. LCD INTERFACE (LCDIF) 419
16.1. Overview 419
16.2. LCD Interface Operation Example 420
16.2.1. Initialization Steps 421
16.2.2. Run Time Steps 421
16.3. LCDIF Pin Timing Diagrams 422
16.4. Behavior During Reset 422
16.5. Programmable Registers 423
16.5.1. LCD Interface Control and Status Register Description 423
16.5.2. LCD Interface Timing Register Description 425
16.5.3. LCD Interface Data Register Description 425
16.5.4. LCD Interface Debug Register Description 426

17. PIN CONTROL AND GPIO 429
17.1. Overview 429
17.2. Pin Interface Multiplexing 429
17.3. Pin Drive Strength Selection 434
17.4. GPIO Interface 434
17.4.1. Output Operation 434
17.4.2. Input Operation 435
17.4.3. Input Interrupt Operation 436
17.5. Behavior During Reset 438
17.6. Programmable Registers 439
17.6.1. PINCTRL Block Control Register Description 439
17.6.2. PINCTRL Bank 0 Lower Pin Mux Select Register Description 440
17.6.3. PINCTRL Bank 0 Upper Pin Mux Select Register Description 441
17.6.4. PINCTRL Bank 0 Drive Strength Register Description 441
17.6.5. PINCTRL Bank 0 Data Output Register Description 442
17.6.6. PINCTRL Bank 0 Data Input Register Description 443
17.6.7. PINCTRL Bank 0 Output Enable Register Description 443
17.6.8. PINCTRL Bank 0 Interrupt Select Register Description 444
17.6.9. PINCTRL Bank 0 Interrupt Mask Register Description 445
17.6.10. PINCTRL Bank 0 Interrupt Level/Edge Register Description 446
17.6.11. PINCTRL Bank 0 Interrupt Polarity Register Description 446
17.6.12. PINCTRL Bank 0 Interrupt Status Register Description 447
17.6.13. PINCTRL Bank 1 Lower Pin Mux Select Register Description 448
17.6.14. PINCTRL Bank 1 Upper Pin Mux Select Register Description 448
17.6.15. PINCTRL Bank 1 Drive Strength Register Description 449
17.6.16. PINCTRL Bank 1 Data Output Register Description 450
17.6.17. PINCTRL Bank 1 Data Input Register Description 451
17.6.18. PINCTRL Bank 1 Output Enable Register Description 452
17.6.19. PINCTRL Bank 1 Interrupt Select Register Description 452

17.6.20. PINCTRL Bank 1 Interrupt Mask Register Description 453
17.6.21. PINCTRL Bank 1 Interrupt Level/Edge Register Description 454
17.6.22. PINCTRL Bank 1 Interrupt Polarity Register Description 455
17.6.23. PINCTRL Bank 1 Interrupt Status Register Description 456
17.6.24. PINCTRL Bank 2 Lower Pin Mux Select Register Description 456
17.6.25. PINCTRL Bank 2 Upper Pin Mux Select Register Description 457
17.6.26. PINCTRL Bank 2 Drive Strength Register Description 458
17.6.27. PINCTRL Bank 2 Data Output Register Description 459
17.6.28. PINCTRL Bank 2 Data Input Register Description 459
17.6.29. PINCTRL Bank 2 Output Enable Register Description 460
17.6.30. PINCTRL Bank 2 Interrupt Select Register Description 461
17.6.31. PINCTRL Bank 2 Interrupt Mask Register Description 461
17.6.32. PINCTRL Bank 2 Interrupt Level/Edge Register Description 462
17.6.33. PINCTRL Bank 2 Interrupt Polarity Register Description 463
17.6.34. PINCTRL Bank 2 Interrupt Status Register Description 463
17.6.35. PINCTRL Bank 3 Lower Pin Mux Select Register Description 464
17.6.36. PINCTRL Bank 3 Upper Pin Mux Select Register Description 465
17.6.37. PINCTRL Bank 3 Drive Strength Register Description 466
17.6.38. PINCTRL Bank 3 Data Output Register Description 467
17.6.39. PINCTRL Bank 3 Data Input Register Description 468
17.6.40. PINCTRL Bank 3 Output Enable Register Description 469
17.6.41. PINCTRL Bank 3 Interrupt Select Register Description 469
17.6.42. PINCTRL Bank 3 Interrupt Mask Register Description 470
17.6.43. PINCTRL Bank 3 Interrupt Level/Edge Register Description 471
17.6.44. PINCTRL Bank 3 Interrupt Polarity Register Description 472
17.6.45. PINCTRL Bank 3 Interrupt Status Register Description 473

18. TIMERS AND ROTARY DECODER 475
18.1. Overview 475
18.2. Timers 476
18.2.1. Using External Signals as Inputs 477
18.2.2. Timer 3 and Duty Cycle Mode 478
18.2.3. Testing Timer 3 Duty Cycle Modes 479
18.3. Rotary Decoder 480
18.3.1. Testing the Rotary Decoder 482
18.3.2. Behavior During Reset 482
18.4. Programmable Registers 483
18.4.1. Rotary Decoder Control Register Description 483
18.4.2. Rotary Decoder Up/Down Counter Register Description 484
18.4.3. Timer 0 Control and Status Register Description 485
18.4.4. Timer 0 Count Register Description 487
18.4.5. Timer 1 Control and Status Register Description 487
18.4.6. Timer 1 Count Register Description 489
18.4.7. Timer 2 Control and Status Register Description 490
18.4.8. Timer 2 Count Register Description 491
18.4.9. Timer 3 Control and Status Register Description 492
18.4.10. Timer 3 Count Register Description 494

19. REAL-TIME CLOCK, ALARM, WATCHDOG, AND PERSISTENT BITS 497
19.1. Overview 497
19.2. Real-Time Clock 502
19.2.1. Behavior During Reset 502
19.3. Millisecond Resolution Timing Facility 502
19.4. Alarm Clock 502
19.5. Watchdog Reset Register 503
19.6. Laser Fuse Bits 503
19.7. Programmable Registers 503
19.7.1. Real-Time Clock Control Register Description 503
19.7.2. Real-Time Clock Status Register Description 505
19.7.3. Real-Time Clock Milliseconds Counter Description 506
19.7.4. Real-Time Clock Seconds Counter Register Description 507
19.7.5. Real-Time Clock Alarm Register Description 508
19.7.6. Watchdog Timer Register Description 508

19.7.7. Persistent State Register 0 Description 509
19.7.8. Persistent State Register 1 Description 511
19.7.9. Persistent State (On-Chip RAM Configuration) Register 2 Description 512
19.7.10. Persistent State (On-Chip RAM Configuration) Register 3 Description 513
19.7.11. Real-Time Clock Debug Register Description 513
19.7.12. RTC Unlock Register Description 514
19.7.13. HW Laser Fuse Register 0 Description 515
19.7.14. HW Laser Fuse Register 1 Description 516
19.7.15. HW Laser Fuse Register 2 Description 516
19.7.16. HW Laser Fuse Register 3 Description 517
19.7.17. HW Laser Fuse Register 4 Description 517
19.7.18. HW Laser Fuse Register 5 Description 518
19.7.19. HW Laser Fuse Register 6 Description 518
19.7.20. HW Laser Fuse Register 7 Description 519
19.7.21. HW Laser Fuse Register 8 Description 520
19.7.22. HW Laser Fuse Register 9 Description 520
19.7.23. HW Laser Fuse Register 10 Description 521
19.7.24. HW Laser Fuse Register 11 Description 521

20. PULSE-WIDTH MODULATOR (PWM) CONTROLLER 523
20.1. Overview 523
20.2. Operation 523
20.3. Multi-Chip Attachment Mode 526
20.4. Behavior During Reset 527
20.5. Programmable Registers 527
20.5.1. PWM Control and Status Register 0 Description 527
20.5.2. PWM Channel 0 Active Register Description 528
20.5.3. PWM Channel 0 Period Register Description 529
20.5.4. PWM Channel 1 Active Register Description 530
20.5.5. PWM Channel 1 Period Register Description 531
20.5.6. PWM Channel 2 Active Register Description 532
20.5.7. PWM Channel 2 Period Register Description 533
20.5.8. PWM Channel 3 Active Register Description 534
20.5.9. PWM Channel 3 Period Register Description 535
20.5.10. PWM Channel 4 Active Register Description 536
20.5.11. PWM Channel 4 Period Register Description 537

21. I^2C INTERFACE 539
21.1. Overview 539
21.2. I^2C Interface External Pins 539
21.3. I^2C Interrupt Sources 540
21.4. I^2C Bus Protocol 542
21.4.1. Simple Device Transactions 543
21.4.2. Typical EEPROM Transactions 544
21.4.3. Master Mode Protocol 545
21.4.4. Slave Mode Protocol 549
21.5. Programming Examples 552
21.5.1. Five Byte Master Write Using DMA 552
21.5.2. Reading 256 bytes from an EEPROM 553
21.6. Behavior During Reset 555
21.7. Programmable Registers 555
21.7.1. I2C Control Register 0 Description 555
21.7.2. I2C Timing Register 0 Description 558
21.7.3. I2C Timing Register 1 Description 558
21.7.4. I2C Timing Register 2 Description 559
21.7.5. I2C Control Register 1 Description 560
21.7.6. I2C Status Register Description 563
21.7.7. I2C Controller DMA Read and Write Data Register Description 567
21.7.8. I2C Device Debug Register 0 Description 567
21.7.9. I2C Device Debug Register 1 Description 569

22. APPLICATION UART 571
22.1. Overview 571
22.2. Operation 572
22.2.1. Fractional Baud Rate Divider 572
22.2.2. UART Character Frame 573
22.2.3. DMA Operation 573
22.2.4. Data Transmission or Reception 573
22.2.5. Error Bits 574
22.2.6. Overrun Bit 574
22.2.7. Disabling the FIFOs 574
22.3. Behavior During Reset 574
22.4. Programmable Registers 575
22.4.1. UART Receive DMA Control Register Description 575
22.4.2. UART Transmit DMA Control Register Description 576
22.4.3. UART Control Register Description 577
22.4.4. UART Line Control Register Description 580
22.4.5. UART Interrupt Register Description 581
22.4.6. UART Data Register Description 583
22.4.7. UART Status Register Description 584
22.4.8. UART Debug Register Description 586

23. DEBUG UART 589
23.1. Overview 589
23.2. Operation 590
23.2.1. Fractional Baud Rate Divider 590
23.2.2. UART Character Frame 591
23.2.3. Data Transmission or Reception 591
23.2.4. Error Bits 592
23.2.5. Overrun Bit 592
23.3. Disabling the FIFOs 592
23.4. Programmable Registers 592
23.4.1. UART Data Register Description 592
23.4.2. UART Receive Status Register (Read) and Error Clear Register (Write) Description .. 594
23.4.3. UART Flag Register Description 594
23.4.4. UART IrDA Low-Power Counter Register Description 595
23.4.5. UART Integer Baud Rate Divisor Register Description 596
23.4.6. UART Fractional Baud Rate Divisor Register Description 597
23.4.7. UART Line Control Register, High Byte Description 597
23.4.8. UART Control Register Description 599
23.4.9. UART Interrupt FIFO Level Select Register Description 600
23.4.10. UART Interrupt Mask Set/Clear Register Description 601
23.4.11. UART Raw Interrupt Status Register Description 602
23.4.12. UART Masked Interrupt Status Register Description 603
23.4.13. UART Interrupt Clear Register Description 604
23.4.14. UART DMA Control Register Description 606

24. IRDA CONTROLLER 607
24.1. Overview 607
24.2. Operation 608
24.2.1. DMA Operation 608
24.2.2. IR Transmit Processing 608
24.2.3. IR Receive Processing 609
24.2.4. IR Serial Interface 609
24.2.5. IR Clock Configuration 610
24.3. Behavior During Reset 610
24.4. Programmable Registers 611
24.4.1. IR Control Register Description 611
24.4.2. IR Transmit DMA Control Register Description 612
24.4.3. IR Receive DMA Register Description 613
24.4.4. IR Debug Control Register Description 614
24.4.5. IR Interrupt Register Description 615
24.4.6. IR RX Data Register Description 617
24.4.7. IR Status Register Description 618

24.4.8. IR Transceiver Control Register Description 619
24.4.9. IR Serial Interface Read Data Register Description 620
24.4.10. IR Debug Register Description 620

25. AUDIOIN/ADC 623
25.1. Overview 623
25.2. Operation 625
25.2.1. AUDIOIN DMA 626
25.3. ADC Sample Rate Converter and Internal Operation 627
25.4. Microphone 630
25.5. Behavior During Reset 631
25.6. Programmable Registers 631
25.6.1. AUDIOIN Control Register Description 631
25.6.2. AUDIOIN Status Register Description 634
25.6.3. AUDIOIN Sample Rate Register Description 634
25.6.4. AUDIOIN Volume Register Description 636
25.6.5. AUDIOIN Debug Register Description 638
25.6.6. ADC Mux Volume and Select Control Register Description 640
25.6.7. Microphone and Line Control Register Description 641
25.6.8. Analog Clock Control Register Description 643
25.6.9. AUDIOIN Read Data Register Description 644

26. AUDIOOUT/DAC 647
26.1. Overview 647
26.2. Operation 648
26.2.1. AUDIOOUT DMA 649
26.3. DAC Sample Rate Converter and Internal Operation 650
26.4. Reference Control Settings 653
26.5. Headphone 654
26.5.1. Board Components 656
26.5.2. Capless Mode Operation 656
26.6. Behavior During Reset 656
26.7. Programmable Registers 657
26.7.1. AUDIOOUT Control Register Description 657
26.7.2. AUDIOOUT Status Register Description 659
26.7.3. AUDIOOUT Sample Rate Register Description 660
26.7.4. AUDIOOUT Volume Register Description 662
26.7.5. AUDIOOUT Debug Register Description 664
26.7.6. Headphone Volume and Select Control Register Description 665
26.7.7. Speaker Volume Control Register Description 666
26.7.8. Audio Power-Down Control Register Description 667
26.7.9. AUDIOOUT Reference Control Register Description 668
26.7.10. Miscellaneous Audio Controls Register Description 671
26.7.11. Miscellaneous Test Audio Controls Register Description 673
26.7.12. BIST Control and Status Register Description 674
26.7.13. Hardware BIST Status 0 Register Description 675
26.7.14. Hardware AUDIOUT BIST Status 1 Register Description 676
26.7.15. Analog Clock Control Register Description 676
26.7.16. AUDIOOUT Write Data Register Description 677

27. SPDIF TRANSMITTER 679
27.1. Overview 679
27.2. Interrupts 682
27.3. Clocking 682
27.4. DMA Operation 683
27.5. PIO Debug Mode of Operation 684
27.6. Programmable Registers 685
27.6.1. SPDIF Control Register Description 685
27.6.2. SPDIF Status Register Description 687
27.6.3. SPDIF Frame Control Register Description 687
27.6.4. SPDIF Sample Rate Register Description 688
27.6.5. SPDIF Debug Register Description 689
27.6.6. SPDIF Write Data Register Description 690

28. DIGITAL RADIO INTERFACE (DRI) 693
28.1. Overview 693
28.2. Frame Structure 694
28.3. Behavior During Reset 696
28.4. Programmable Registers 697
28.4.1. DRI Control Register Description 697
28.4.2. DRI Timing Register Description 699
28.4.3. DRI Status Register Description 700
28.4.4. DRI Controller DMA Read Data Register Description 701
28.4.5. DRI Device Debug Register 0 Description 702
28.4.6. DRI Device Debug Register 1 Description 703

29. LOW-RESOLUTION ADC AND TOUCH-SCREEN INTERFACE 705
29.1. Overview 705
29.2. Scheduling Conversions 706
29.3. Delay Channels 708
29.4. Behavior During Reset 709
29.5. Programmable Registers 710
29.5.1. LRADC Control Register 0 Description 710
29.5.2. LRADC Control Register 1 Description 711
29.5.3. LRADC Control Register 2 Description 714
29.5.4. LRADC Control Register 3 Description 717
29.5.5. LRADC Status Register Description 719
29.5.6. LRADC 0 Result Register Description 721
29.5.7. LRADC 1 Result Register Description 722
29.5.8. LRADC 2 Result Register Description 723
29.5.9. LRADC 3 Result Register Description 724
29.5.10. LRADC 4 Result Register Description 726
29.5.11. LRADC 5 Result Register Description 727
29.5.12. LRADC 6 (VddIO) Result Register Description 728
29.5.13. LRADC 7 (BATT) Result Register Description 729
29.5.14. LRADC Scheduling Delay 0 Register Description 731
29.5.15. LRADC Scheduling Delay 1 Register Description 732
29.5.16. LRADC Scheduling Delay 2 Register Description 734
29.5.17. LRADC Scheduling Delay 3 Register Description 735
29.5.18. LRADC Debug Register 0 Description 737
29.5.19. LRADC Debug Register 1 Description 738
29.5.20. LRADC Battery Conversion Register Description 739

30. MEMORY COPY DEVICE 741
30.1. Overview 741
30.2. Programming Examples 742
30.2.1. Block Copy 742
30.2.2. ROM Dot Data Copying and BSS Fill 743
30.3. Behavior During Reset 744
30.4. Programmable Registers 744
30.4.1. Memory Copy Device Control and Status Register Description 744
30.4.2. MEMCPY Device DMA Read and Write Data Register Description 745
30.4.3. MEMCPY Device Debug Register Description 746

31. POWER SUPPLY 747
31.1. Overview 747
31.2. DC-DC Converters 748
31.2.1. DC-DC Operating Modes 748
31.2.2. DC-DC Operation 749
31.3. Linear Regulators 752
31.3.1. USB Compliance Features 752
31.3.2. 5V to Battery Power Interaction 753
31.3.3. Power-Up Sequence 754
31.3.4. Power-Down Sequence 754
31.3.5. Reset Sequence 755
31.3.6. Power Up, Power Down, and Reset Flow Chart 756
31.4. PSWITCH Pin Functions 757

31.4.1. Power On 757
31.4.2. Power Down 757
31.4.3. Software Functions/Recovery Mode 757
31.5. Battery Monitor 757
31.6. Battery Charger 758
31.7. Silicon Speed Sensor 759
31.8. DC-DC Programmable Registers 759
31.8.1. Power Control Register Description 759
31.8.2. DC-DC 5V Control Register Description 760
31.8.3. DC-DC Minimum Power and Miscellaneous Control Register Description 763
31.8.4. Battery Charge Control Register Description 765
31.8.5. VDDD and VDDIO Supply Targets and Brownouts Control Register Description 766
31.8.6. DC-DC#1 MultiOutput Converter Modes Control Register Description 768
31.8.7. DC-DC#1 Duty Cycle Limits Control Register Description 770
31.8.8. DC-DC#2 Duty Cycle Limits Control Register Description 770
31.8.9. Converter Loop Behavior Control Register Description 771
31.8.10. Power Subsystem Status Register Description 773
31.8.11. Temperature and Transistor Speed Control and Status Register Description 775
31.8.12. Battery Level Monitor Register Description 777
31.8.13. Power Module Reset Register Description 778
31.8.14. Power Module Debug Register Description 779
31.9. DC-DC Converter Efficiency 780
31.9.1. DC-DC1 Mode 3 Efficiency 780
31.9.2. DC-DC1 Mode 1 Efficiency 781
31.9.3. DC-DC1 Mode 2/Mode 0 Efficiency 783
31.9.4. DC-DC2 Mode 2 Efficiency 783
31.9.5. DC-DC2 Mode 0 Efficiency 784
31.9.6. Max Power Out in Boost Modes 785
31.9.7. Max Power Out in Buck Modes 786

32. BOOT MODES 787
32.1. Overview 787
32.2. Mode Selection 787
32.2.1. Boot Pins and Boot Modes 787
32.3. Boot Loader 790
32.3.1. Transition from Boot Loader to Runtime Image 790
32.3.2. Constructing Image to Be Loaded by Boot Loader 791
32.3.3. On-Chip RAM Used by Boot Loader 792
32.4. Preparing Bootable Images 792
32.4.1. I^2C EEPROM Boot Mode 792
32.4.2. Regular NOR Boot Mode 792
32.4.3. STMP (USB Recovery) Boot Mode 792
32.4.4. ATA Hard Disk Drive Boot Mode 792
32.4.5. Large-Block NAND Boot Mode 794

33. REGISTER MACRO USAGE 801
33.1. Definitions 801
33.2. Background 801
33.3. Naming Convention 802
33.4. Examples 803
33.4.1. Setting 1-Bit Wide Field 803
33.4.2. Clearing 1-Bit Wide Field 803
33.4.3. Toggling 1-Bit Wide Field 804
33.4.4. Modifying n-Bit Wide Field 804
33.4.5. Modifying Multiple Fields 804
33.4.6. Writing Entire Register (All Fields Updated at Once) 804
33.4.7. Reading a Bit Field 804
33.4.8. Reading Entire Register 805
33.4.9. Accessing Multiple Instance Register 805
33.4.10. Correct Way to Soft Reset a Block 805
33.5. Summary Preferred 805
33.6. Summary Alternate Syntax 806
33.7. Assembly Example 806

34. MEMORY MAP 807

35. PIN DESCRIPTIONS 809
35.1. Pin Placement and Definitions 809
35.1.1. Pin Definitions for 100-Pin TQFP Package 810
35.1.2. Pin Definitions for 169-Pin BGA Package 814
35.2. Functional Pin Groups 823
35.2.1. Analog Pins 823
35.2.2. DC-DC Converter Pins 824
35.2.3. General-Purpose Media Interface (GPMI) Pins 825
35.2.4. Synchronous Serial Port (SSP) Pins 827
35.2.5. Application and Debug UART Pins 827
35.2.6. External Memory Interface (SDRAM/NOR) Pins 828
35.2.7. I^2C Interface Pins 831
35.2.8. Digital Radio Interface (DRI) Pins 831
35.2.9. LCD Interface (LCDIF) Pins 831
35.2.10. Power Pins 833
35.2.11. System Pins 833
35.2.12. Timer and PWM Pins 834
35.2.13. USB Pins 835
35.2.14. General-Purpose Input/Output (GPIO) Pins 835

36. PACKAGE DRAWINGS 843
36.1. 100-Pin TQFP 843
36.2. 169-Pin fpBGA 844

37. STMP36XX PART NUMBERS AND ORDERING INFORMATION 845

APPENDIX:ACRONYMS AND ABBREVIATIONS 847

INDEX: REGISTER NAMES 851

LIST OF FIGURES

Figure 1. System Block Diagram 26
Figure 2. Chip Package Photographs 26
Figure 3. STMP36xx SOC Block Diagram 28
Figure 4. Memory Map for D-AHB Devices 29
Figure 5. Clock Diagram 32
Figure 6. Mixed Signal Audio Elements 36
Figure 7. ARM926 RISC Processor Core 44
Figure 8. ARM Programmable Registers 45
Figure 9. STMP36xx Clock Tree 50
Figure 10. Detail Diagram of Key Clocks 51
Figure 11. PLL Block Diagram 52
Figure 12. USB 2.0 PHY Startup Flowchart 54
Figure 13. USB 2.0 PHY PLL Suspend Flowchart 55
Figure 14. Interrupt Collector Diagram for IRQ Generation 69
Figure 15. Interrupt Collector Bit “37” Logic 70
Figure 16. IRQ Control Flow 71
Figure 17. Nesting of Multi-Level IRQ Interrupts 72
Figure 18. FIQ Generation Logic 74
Figure 19. Default First-Level Page Table (DFLPT) Block Diagram 115
Figure 20. Digital Control (DIGCTL) Block Diagram 123
Figure 21. On-Chip RAM Partitioning 124
Figure 22. On-Chip RAM E_Fuse Control 125
Figure 23. USB 2.0 Device Controller Block Diagram 156
Figure 24. USB 2.0 Check_USB_Plugged_In Flowchart 158
Figure 25. USB 2.0 USB PHY Startup Flowchart 159
Figure 26. USB 2.0 PHY PLL Suspend Flowchart 160
Figure 27. UTMI Powerdown 160
Figure 28. USB 2.0 PHY Block Diagram 161
Figure 29. USB 2.0 PHY Analog Transceiver Block Diagram 163
Figure 30. USB 2.0 PHY Transmitter Block Diagram 166
Figure 31. 45Ω Calibration Flowchart 168
Figure 32. AHB-to-APBH Bridge DMA Block Diagram 185
Figure 33. AHB-to-APBH Bridge DMA Channel Command Structure 187
Figure 34. AHB-to-APBH Bridge DMA HWECC Example Command Chain 191
Figure 35. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command 192
Figure 36. AHB-to-APB Bridge Device Interface 193
Figure 37. AHB-to-APBX Bridge DMA Block Diagram 257
Figure 38. AHB-to-APBX Bridge DMA Channel Command Structure 259
Figure 39. AHB-to-APBX Bridge DMA AUDIOOUT (DAC) Example Command Chain 262
Figure 40. External Memory Interface Block Diagram 326
Figure 41. SDRAM Programmable Timing Parameters 327
Figure 42. General-Purpose Media Interface Controller Block Diagram 342
Figure 43. ATA PIO Timing Mode 343
Figure 44. UDMA Timing 344
Figure 45. ATA Command/IRQ/Check Status Example 344
Figure 46. BASIC NAND Timing 346
Figure 47. NAND Command and Address Timing Example 346
Figure 48. GPMI NAND Read Path Timing 347
Figure 49. Hardware ECC Accelerator Block Diagram 361
Figure 50. Hardware ECC Reed-Solomon Block Coding—Encoder 363
Figure 51. Hardware ECC Reed-Solomon Encode Flowchart 364
Figure 52. Hardware ECC Reed-Solomon Encode DMA Chain 365
Figure 53. Hardware ECC Reed-Solomon Decode Flowchart 367
Figure 54. Hardware ECC Reed-Solomon Block Coding—Decoder Phase 1 368
Figure 55. Hardware ECC Reed-Solomon Block Decode DMA Chain 369

Figure 56. Synchronous Serial Port Block Diagram 383
Figure 57. Motorola SPI Frame Format (Single Transfer) with POLARITY=0 and PHASE=0 386
Figure 58. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=0 386
Figure 59. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=1 387
Figure 60. Motorola SPI Frame Format (Single Transfer) with POLARITY=1 and PHASE=0 388
Figure 61. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=1 and PHASE=0 388
Figure 62. Motorola SPI Frame Format with POLARITY=1 and PHASE=1 389
Figure 63. Texas Instruments Synchronous Serial Frame Format (Single Transfer) 390
Figure 64. Texas Instruments Synchronous Serial Frame Format (Continuous Transfer) 391
Figure 65. Microwire Frame Format (Single Transfer) 391
Figure 66. Microwire Frame Format (Continuous Transfer) 392
Figure 67. Microwire Frame Format (Continuous Transfer) 393
Figure 68. SD/MMC Block Transfer Flowchart 396
Figure 69. Basic MS Protocols 399
Figure 70. MS Operation Flowchart 400
Figure 71. MS Four-State Read and Write 401
Figure 72. LCD Interface Block Diagram 419
Figure 73. LCDIF Timing (Command Cycle) 422
Figure 74. Pin Control Mux Chart (Banks 0 and 1) 432
Figure 75. Pin Control Mux Chart (Banks 2 and 3) 433
Figure 76. GPIO Output Setup Flowchart 435
Figure 77. GPIO Input Setup Flowchart 436
Figure 78. GPIO Interrupt Flowchart 437
Figure 79. GPIO Interrupt Generation 438
Figure 80. Timers and Rotary Decoder Block Diagram 475
Figure 81. Timer 0, Timer 1, or Timer 2 Detail 476
Figure 82. Timer 3 Detail 478
Figure 83. Pulse-Width Measurement Mode 479
Figure 84. Detail of Rotary Decoder 480
Figure 85. Rotary Decoding Mode—Debouncing Rotary A and B Inputs 481
Figure 86. Rotary Decoding Mode—Input Transitions 482
Figure 87. RTC, Watchdog, Alarm, and Persistent Bits Block Diagram 498
Figure 88. RTC Initialization Sequence 499
Figure 89. Analog/Digital Interface Timing 500
Figure 90. RTC Writing to a Master Register from CPU 501
Figure 91. Pulse-Width Modulation Controller (PWM) Block Diagram 524
Figure 92. PWM Output Example 525
Figure 93. PWM Differential Output Pair Example 526
Figure 94. PWM Output Driver 527
Figure 94. PWM Output Driver 527
Figure 95. I^2C Interface Block Diagram 540
Figure 96. I^2C Data and Clock Timing 542
Figure 97. I^2C Data and Clock Timing Generation 543
Figure 98. I^2C Master Mode Flow Chart—Initial States 546
Figure 99. I^2C Master Mode Flow Chart—Receive States 547
Figure 100. I^2C Master Mode Flow Chart—Transmit States 548
Figure 101. I^2C Master Mode Flow Chart—Send Stop States 549
Figure 102. I^2C Slave Mode Flow Chart 551
Figure 103. I^2C Writing Five Bytes 552
Figure 104. I^2C Reading 256 Bytes from an EEPROM 553
Figure 105. Application UART Block Diagram 572
Figure 106. Application UART Character Frame 573
Figure 107. Debug UART Block Diagram 590
Figure 108. Debug UART Character Frame 591
Figure 109. IrDA Controller Block Diagram 607
Figure 110. Example of 1-Byte Serial Interface Write Command 610
Figure 111. Mixed Signal Audio Elements 624

Figure 112. AUDIOIN/ADC Block Diagram 625
Figure 113. Variable-Rate A/D Converter 629
Figure 114. External Microphone Bias Generation 630
Figure 115. Internal Microphone Bias Generation 631
Figure 116. AUDIOOUT/DAC Block Diagram 648
Figure 117. Stereo Sigma Delta D/A Converter 652
Figure 118. Conventional Stereo Headphone Application Circuit 654
Figure 119. Stereo Headphone Application Circuit with Common Node 654
Figure 120. Stereo Headphone Common Short Detection and Powerdown Circuit 655
Figure 121. Stereo Headphone L/R/ Short Detection and Powerdown Circuit 655
Figure 122. SPDIF Transmitter Block Diagram 680
Figure 123. SPDIF Flow Chart 681
Figure 124. SPDIF DMA Two-Block Transmit Example 683
Figure 125. Digital Radio Interface (DRI) Block Diagram 693
Figure 126. DRI Synchronization and Data Recovery 694
Figure 127. Digital Radio Interface (DRI) Framing 694
Figure 128. Digital Radio Interface (DRI) Digital Signals into Analog Line-In 696
Figure 129. Low-Resolution ADC and Touch-Screen Interface Block Diagram 705
Figure 130. Low-Resolution ADC Successive Approximation Unit 707
Figure 131. Using Delay Channels to Oversample a Touch-Screen 709
Figure 132. Memory Copy Device Block Diagram 741
Figure 133. Power Supply Block Diagram 748
Figure 134. Brownout Detection Flowchart 751
Figure 135. Power Up, Power Down, and Reset Flow Chart 756
Figure 136. Efficiency of DC-DC #1 in Mode 3 for Various VDDD Loads 780
Figure 137. Efficiency of DC-DC #1 in Mode 3 for Various VDDIO Loads 781
Figure 138. Efficiency of DC-DC #1 in Mode 1 for Various VDDD Loads 782
Figure 139. Efficiency of DC-DC #1 in Mode 1 for Various VDDIO Loads 782
Figure 140. Efficiency of DC-DC #1 in Mode 2/Mode 0 for Various VDDD Loads 783
Figure 141. Efficiency of DC-DC #2 in Mode 2 for Various Current Loads 784
Figure 142. Efficiency of DC-DC #2 in Mode 0 for Various Current Loads 785
Figure 143. 785
Figure 144. Creating a Boot Loader Image 791
Figure 145. ATA Media Layout 793
Figure 146. LB NAND with Generalized Page Layout Detail 794
Figure 147. ECC Coverage for NAND Pages 795
Figure 148. LB NAND with Specialized BCB Page Layout Detail 797
Figure 149. Data Organization in Multiple NANDs 800
Figure 150. Chip Package Photographs 809
Figure 151. 100-Pin TQFP Package Drawing 843
Figure 152. 169-Pin fpBGA Package Drawing 844

REVISION HISTORY

REVISION	DESCRIPTION
1.02	Entered changes to close the following ClearQuest STMP defect entries (STMP000nnnnn): 08170, 10337, 10513, 10545, 10550, 10614, 10630, 10902, 10903, 10907, 10917, 10920, 10921, 11001, 11005, 11011, 11088, 11089, 11131 • Updated descriptions in the four persistent registers, beginning on page 509, to clarify bit allocation and usage by the ROM, the SDK, SigmaTel, and the customer. (CQ08170) • Updated description of RST_ALL and RST_DIG bit fields in Table 998, "HW_POWER_RESET Bit Field Descriptions," on page 778. Updated description of PWDN_5VBRNOUT bit field in Table 976, "HW_POWER_5VCTRL Bit Field Descriptions," on page 761. (CQ10337) • Updated description of VDD_TRG and VDDIO_TRG bit fields in Table 982, "HW_POWER_VDDCTRL Bit Field Descriptions," on page 767. (CQ10513) • Updated dri_clk information in Section 28.2, "Frame Structure" on page 694. (CQ10545) • Removed 4.1-V option from BATT_CHARGE bit description in Table 980, "HW_POWER_BATTCHRG Bit Field Descriptions," on page 766 and Section 31.6, "Battery Charger" on page 758. (CQ10614) • Updated Section 9.4.2.6, "Resistor Calibration Mode" on page 167, and Figure 31, "45W Calibration Flowchart" on page 168. Updated TXENCAL45DP, TXENCAL45DN, and TXCALIBRATE bit fields in Table 185, "HW_USBPHY_TX Bit Field Descriptions," on page 171. (CQ10630) • Corrected value in Table 972, "DC-DC Battery Modes," on page 749 for DC-DC mode 2 resistor. (CQ10902) • Added Section 13.3.5, "NAND Read Timing" on page 346. (CQ10903) • Added UART and IrDA PIO word mapping information to Section 11.2, "APBX DMA" on page 258, Section 22.2.3, "DMA Operation" on page 573, and Section 24.2.1, "DMA Operation" on page 608. (CQ10907) • Updated Section 25.3, "ADC Sample Rate Converter and Internal Operation" on page 627 and the example in "AUDIOIN Sample Rate Register Description" on page 634. (CQ10917) • On the cover page, reinstated the USB Hi-Speed Certification logo, which had been removed in error. (CQ10920) • Added Table 5, "PLL Voltage Requirements," on page 41. Updated description of PLLV2ISEL bit field in Table 12, "HW_CLKCTRL_PLLCTRL0 Bit Field Descriptions," on page 56. (CQ10921) • Updated Section 8.4, "USB DMA Interface" on page 157 for on-chip RAM and SDRAM requirements. (CQ11001) • Added Table 7, "Recommended Operating Conditions for Specific EMICLK Targets," on page 42. (CQ11005) • Updated Table 4, "Recommended Operating Conditions for Specific CPUCLK Targets," on page 41 with new values. (CQ11011) • Updated Figure 1, "System Block Diagram" on page 26 to show correct LineIn and microphone amplifier connections. (CQ11088) • Updated Section 29.1 on page 705 with LRADC absolute accuracy value. (CQ11089) • Updated Section 29.1 on page 705 about using an external thermistor for temperature sensing. (CQ11131)

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

REVISION	DESCRIPTION
1.01	Updated “Recommended Operating Conditions for Specific Clock Targets” for VDDD requirements. Reorganized “Recommended Operating Conditions for Specific HCLK Targets” and “Recommended Operating Conditions for Specific CPUCLK Targets” for consistency. Added TBD values for maximum CPUCLK targets from 0 to 85 MHz. Added typical power dissipation value to “DC Characteristics”.
	Entered changes to close the following ClearQuest STMP defect entries (STMP000nnnnn): 10137, 10138, 10139, 10180 • Updated description of PLLV2ISEL bit field in “HW_CLKCTRL_PLLCTRL0 Bit Field Descriptions”. (CQ10137) • Updated description of TRAN_NOHYST in “HW_POWER_LOOPCTRL Bit Field Descriptions” and “DC-DC Extended Battery Life Features”. (CQ10138) • Updated “ARM 926 Processor Core” to add little endian information. (CQ10139) • Added figures and text to “AUDIOIN/ADC” to describe the microphone. Added figures and text to “AUDIOOUT/DAC” to describe the headphones. (CQ10180)
1.00	Initial public release.

1. PRODUCT OVERVIEW

The STMP36xx is SigmaTel's fourth-generation single-chip digital media SOC for applications such as digital audio players, PDAs, voice recorders, cell phones, portable video players, and digital photo wallets.

This chapter provides an general overview of the product and describes hardware features, application capability, design support, and additional documentation. A system block diagram (Figure 1), chip block diagram (Figure 3), clock overview diagram (Figure 5), and mixed signal audio diagram (Figure 6) are also provided in this chapter.

1.1. Hardware Features

- **ARM926 CPU Running at up to 200 MHz**
  - Integrated ARM926EJ-S CPU
  - 8KB + 8KB caches
  - ARM Embedded Trace Macrocell (ETM) version 9-medium
- **256KB of Integrated Low-Power On-Chip RAM**
- **Universal Serial Bus (USB) High-Speed On-The-Go (OTG)—Up to 480Mb/s**
  - High-speed USB device and host functions
  - Fully integrated high-speed OTG Physical Layer Protocol (PHY)
  - Complete OTG support
- **Power Management Unit**
  - Multi-channel DC-DC converter supports all common battery configurations
  - Features multi-channel boost, dual-output buck and buck/boost modes
  - PFM mode for low standby power
  - Improved, high-current battery charger for Lithium Ion (Li-Ion) and Nickel Metal Hydride (NiMH) batteries
  - Direct power from 5-V source (USB, wall power, or other source)
  - Can generate 5V from Li-Ion battery for USB OTG and other applications
  - Can generate 3.3V for hard drive
  - Silicon speed and temperature sensors enable adaptive power management over temperature and silicon process
- **Optimized for Very Long Battery Life**
  - 50 mW system power consumption while playing 128-kbps MP3 from SDRAM
- **Audio Codec**
  - Stereo DAC 99dB SNR
  - Stereo ADC with 90dB SNR
  - Stereo headphone amplifier with direct drive to eliminate bulky capacitors
  - Mono speaker amplifier with direct drive
  - Amplifiers are designed for click/pop free operation and have short-circuit protection
  - Two stereo line inputs
  - Microphone input
  - SPDIF digital out
- **8-Channel A/D converter**
  - 6 external channels, 2 internal channels
  - Resistive touch screen controller
  - Temperature sensor controller

- **Security Features**
  - Read-only unique ID for digital rights management algorithms
  - Secure boot
- **External Memory Interface (EMI)**
  - Provides memory-mapped (load/store) access to external memories
  - SDRAM
  - NOR flash
- **Wide Assortment of External Media Interfaces**
  - ATA hard drive
  - Up to four NAND flash with hardware management of device interleaving
  - High-speed MMC, secure digital, compact flash, MS
  - Hardware Reed-Solomon Error Correction Code (ECC) engine offers industry-leading protection and performance for NAND
- **Dual Peripheral Bus Bridges with 16 DMA Channels**
  - Multiple peripheral clock domains save power while optimizing performance
  - Direct Memory Access (DMA) with sophisticated linked DMA command architecture saves power and off loads the CPU
- **Liquid Crystal Display (LCD) Interface Works with All Standard LCD Modules**
  - 8- or 16-bit bus
- **Two Universal Asynchronous Receiver-Transmitters (UARTs)**
  - High-speed UART operates up to 1.5 Mb/s
- **I^2C Master/Slave**
  - DMA control of an entire EEPROM or other device read/write transaction without CPU intervention
- **Synchronous Serial Port (for SPI, Microwire, MMC, SDIO, MS)**
- **Four-Channel 16-Bit Timer with Rotary Decoder**
- **Five-Channel Pulse Width Modulator (PWM)**
- **Real-Time Clock**
  - Alarm clock can turn the system on
  - Uses the existing 24-MHz XTAL for low cost or 32.76-kHz for low power
- **SPDIF Transmit**
- **Flexible I/O Pins**
  - All digital pins have drive strength (4mA, 8mA) controls
  - Almost all digital pins have General-Purpose Input/Output (GPIO) mode
- **Offered in 100-Pin Thin Quad Flat Pack (TQFP) and 169-Pin Ball Grid Array (BGA) Packages**

1.2. Application Capability

- **Multi-Format Compressed Audio Encode and Decode**
- **Digital Rights Management (DRM)**
  - Microsoft PDDRM (Portable Device Digital Rights Management/DRM9)
  - WMDRM10 (Windows Media Digital Rights Management 10/Janus)

- **Voice Record in ADPCM or Nearly Any Other Format**
- **Graphical Equalizer**
- **Sound Effects and Spatialization**
- **JPEG Image Decode and Encode**
  - Simultaneous JPEG decoding and compressed audio playback
- **Video Decoder Capability**
  - Multi-format compressed video decode
- **Flexible USB Connectivity**
  - Mass storage device
  - Media transfer protocol device
  - Also supports proprietary USB device drivers
  - Mass storage host
  - USB OTG with mass storage, MTP or PTP
- **Field-Upgradeable Firmware**
  - Upgradeable to future compressed audio and video codecs via software
- **Ready for Wi-Fi 802.11a/b/g Using SDIO**
- **Ready for Bluetooth Using SDIO or UART**

1.3. Design Support

- **Green Hills Integrated Development Environment, Software Development Kit (SDK), and Debugger**
  - Optional real-time trace port
- **Application Notes, Reference Schematics, Sample PCB Layouts are Available**

1.4. Additional Documentation

Additional documentation and information is available from SigmaTel, including the following:

- **Extensive Software Development Kit (SDK) with Peripheral Device Files**
- **Application Notes**
- **Reference Schematics**
- **Sample Printed Circuit Board (PCB) Layouts**
- **Sample Bill of Materials**

SigmaTel specifically refers the reader to the peripheral device *Include* files from the SDK. These files provide constant declarations for address offsets to the registers defined in this document. Note that the name of each programmable register defined in this datasheet corresponds to a C language *#define* or assembly language *equate* of the exact same name. These files also contain declarations that allow symbolic access to individual bit fields within the registers.

User programs can include all of these peripheral include files by simply including the file *hw_equ.inc* into the assembly files and *hw_equ.h* into the C files.

1.5. STMP36xx System Block Diagram

Figure 1 shows a block diagram of a typical system based on the STMP36xx. Figure 2 shows photographs of the two different chip packages.

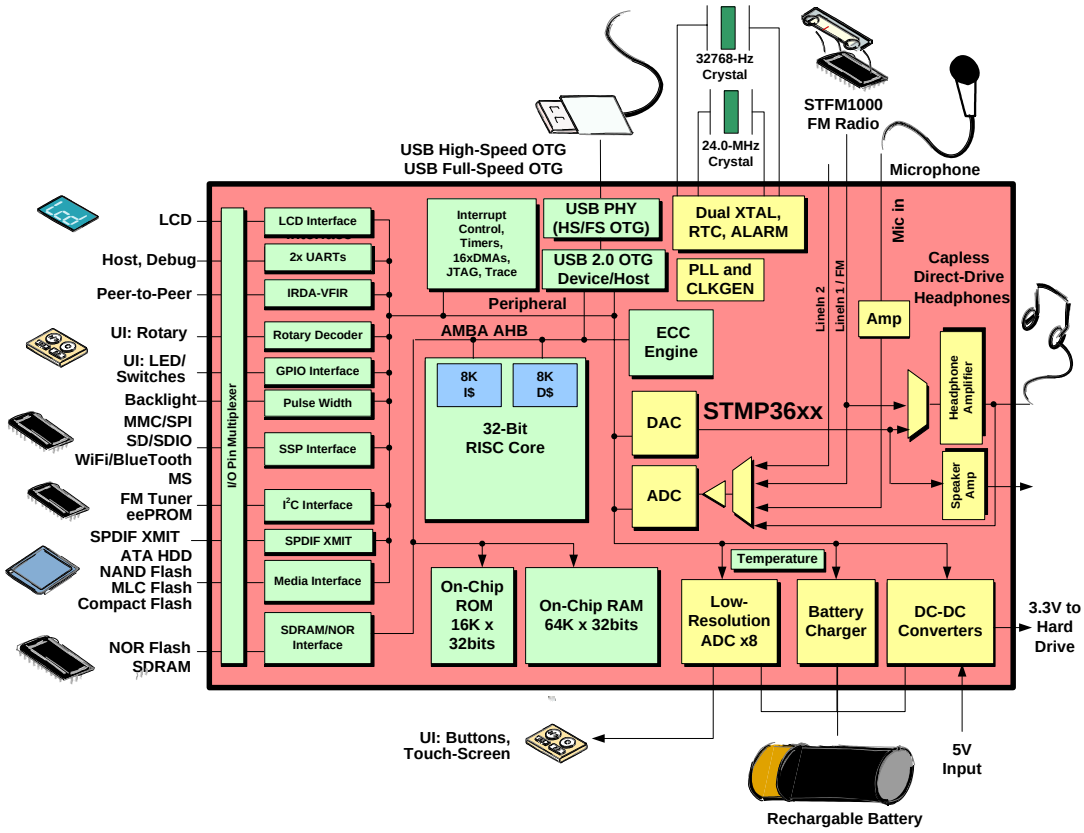
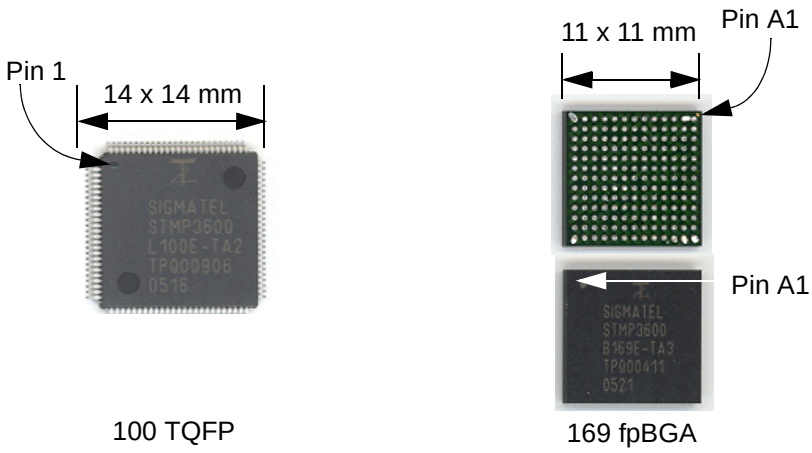


Figure 1. System Block Diagram



Note: For additional package measurements, see Chapter 36, "Package Drawings" on page 843.

Figure 2. Chip Package Photographs

1.6. STMP36xx Product Features

The STMP36xx is SigmaTel's fourth generation single-chip digital media system for applications such as digital audio players, PDAs, voice recorders, cell phones, portable video players, and digital photo wallets. The STMP36xx offers long battery life, minimal external components, high processing performance, and excellent software development and debug support.

The STMP36xx features low power consumption to enable long battery life in portable applications. The integrated power management unit includes a high efficiency, on-chip DC-DC converter that supports many different battery configurations including 1xAA, 1xAAA, 2xAA, 2xAAA and Li-Ion. The power management unit also includes an intelligent battery charger for Li-Ion cells and is designed to support Adaptive Voltage Control (AVC), which can reduce system power consumption by half. AVC also allows the chip to operate at a higher peak CPU operating frequency than typical voltage control systems.

To provide the maximum application flexibility, the STMP36xx integrates a wide range of I/O ports. It can efficiently interface to nearly any type of flash memory, ATA drive, serial bus, or LCD. It is also ready for advanced connectivity applications such as Bluetooth and WiFi via its integrated 4-bit SDIO controller and high-speed UART.

As with previous STMP3xxx products, the STMP36xx integrates the entire suite of analog components needed for a portable audio player. This includes a high-resolution audio codec with headphone and speaker amplifiers, 8-channel 12-bit ADC, high-current battery charger, linear regulators for 5-V operation, high-speed USB OTG PHY, and various system monitoring and infrastructure systems.

An ARM 926 EJ-S CPU with 256 Kbytes of on-chip SRAM and an integrated memory management unit provides the processing power needed to support advanced features such as audio cross-fading, as well as still and motion video decoding. These and other advanced features are integrated into software development kits that support the STMP36xx. Contact your local SigmaTel representative or visit the SigmaTel extranet at: http://extranet.sigmatel.com for more information on the software development kits available for the STMP36xx.

1.6.1. ARM 926 Processor Core

The on-chip RISC processor core is an ARM, Ltd. 926EJ-S. This CPU implements the ARM v5TE instruction set architecture. The ARM9EJ-S has two instructions sets, a 32-bit instruction set used in the ARM state and a 16-bit instruction set used in Thumb state. The core offers the choice of running in the ARM state or the Thumb state or a mix of the two. This enables optimization for both code density and performance. ARM studies indicate that Thumb code is typically 65% the size of equivalent ARM code, while providing 160% of the effective performance in constrained memory bandwidth applications. The ARM CPU is described in Chapter 3, "ARM CPU Complex" on page 43.

The ARM RISC CPU is the central controller for the entire STMP36xx SOC, as shown in Figure 3. The ARM 926 core includes two AHB masters. One is used for instruction fetches while the other is used for data load/stores, page table accesses, DMA traffic, etc. The AHB has three other bus masters, the ARM core I and D masters, the USB master, and an AHB-to-APBH bridge DMA master, and an AHB-to-APBX bridge DMA master. The AHB has six slaves: the USB slave, the on-chip RAM, the on-chip ROM, the external memory interface (SDRAM), default first-level page table, and the two APB bridges.

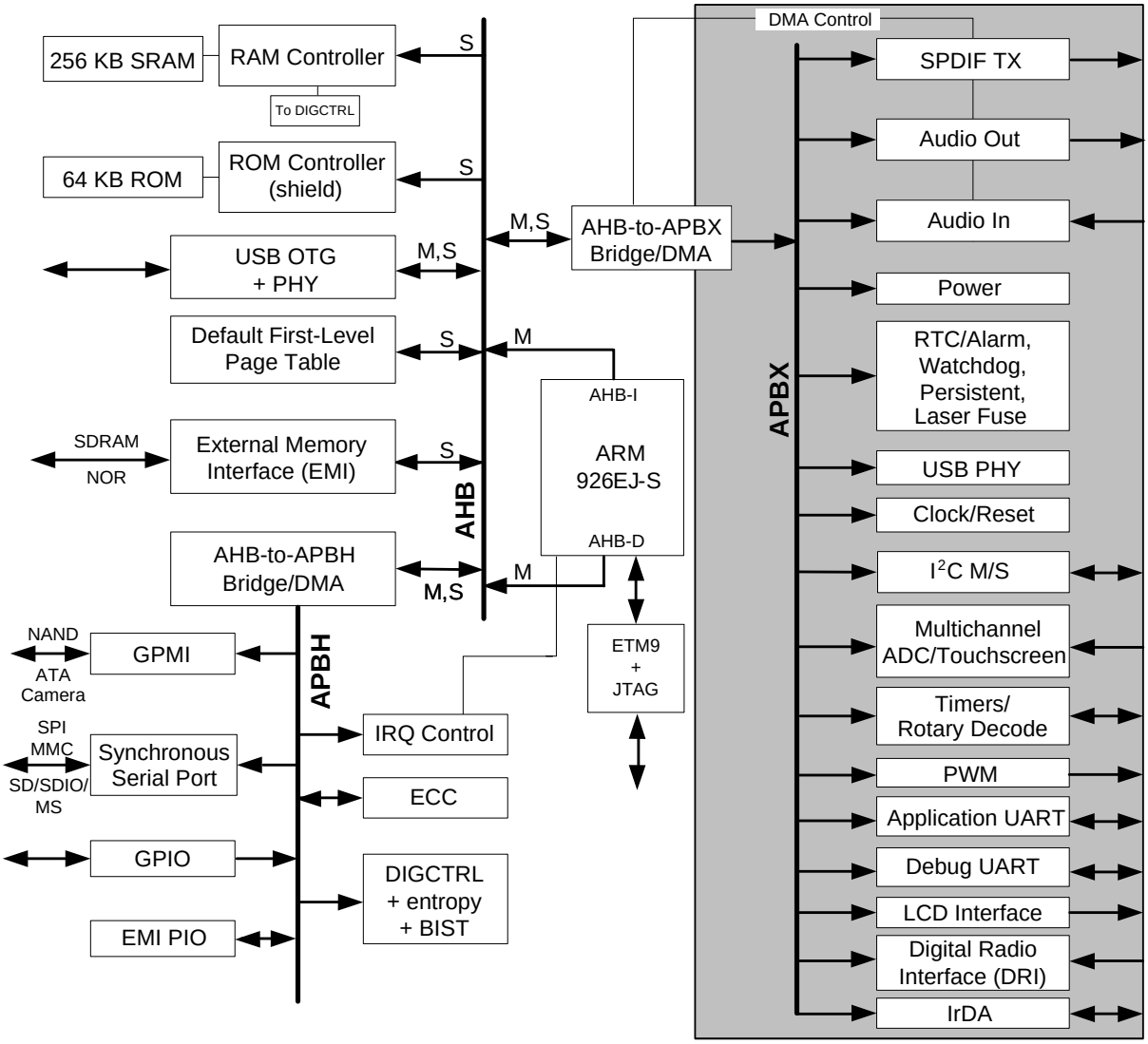


Figure 3. STMP36xx SOC Block Diagram

Execution always begins in on-chip ROM after reset, unless over-ridden by the debugger. There is a 2-Kbyte lock out region (ROM shield) within the on-chip ROM. This region is not visible when the JTAG debugger is attached. It is further disabled by a write-once bit in the digital control block. When written by instructions in the ROM boot loader, this feature enables the expansion of a trust zone to a certified boot manager software loaded by the ROM.

A number of devices are programmed only at initialization or application state change, such as DC-DC converter voltages, clock generator settings, etc. Certain other devices either operate in the crystal clock domain or have significant portions that operate in the crystal clock domain, e.g., ADC, DAC, PLL, etc. These devices operate on a slower speed asynchronous peripheral bus. Write posting in the ARM

core, additional write post buffering in the peripheral AHB, and set/clear operations at the device registers make these operations efficient.

Figure 4 shows the memory map for the D-AHB devices.

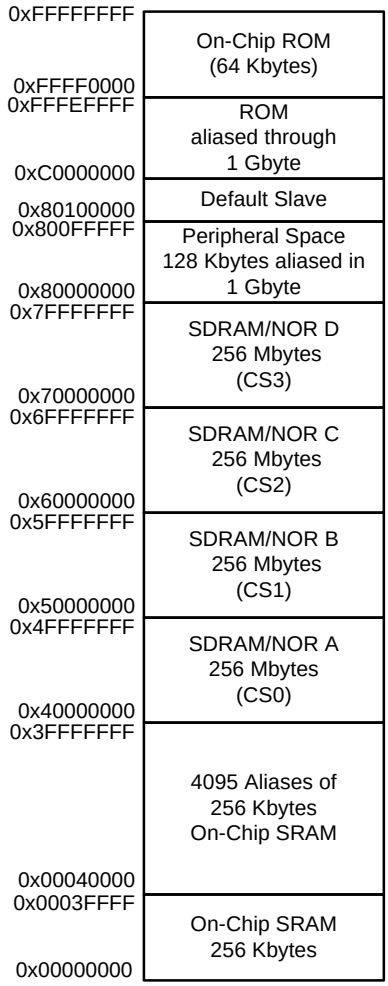


Figure 4. Memory Map for D-AHB Devices

The DC-DC converters and the clock generator can be reprogrammed on-the-fly to dynamically trade off power versus performance.

1.6.2. On-Chip RAM and ROM

The STMP36xx includes 64Kx32-bit on-chip RAM. The RAM includes embedded redundancy. Any necessary RAM repairs are done by the ROM, as described in Chapter 7, “Digital Control and On-Chip RAM” on page 123.

The STMP36xx also includes 16K 32-bit words of on-chip masked programmed ROM. The ROM contains initialization code written by SigmaTel, Inc. to handle the initial boot and hardware initialization. Software in this ROM offers a large number of BOOT configuration options, including manufacturing boot modes for burn-in and tester operation.

Other boot modes are responsible for loading application code from off-chip into the on-chip RAM. It supports initial program loading from a number of sources:

- NAND flash devices
- NOR flash devices
- ATA hard drive
- I^2C master mode from EEPROM devices
- USB recovery mode

At power-on time, the first instruction executed by the ARM core comes from this ROM. The reset boot vector is located at 0xFFFF0000. The on-chip boot code includes a firmware recovery mode. If the device fails to boot from NAND flash, NOR flash, or hard drive, for example, the device will attempt to boot from a PC host connected to its USB port.

The on-chip RAM and ROM run on the AHB HCLK domain. The maximum HCLK frequency is 100 MHz. At this frequency, the on-chip RAM and ROM can supply a maximum of 400 Mbytes per second.

The ROM boot loader can boot images from different devices, depending on the boot modes. This function is enabled by different boot mode pin configurations. Additional laser fuse bits select one of 16 customer keys, which are further modified by laser fuses. A polynomial LSFR is used to decrypt the supplied boot image. A second polynomial computes an authentication code for the boot image. If the decrypted image does not compute the correct authentication code, then the boot loader will enter recovery mode and attempt to boot from the USB device.

These features are described in Chapter 32, “Boot Modes” on page 787.

1.6.3. Interrupt Collector

The STMP36xx contains a 64-bit vectored interrupt collector for the CPU’s IRQ input and a separate non-vectored interrupt collection mechanism for the CPU’s FIQ input. Each interrupt can be assigned to one of four levels of priority. The interrupt collector supports nesting of interrupts that preempt an interrupt service routine running at a lower priority level.

The interrupt collector is described in Chapter 5, “Interrupt Collector” on page 69.

1.6.4. Default First-Level Page Table

The STMP36xx contains a default first-level page table implemented as an AHB slave. This device provides an economical way to present 16 Kbytes of nearly static data to the ARM CPU’s MMU. The default first-level page table provides access to the PIO block at 0x80000000, as well as up to 16 Mbytes of virtual memory defined in up to 16 secondary page tables.

This feature is described more completely in Chapter 6, “Default First-Level Page Table for ARM926 MMU” on page 115.

1.6.5. External Memory Interface (SDRAM/NOR Flash Controller)

The STMP36xx contains an external memory interface (EMI) controller that can be used to connect external SDRAM memory chips. The controller is designed to work with 16 bit wide memory systems. It supports SDRAM products from 16Mbit to 512MBit JEDEC families. The EMI also supports external NOR flash devices at up to 512Mbit per chip.

The EMI's AHB slave supports split transactions to improve system wide performance and overlap with the on-chip RAM.

The external memory interface is described in Chapter 12, "External Memory Interface (EMI)" on page 325.

1.6.6. DMA Controller

Many peripherals on the STMP36xx utilize direct memory access (DMA) transfers. Some peripherals, such as the USB controller, make highly random accesses to system memory for a large number of descriptor, queue heads, and packet payload transfers. This highly random access nature is supported by integrating a dedicated DMA into the USB controller and connecting it directly to the high-speed AHB bus.

Other peripherals have a small number of highly sequential transactions, for example the ADC or DAC streams, SPDIF transmitter, etc. These devices share a centralized address generation and data transfer function that allows them to share a single shared master on the AHB.

There are two AMBA peripheral buses on the STMP36xx:

- The APBH bus runs completely synchronous to the AHB's HCLK.
- The APBX bus runs in an independent clock domain that can be slowed down significantly for power reduction.

Thus, the AHB and APBH can run at 60 MHz, while the APBX runs at 6 MHz. See Chapter 10, "AHB-to-APBH Bridge with DMA" on page 185, and Chapter 11, "AHB-to-APBX Bridge with DMA" on page 257, for more detailed information.

The two bridge DMAs are controlled through linked DMA command lists. The CPU sets up the DMA command chains before starting the DMA. The DMA command chains include set-up information for a peripheral and associated DMA channel. The DMA controller reads the DMA command, writes any peripheral set up, tells the peripheral to start running and then transfers data, all without CPU intervention. The CPU can add commands to the end of a chain to keep data moving without interventions.

The linked DMA command architecture offloads most of the real-time aspects of I/O control from the CPU to the DMA controller. This provides better system performance, while allowing longer interrupt latency tolerances for the CPU.

1.6.7. Clock Generation Subsystem

The STMP36xx uses twenty-five domains to provide clocks to the various subsystems, as shown in Figure 5. These clocks are either derived from the 24-MHz crystal or from the integrated high-speed PLL. The PLL output is programmable from 240 MHz to 480 MHz in 4 MHz steps. The PLL must be set to 480 MHz for USB or SPDIF operation.

More details about the system clock architecture can be found in Chapter 4, "Clock Generation and Control" on page 47.

The system includes a real-time clock that can use either the 24-MHz system crystal or a 32.768-kHz RTC crystal. An integrated watchdog reset timer is also available for automatic recovery from errant code execution. See Chapter 19, "Real-Time Clock, Alarm, Watchdog, and Persistent Bits" on page 497 for more information about these features.

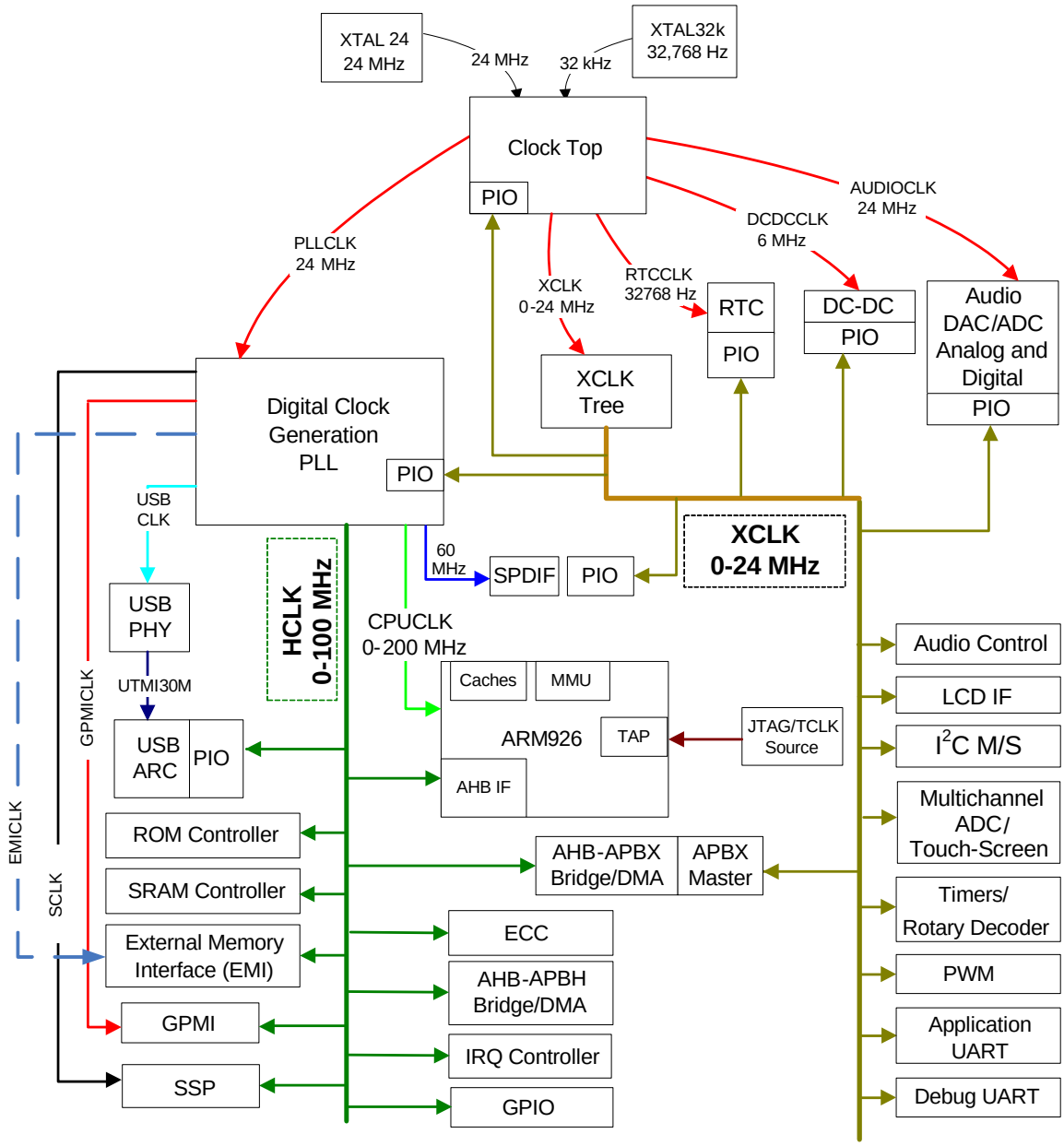


Figure 5. Clock Diagram

1.6.8. Power Management Unit

The STMP36xx contains a sophisticated Power Management Unit (PMU), including two integrated DC-DC converters and two linear regulators. The PMU can operate from a battery (1-cell, 2-cell, or Li-Ion) using the DC-DC converter(s) or a 5-V supply using the linear regulators and can automatically switch between them without interrupting operation. The PMU includes circuits for battery and system voltage brown-out detection, as well as on-chip temperature, digital speed, and process monitoring.

The chip has two programmable integrated DC-DC converters that can be used to provide power for the device *as well as the entire application*. The converters can be configured to operate from standard battery chemistries in the range of 0.9-4.2 volts including alkaline cells, NiMH, Li-Ion, etc. These converters use off-chip reactive components (L/C) in a pulse width or frequency modulated DC-DC converter.

The real-time clock includes an alarm function that can be used to “wake-up” the DC-DC converters, which will then wake up the rest of the system.

The power subsystem is described in Chapter 31,“Power Supply” on page 747.

1.6.9. USB Interface

The chip includes a high-speed Universal Serial Bus (USB) version 2.0 controller and integrated USB Transceiver Macrocell Interface (UTMI) PHY. The STMP36xx device interface can be attached to USB 2.0 hosts and hubs running in the USB 2.0 high-speed mode at 480Mbit/second. It can be attached to USB 2.0 full-speed interfaces at 12Mbit/second.

The USB controller and integrated PHY support high-speed OTG modes for peer-to-peer file interchange. The STMP36xx has a high-current PWM channel that can be used with low-cost external components to generate up to 8mA of 5 volts on the OTG VBUS for OTG session initiation. The USB controller can also be configured as a high-speed host.

The USB subsystem is designed to make efficient use of system resources within the STMP36xx. It contains a random access DMA engine that reduces the interrupt load on the system and reduces the total bus bandwidth that must be dedicated to servicing the eight on-chip physical endpoints.

It is a dynamically configured port that can support up to 5 endpoints, each of which may be configured for bulk, interrupt, or isochronous transfers. The USB configuration information is read from on-chip memory via the USB controller’s DMA.

See Chapter 8, “USB High-Speed On-the-Go (Host/Device) Controller” on page 155 and Chapter 9, “Integrated USB 2.0 PHY” on page 161 for more information.

1.6.10. General-Purpose Media Interface (GPMI)

The chip includes a general-purpose media interface (GPMI) controller that supports NAND and ATA devices. The NAND flash interface provides a state machine that provides all of the logic necessary to perform DMA functions between on- or off-chip RAM and up to four NAND flash devices. The controller and DMA are sophisticated enough to manage the sharing of a single 16 bit wide data bus among 4 NAND devices without detailed CPU intervention. This allows the STMP36xx to provide unprecedented levels of NAND performance. The GPMI’s ATA mode provides a high-speed link to a hard drive or CD-ROM. It supports PIO-4 and UDMA mode 4 (up to 66MB/s).

The general-purpose media interface can be described as two fairly independent devices in one. Unlike previous generations, the three operating modes are integrated into one overall state machine that can freely intermix cycles to different device types on the media interface. There are four chip selects on the media interface. Each chip select can be programmed to have a different type device installed, as shown in Table 1. For NAND MP3 player applications, these might be all NAND flash devices.

The chip selects are shared with the external memory interface, so that chip select pins that are unused by the GPMI for NAND or ATA devices can be used by the EMI

for SDRAM or NOR flash devices. The two interfaces have independent data and control paths, so that simultaneous transfers can take place on both GPMI and the EMI.

Table 1. Media Interface Options by Application

CHIP SELECT	NAND PLAYER	HARD DISK
0	NAND	Hard Disk
1	NAND	Hard Disk
2	NAND	SDRAM
3	NAND	NOR

The GPMI pin timings are based on a dedicated clock divider from the PLL, allowing the CPU clock divider to change without affecting the GPMI.

See Chapter 13, "General-Purpose Media Interface (GPMI)" on page 341 for more information.

1.6.11. Hardware Acceleration for ECC for Robust External Storage

The forward error correction circuit (ECC) is used to provide STMP36xx applications with a reliable interface to various storage media that would otherwise have unacceptable bit error rates. The ECC module consists of two different error correcting code processors:

- 1-bit error correcting Samsung SSFDC (Hamming code) encoder/decoder
- 4-symbol error correcting (9 bits/symbol) Reed-Solomon encoder/decoder

The 1-bit hamming code is SSFDC compliant and can be used with most SLC NAND flash memories. This code is capable of correcting a single bit or detecting two incorrect bits over a 256-byte block.

The Reed-Solomon mode is used for memories that have a higher native defect probability, such as MLC NAND. It can correct up to four 9-bit symbols over a 512-byte block.

Both of these error correction encoder/decoders use DMA transfers to move data to and from on-chip RAM completely in parallel with the CPU performing other useful work.

The ECC reads source data blocks and parity bytes from the shared AHB master, decodes the error correction code, and generates an error report telling the CPU which, if any, bits need to be modified.

See Chapter 14, "Hardware ECC Accelerator (HWECC)" on page 361 for more information.

1.6.12. Memory Copy Unit

The SOC contains a memory-to-memory copy controller using two channels of the DMA. The source can be either from the on-chip RAM, ROM, external SDRAM, or NOR flash. SImilarly, the destination can target either on-chip or off -chip RAM.

See Chapter 30, "Memory Copy Device" on page 741 for more information.

1.6.13. Mixed Signal Audio Subsystem

The STMP36xx contains an integrated high-quality mixed signal audio subsystem, including high-quality sigma delta D/A and A/D converters, as shown in Figure 6. The D/A converter is the mainstay of the audio decoder/player product application, while the A/D converter is used for voice recording and MP3 encoding applications.

The chip includes a low-noise headphone driver that allows it to directly drive low impedance (8Ω or 16Ω) headphones. The direct drive, or "capless" mode, removes the need for large, expensive DC blocking capacitors in the headphone circuit. The headphone power amplifier can detect headphone shorts and report them via the ICOLL interrupt system. A digitally programmable master volume control allows user control of the headphone volume. Annoying clicks and pops are eliminated by zero crossing updates in the volume/mute circuits and by headphone driver startup and shutdown circuits.

The microphone circuit has a mono to stereo programmable gain pre-amp and an optional microphone bias generator.

These features are described in Chapter 25, "AUDIOIN/ADC" on page 623, and Chapter 26, "AUDIOOUT/DAC" on page 647.

1.6.14. Master Digital Control Unit (DIGCTL)

The Master Digital Control Unit (DIGCTL) provides control registers for a number of blocks that do not have their own AHB or APB slaves, notably the on-chip RAM and on-chip ROM controllers. In addition, it provides control registers for the SDRAM controller. Finally, it provides several security features, including an entropy register, as well as the ROM shield and JTAG shield trust zone controls. See Chapter 7, "Digital Control and On-Chip RAM" on page 123 for more information.

1.6.15. Synchronous Serial Port (SSP)

The SSP supports a wide range of synchronous serial interfaces, including:

- 4-bit high-speed MMC/SD/SDIO
- SPI
- 1-bit MS
- TI SSI
- Microwire

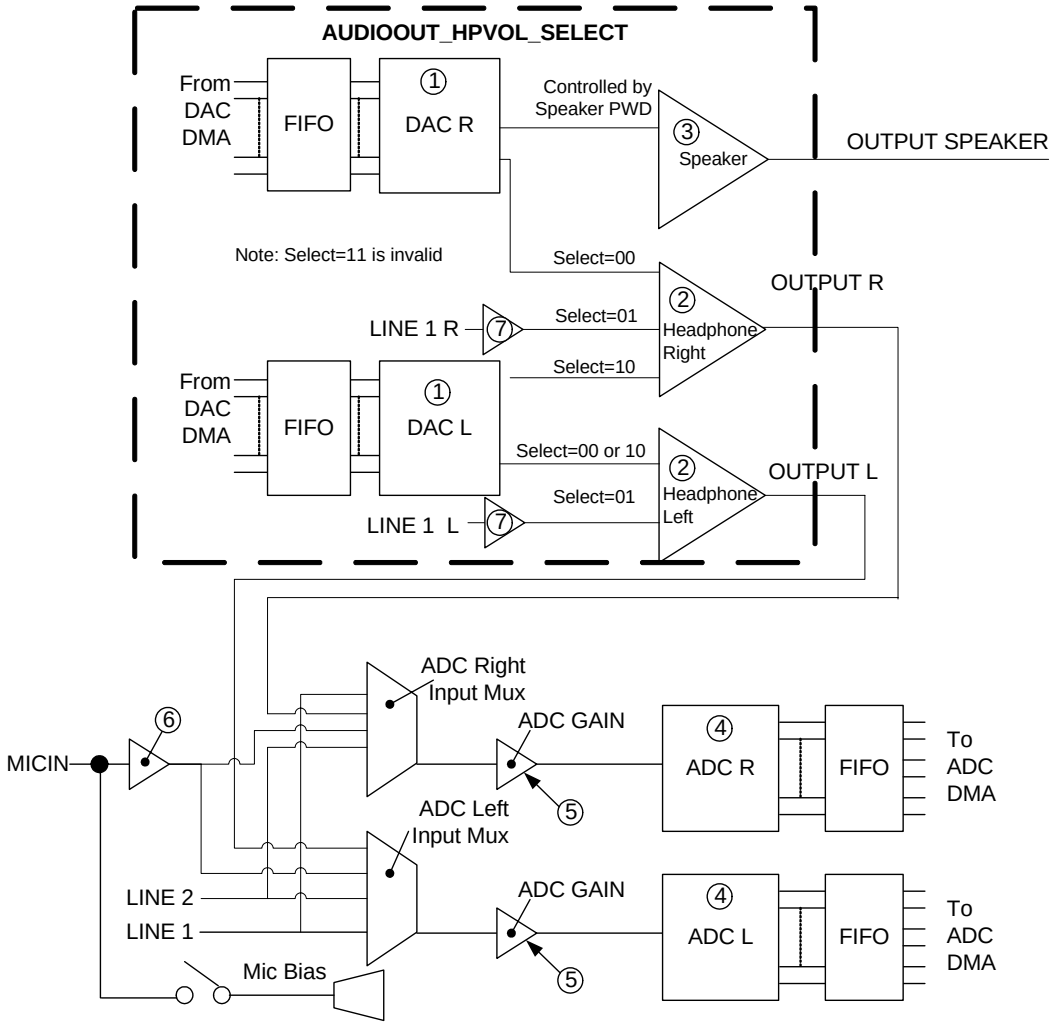
The SSP has a dedicated DMA channel and a dedicated clock divider from the PLL.

See Chapter 15, "Synchronous Serial Port (SSP)" on page 383 for more information about these features.

1.6.16. I^2C Interface

The chip contains a two-wire SMB/I^2C bus interface. It can act as either a slave or master on the SMB interface. The on-chip ROM supports boot operations from I^2C mastered EEPROMs, as well as slave I^2C boot mode.

See Chapter 21, "I2C Interface" on page 539 for more information.



Notes:

1. audioout_dacvolume: Digital volume control.
2. audioout_hpvol: Analog volume control.
3. audioout_spkrvol: Analog volume control that works on the speaker amp output.
4. audioin_adcvolume: Digital volume control.
5. audioin_adcvol: Analog volume control that controls the ADC Gain block.
6. audioin_micline_micgain: Analog volume control that controls the mic amp.
7. atten_line bit

Figure 6. Mixed Signal Audio Elements

1.6.17. General-Purpose Input/Output (GPIO)

The STMP36xx contains 85 GPIO pins in the 169-pin package. Most digital pins that are available for specific functions, for example, the SDRAM interface, are also available as GPIO pins if they are not otherwise used in a particular application.

See Chapter 17, “Pin Control and GPIO” on page 429 for more information

1.6.18. LCD Controller

The LCD controller has a dedicated DMA channel and can be used to transfer data directly to 8- or 16-bit LCD modules. It has programmable pin timing, support for 8080 and 6800 modes and automatically handshakes transfers to the LCD.

See Chapter 16, "LCD Interface (LCDIF)" on page 419 for more information.

1.6.19. SPDIF Transmitter

The STMP36xx includes a Sony-Philips Digital Interface Format (SPDIF) transmitter. It includes independent sample-rate conversion hardware so that the A/D, D/A, and SPDIF can run simultaneously. The SPDIF has a dedicated DMA channel. The SPDIF has its own clock divider from the PLL.

See Chapter 27, "SPDIF Transmitter" on page 679 for more information.

1.6.20. Rotary Decoder

An automatic rotary decoder function is integrated into the chip. Two digital inputs are monitored to determine which is leading and by how much. In addition, the hardware automatically determines the period for rotary inputs.

See Chapter 18, "Timers and Rotary Decoder" on page 475 for more information.

1.6.21. Dual UARTs

Each of two UARTs, similar to a 16550 UART, are provided—one for application use and one for debug use. Both UARTs are high-speed with 16-byte Rx and Tx FIFOs. The Application UART supports DMA and flow control (CTS/RTS).

See Chapter 22, "Application UART" on page 571, and Chapter 23, "Debug UART" on page 589 for more information.

1.6.22. Infrared Interface

The infrared interface supports Serial Infrared (SIR), Mid Infrared (MIR), Fast Infrared (FIR), and Very Fast IrDA (VFIR) rates. It shares pins and DMA channels with the application UART.

See Chapter 24, "IrDA Controller" on page 607 for more information.

1.6.23. Low-Resolution ADC and Touch-Screen Interface

Eight channels of 12-bit resolution analog-to-digital conversion are provided. Channel 7 is always connected to the battery and cannot be used for any other purpose other than battery voltage measurement. Channel 6 can be configured to monitor a number of internal system parameters. The remaining six channels are available for other uses and can be used for resistive button sense, touch screens, or other analog input. Channels 0 and 1 have integrated drivers for external temperature monitor thermistors. Channels 2–5 have integrated drivers for resistive touch-screens. The LRADC provides typical performance of 11-bit no-missing-codes and 9-bit SNR.

See Chapter 29, "Low-Resolution ADC and Touch-Screen Interface" on page 705 for more information.

1.6.24. Pulse Width Modulator (PWM) Controller

The STMP36xx contains four PWM output controllers that can be used in place of GPIO pins. Applications include LED brightness control and high-voltage generators for electroluminescent lamp (EL) display backlights. Independent output control of

each phase allows zero, one, or high-Z to be independently selected for the active and inactive phases. Individual outputs can be run in lock step with guaranteed non-overlapping portions for differential drive applications.

See Chapter 20, “Pulse-Width Modulator (PWM) Controller” on page 523 for more information.

1.6.25. Camera Interface

The GPMI (ATA/NAND interface) has an experimental mode to support a standard digital camera. When camera data is being downloaded, the GPMI cannot perform other functions. Contact your SigmaTel representative for more information before using this feature.

2. CHARACTERISTICS AND SPECIFICATIONS

This chapter describes the characteristics and specifications of the STMP36xx and includes sections on absolute maximum ratings, recommended operating conditions, and DC characteristics.

2.1. Absolute Maximum Ratings

Table 2. Absolute Maximum Ratings

PARAMETER	MIN	MAX	UNITS
Storage Temperature	–40	125	°C
Battery Pin (BATT)—DC-DC Modes 0 and 1	–0.3	4.242	V
Battery Pin (BATT)—DC-DC Mode 2	–0.3	3.40	V
Battery Pin (BATT)—DC-DC Mode 3	–0.3	1.98	V
5-Volt Source Pin (VDD5V)	–0.3	5.25	V
PSWITCH—DC-DC Mode 3 (Note 1)	–0.3	BATT	V
PSWITCH—All Other DC-DC Modes (Note 2)	–0.3	BATT/2	V
Analog/Digital Supply Voltage (VDDA1, VDDD1, VDDD2, VDDD3)	–0.3	1.98	V
I/O Supply (VDDIO1, VDDIO2, VDDIO3, VDDIO4)	–0.3	3.63	V
DC-DC Converter #1 (DCDC_VDDD)	–0.3	1.98	V
DC-DC Converter #1 (DCDC_VDDIO)—DC-DC Mode 0	–0.3	4.242	V
DC-DC Converter #1 (DCDC_VDDIO)—All Other DC-DC Modes	–0.3	3.63	V
DC-DC Converter #1 (DCDC1_BATT)	–0.3	max (VDDIO, BATT)	V
DC-DC Converter #2 (DCDC2_VDDIO)	–0.3	3.63	V
DC-DC Converter #2 (DCDC2_PFET)	–0.3	4.242	V
Input Voltage on DCDC_MODE Input Pin Relative to Ground (Note 2)	–0.3	BATT	V
Input Voltage on Any Digital I/O Pin Relative to Ground (DIO3) (Note 2)	–0.3	VDDIO+0.3	V
Input Voltage on USB D+, D– Pins Relative to Ground (USBIO) (Note 2)	–0.3	3.63	V
Input Voltage on Any Analog I/O Pin Relative to Ground (AIO) (Note 2)	–0.3	VDDA+0.3	V

Notes:

1. PSWITCH can tolerate VDDIO driven through a 47k resistor, as the on-chip circuitry prevents the actual voltage on the pin from exceeding acceptable levels.
2. Pin sets for DCDC_MODE, DIO3, AIO, and USBIO, are defined in the pin list in Chapter 35, beginning on page 809.

2.2. Recommended Operating Conditions

Table 3. Recommended Operating Conditions

PARAMETER	MIN	TYP	MAX	UNITS
Ambient Operating Temperature (Note 1)	–10		70	°C
Digital/Analog Core Supply Voltage—VDDD1, VDDD2, VDDD3, VDDA1. *Specification dependent on frequency.* (Note 2)	1.35	-	1.98	V
Digital I/O Supply Voltage—VDDIO1, VDDIO2, VDDIO3, VDDIO4	2.90	3.0	3.63	V
Minimum Battery Startup Voltage:				
DC-DC Mode 0	-	3.1	-	V
DC-DC Mode 1	-	2.9	-	V
DC-DC Mode 3	-	0.9	-	V
Standby Current (Note 3):				
DC-DC Mode 0 (32-kHz RTC off), BATT = 4.2 V	-	45		µA
DC-DC Mode 0 (32-kHz RTC on), BATT = 4.2 V	-	43		µA
DC-DC Mode 3 (32-kHz RTC off), BATT = 1.6 V	-	3		µA
DC-DC Mode 3 (32-kHz RTC on), BATT = 1.6 V	-	5		µA
Microphone:				
Full-Scale Input Voltage (0 dB gain)	-	0.6	-	Vrms
Full-Scale Input Voltage (20 dB gain)	-	0.06	-	Vrms
Full-Scale Input Voltage (40 dB gain)	-	0.006	-	Vrms
Input Resistance	-	100	-	kΩ
Line Inputs:				
Full-Scale Input Voltage (Note 4)	-	0.6	-	Vrms
Crosstalk between Input Channels (16Ω load)	-	–75	-	dB
Input Resistance (Note 5)	-	50	-	kΩ
LineIn-to-HP SNR Idle Channel (Note 6)	95	99	-	dB
ADC SNR Idle Channel (Note 6)	-	85	-	dB
ADC –60 dB Dynamic Range (Note 6)	-	85	-	dB
Headphone:				
Full-Scale Output Voltage (VDDA = 1.8 V, 16Ω load)	-	0.54	-	Vrms
Full-Scale Output Voltage (VDDA = 1.35 V, 16Ω load)	-	0.42	-	Vrms
Output Resistance	-	-	<1	Ω
THD+N (16Ω load)	-	–79	–66	dB
THD+N (10KΩ load)	-	–84	-	dB
DAC SNR Idle Channel (Note 6)	-	99	-	dB
DAC –60 dB Dynamic Range (Note 6)	95	99	-	dB
Speaker:				
Full-Scale Output Voltage (VDDA = 1.8 V, 8Ω load)	-	0.83	-	Vrms
Full-Scale Output Voltage (VDDA = 1.35 V, 8Ω load)	-	0.62	-	Vrms
Full-Scale Output Voltage (VDDA = 1.8 V, 4Ω load)	-	0.62	-	Vrms
Full-Scale Output Voltage (VDDA = 1.35 V, 4Ω load)	-	0.45	-	Vrms
Output Resistance	-	-	<1	Ω
THD+N (8Ω load)		–66		dB
THD+N (4Ω load)		–60		dB
SNR Idle Channel (8Ω load) (Note 6)		90		dB
SNR Idle Channel (4Ω load) (Note 6)		90		dB

Notes:

1. Contact SigmaTel for extended temperature range options. In most system designs, battery and display specifications will limit the operating range to well within these specifications. Most battery manufacturers recommend enabling battery charge only when the ambient temperature is between 0° and 40°C. To ensure that battery charging does not occur outside the recommended temperature range, the player ambient temperature may be monitored by connecting a thermistor to the LRADC0 or LRADC1 pin on the STMP36xx.
2. These limits should be guard-banded by 100 mV. Recommended operating voltages for CPUCLK can be found in Table 4. Recommended operating voltages for HCLK can be found in Table 6.
3. When the real-time clock is enabled, the chip consumes current when in the OFF state to keep the crystal oscillator and the real-time clock running. With a typical 2850 mAhour AA battery, this OFF state standby current would take more than one year to drain the battery fully.
4. At 1.35 VddA, max input is 0.45 Vrms.
5. Input resistance changes with volume setting: 20KΩ at +12 dB, 50KΩ at 0 dB, 100KΩ at –34.5 dB.
6. Measured “A weighted” over a 20-Hz to a 20-kHz bandwidth, relative to full scale output voltage (when VDDA = 1.8 V).

2.2.1. Recommended Operating Conditions for Specific Clock Targets

Use the tables in this section to select a proper setting for VDDD and VDDD brownout voltages based on standard analysis of worst case design and characterization data.

Notes: *VDDD must be set to the higher of the voltages listed in Table 4, Table 5, Table 6, and Table 7 for the specific clock targets. Measured supply voltage may not match the programmed value; see Table 982, “HW_POWER_VDDCTRL Bit Field Descriptions,” on page 767 for more information.*

Table 4. Recommended Operating Conditions for Specific CPUCLK Targets

Max CPUCLK Target (MHz)	Min VDDD Target Voltage	HW_POWER_VDDCTRL_VDDD_TRG (Using DC-DC Converters)	Corresponding VDDD Brownout Voltage	HW_POWER_VDDCTRL_VDDD_BO
up to 150	1.440	0xD	1.344	0xA
160	1.472	0xE	1.376	0xB
170	1.536	0x10	1.440	0xD
180	1.600	0x12	1.504	0xF
190	1.664	0x14	1.568	0x11
200	1.696	0x15	1.600	0x12

Table 5. PLL Voltage Requirements

PLL Frequency (MHz)	HW_CPUCTRL_PLLCTRL0_FREQ	HW_CPUCTRL_PLLCTRL0_PLLV2ISEL	Min VDDD Target Voltage	HW_POWER_VDDCTRL_VDDD_TRG (Using DC-DC Converters)	Corresponding VDDD Brownout Voltage	HW_POWER_VDDCTRL_VDDD_BO
240–300	0x0F0–0x12C	0x2	1.376	0xB	1.312	0x9
304–360	0x130–0x168	0x2	1.504	0xF	1.408	0xC
364–400	0x16C–0x190	0x2	1.600	0x12	1.504	0xF
404–480	0x194–0xIE0	0x2	1.888	0x1B	1.792	0x18
300–480	0x12C–0xIE0	0x0	1.376	0xB	1.312	0x9

Note: PLLV2ISEL transitions from 0x2 to 0x0 are recommended only when also changing FREQ from 240 to 480 MHz. Similarly, PLLV2ISEL transitions from 0x0 to 0x2 are recommended only when changing FREQ from 480 to 240 MHz.

Table 6. Recommended Operating Conditions for Specific HCLK Targets

Max HCLK Target (MHz)	Min. VDDD Target Voltage	HW_POWER_VDDCTRL_VDDD_TRG (Using DC-DC Converters)	Corresponding VDDD Brownout Voltage	HW_POWER_VDDCTRL_VDDD_BO
up to 80	1.472	0xE	1.376	0xB
81–90	1.504	0xF	1.408	0xC
91–100	1.600	0x12	1.504	0xF

Table 7. Recommended Operating Conditions for Specific EMICLK Targets

VDDIO Target Voltage	Max EMICLK[1] Target (MHz)	PLL	Min VDDD Target Voltage	HW_POWER_VDDCTRL_VDDD_TRG (Using DC-DC Converters)	Corresponding VDDD Brownout Voltage	HW_POWER_VDDCTRL_VDDD_BO
3.585	up to 24	on or off	1.472	0xE	1.376	0xB
3.585	30–60	on	1.504	0xF	1.408	0xC
3.585	70	on	1.536	0x10	1.440	0xD
3.585	80	240 MHz only	1.600	0x12	1.504	0xF
3.329	up to 24	on or off	1.472	0xE	1.376	0xB
3.329	30–60	on	1.472	0xE	1.376	0xB
3.329	70	on	1.504	0xF	1.408	0xC
3.329	80	240 MHz only	1.504	0xF	1.408	0xC
3.073	up to 24	on or off	1.472	0xE	1.376	0xB
3.073	30–70	on	1.472	0xE	1.376	0xB
3.073	80	on	1.504	0xF	1.408	0xC
2.945	up to 24	off	1.440	0xD	1.344	0xA

Note: 1. EMICLK = HCLK = CPUCLK

After split-lot characterization of the part performance versus speed-sensor values, a closed-loop method for setting VDDD voltage and brownout levels will be provided that allows VDDD settings to be tuned to the actual process corner of a part, at the then current ambient temperature and voltage.

2.3. DC Characteristics

Table 8. DC Characteristics

PARAMETER	MIN	TYP	MAX	UNITS
Power Dissipation: VDDD = 1.35 V, VDDA = 1.35 V, VDDIO = 3.3 V, DC-DC_MODE = 00 (Li-Ion H), VDDD brownout = 1.30 V, CPUCLK = 24 MHz, HCLK = 24 MHz, PLL off, USB off, Application = MP3 Play, minimum power configuration selected.		45		mW
V_{iH} (DIO3)—Input high voltage for DIO3 digital I/O pin set in 3.3-V mode.	2.0			V
V_{IL} (DIO3)—Input low voltage for DIO3 digital I/O pin set in 3.3-V mode			0.8	V
V_{OH} (DIO3)—Output high voltage for DIO3 digital I/O pin set in 3.3-V mode, 4-mA mode	0.7*VDDIO			V
V_{OH} (DIO3)—Output high voltage for DIO3 digital I/O pin set in 3.3-V mode, 8-mA mode	0.7*VDDIO			V
V_{OL} (DIO3)—Output low voltage for DIO3 digital I/O pin set in 3.3-V mode.			0.4	V

3. ARM CPU COMPLEX

This chapter describes the ARM CPU included on the STMP36xx and includes sections on the processor core, the JTAG debugger, and the embedded trace macrocell (ETM) interface.

3.1. ARM 926 Processor Core

The on-chip Reduced Instruction Set Computer (RISC) processor core is an ARM, Ltd. 926EJ-S. This CPU implements the ARM v5TE instruction set architecture, which includes enhanced DSP instructions.

The ARM9EJ-S has two instruction sets: a 32-bit instruction set used in the ARM state and a 16-bit instruction set used in Thumb state. The core offers the choice of running in the ARM state or the Thumb state or a mix of the two. This enables optimization for both code density and performance. ARM studies indicate that Thumb code is typically 65% the size of equivalent ARM code, while providing 160% of the effective performance in constrained memory bandwidth applications.

A block diagram of the ARM926EJ-S core is shown in Figure 7.

See the following ARM documentation for more information on the ARM926EJ-S core (http://www.arm.com/documentation/ARMProcessor_Cores/index.html):

- ARM926EJ-S Technical Reference Manual, DDI0198D
- ARM926EJ-S Development Chip Reference Manual, DDI0287A

The ARM9 core has a total of 37 programmer-visible registers, including 31 general-purpose 32-bit registers, six 32-bit status registers, and a 32-bit program counter, as shown in Figure 8. In ARM state, 16 general-purpose registers and one or two status registers are accessible at any one time. In privileged modes, mode-specific banked registers become available.

The ARM state register set contains 16 directly addressable registers, r0 through r15. An additional register, the current program status register (CPSR), contains condition code flags and the current mode bits. Registers r0–r13 are general-purpose registers used to hold data and address values, with R13 being used as a stack pointer. R14 is used as the subroutine link register (lr) to hold the return address. Register r15 holds the program counter (PC).

The Thumb state register set is a subset of the ARM register set. The programmer has access to eight general-purpose registers, r0–r7, the PC (ARM r15), the stack pointer (ARM r13), the link register (ARM r14), and the cpsr.

Exceptions arise whenever the normal flow of program execution has to be temporarily suspended, for example, to service an interrupt from a peripheral. Before attempting to handle an exception, the ARM core preserves the current processor state, so that the original program can resume when the handler is finished.

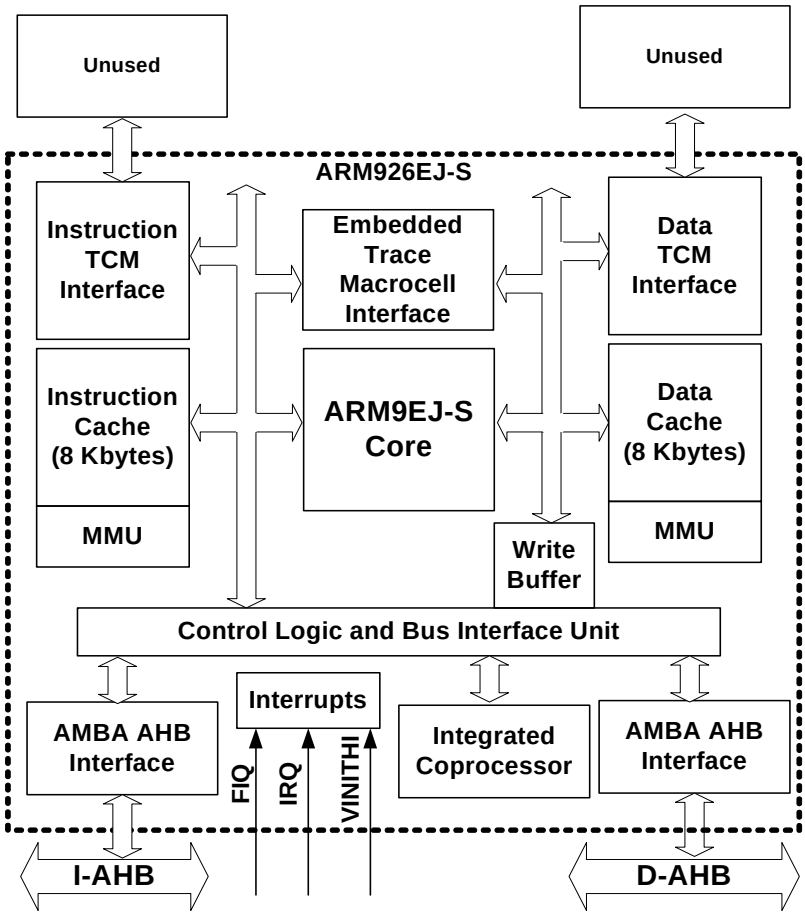


Figure 7. ARM926 RISC Processor Core

The following exceptions are recognized by the core:

- SWI—Software interrupt
- UNDEF—Undefined instruction
- PABT—Instruction prefetch abort
- FIQ—Fast peripheral interrupt
- IRQ—Normal peripheral interrupt
- DABT—Data abort
- RESET—Reset
- BKPT—Breakpoint

The vector table pointing to these interrupts can be located at physical address 0x00000000 or 0xFFFF0000. The STMP36xx maps its 64-Kbyte on-chip ROM to the address 0xFFFF0000 to 0xFFFFFFFF. The core is hardwired to use the high address vector table at hard reset (core port VINITHI =1).

The ARM 926 core includes an 8-Kbyte instruction cache and 8-Kbyte data cache and has two master interfaces to the AMBA AHB, as shown in Figure 7.

The STMP36xx always operates in little endian mode.

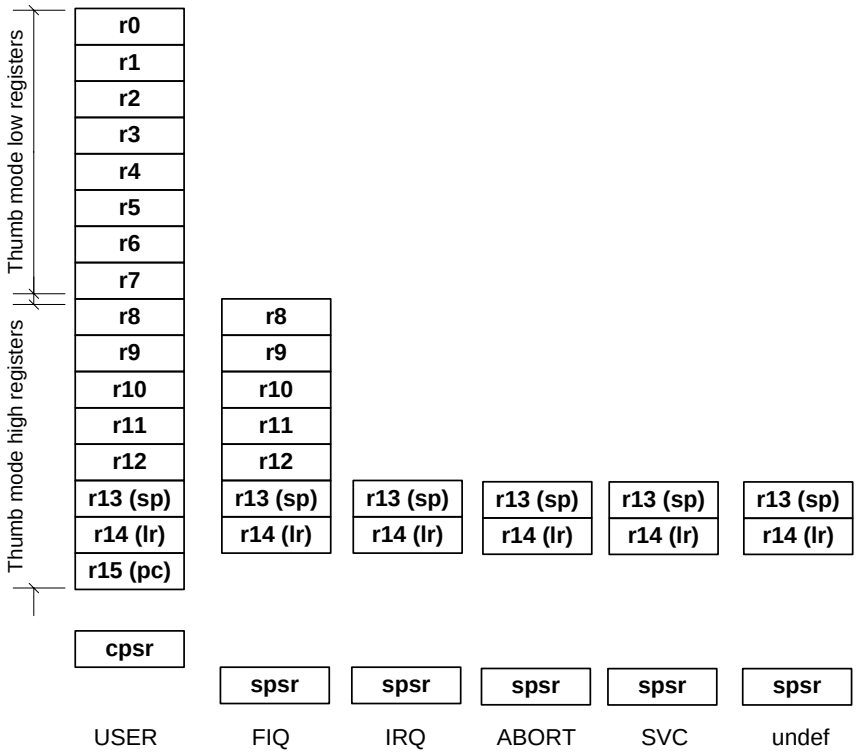


Figure 8. ARM Programmable Registers

3.2. JTAG Debugger

The TAP controller of the ARM core in the STMP36xx performs the standard debugger instructions.

3.2.1. JTAG READ ID

The TAP controller returns the following 32-bit data value in response to a JTAG READ ID instruction: 0x0792_64F3

3.2.2. JTAG Hardware Reset

The JTAG reset instruction can be accomplished by writing 0xDEADC0DE to ETM address 0x70. The ETM is on scan chain 6. The bit stream is 0xF0DEADC0DE.

The digital wide reset does not affect the DC-DC converters or the contents of the persistent registers in the analog side of the RTC.

3.2.3. JTAG Interaction with CPUCLK

Because the JTAG clock is sampled from the processor clock CPUCLK, there are cases in which the behavior of CPUCLK affects the ability to make use of JTAG. Specifically, the JTAG block will not function as expected if:

- CPUCLK is stalled due to an interrupt
- CPUCLK is less than 3x the JTAG clock
- CPUCLK is disabled for any reason

3.3. Embedded Trace Macrocell (ETM) Interface

The STMP36xx includes an ARM ETM-9 trace module implementing a medium mode trace buffer. See the pin list in Chapter 35 for the pinout of trace information.

4. CLOCK GENERATION AND CONTROL

This chapter describes the clock generation and control features of the STMP36xx and includes sections on the crystal oscillators, clock domains, low-power operation, clock dividers, PLL, USB PHY initialization, and clocking behavior during reset. Figure 9 shows a comprehensive view of all clocks included on the STMP36xx and their relationships to each other. The programmable registers are described in Section 4.9.

4.1. Overview

The STMP36xx clock architecture is designed to offer high performance, low power, and efficient software power management. The STMP36xx has up to three clock sources (two crystals and a Phase-Locked Loop (PLL)) that are distributed to twenty-four clock domains. Many of the clock domains have variable frequency and gating to minimize power consumption. The high-speed bus clock, used by many of the peripherals, has an automatic slow-down mode to reduce power while maintaining high performance.

4.2. Crystal Oscillators

The STMP36xx integrates two crystal oscillators. A 24-MHz crystal is mandatory and provides the clock source for the PLL and the main digital blocks. The 32.768-kHz crystal oscillator is available in the 169BGA package and can optionally be used as a clock reference for the real-time clock (RTC). The 32.768-kHz oscillator is used only to provide a low power, accurate reference for the RTC and is not used for any other functions.

The crystal oscillators have several configurable parameters, including:

- Crystal on or off when the STMP36xx is powered off.
- Real-time clock circuit can use either crystal
- Bias current adjustment
- Extra load capacitor (used to adjust frequency error)

The crystal configuration registers are persistent through the normal digital reset and are located with all the other persistent control bits in the real-time clock block.

4.3. Clock Domains

To offer the best combination of performance, power consumption, and ease of use, the STMP36xx has 25 clock domains, which are listed Table 9.

Table 9. Clock Domains

CLK DOMAIN	USED BY	SPEED RANGE	COMMENTS
XTAL_CLK24M	Most Clock Generators	24 MHz	The root clock for most of the chip. It is converted into other clocks through dividers and muxes.
CPUCLK	ARM CPU	0.1–200 MHz	Divided from PLL or XTAL_CLK24M. Always an integer multiple of HCLK.
HCLK	Main and HBUS Peripherals	0.1–100 MHz	Divided from CPUCLK. Always an integer multiple of EMICLK.
XCLK	XBUS Peripherals	0.1–24 MHz	Lower speed peripherals, divided from XTAL_CLK24M.
ANA_CLK24M	DC-DC, DAC, ADC	24 MHz	Low Jitter Analog Clock, sourced from XTAL_CLK24M.
DIGCTRL_CLK1M	DIGCTL 1-µs timer	1 MHz	Gated and divided clock sourced from XTAL_CLK24M.
DRI_CLK24M	Digital Radio Interface	24 MHz	Gated clock sourced from XTAL_CLK24M.
EMICLK	EMI	HCLK/n	Divided from HCLK.
EXRAM_CLK16K	SDRAM Controller	16 kHz	Gated and divided clock sourced from XTAL_CLK24M.
FILT_CLK24M	DAC/ADC Filters	24 MHz	Gated clock sourced from XTAL_CLK24M.
GPMICLK	GPMI	16–120 MHz	Allows constant bus speed while HCLK varies. Sourced from the PLL.
IRCLK	IR	2400 Hz–24 MHz	Sourced from IROVCLK.
IROVCLK	IR	1.8432–120 MHz	Source from the PLL.
LRADC_CLK2K	LRADC	2 kHz	Gated and divided clock sourced from XTAL_CLK24M.
OCRAM_CLK	Main On-Chip RAM	0–24 MHz (32 kHz typ.)	Gated and divided clock sourced from XTAL_CLK24M.
PCM_SPDIFCLK	SPDIF	4.096–6.144 MHz	Fractional divider from SPDIFCLK.
RTC_CLK32K	RTC	32.768 kHz or 32 kHz	Clock tree for RTC_ANA, sourced from either XTAL_CLK24M or XTAL_CLK32K.
SCLK	SSP	20–120 MHz	Allows constant serial clock while HCLK varies. Sourced from the PLL.
SPDIFCLK	SPDIF	120 MHz	Sourced from the PLL.
TIMROT_CLK32k	Timers/Rotary Decoder	32 kHz	Gated and divided clock sourced from XTAL_CLK24M.
UART_CLK	UART	0.1–24 MHz	Gated clock sourced from XCLK.
UTMI_CLK120M	USB	120 MHz	Clock for USB PHY-to-controller interface.
UTMI_CLK30M	USB	30 MHz	Clock for USB PHY-to-controller interface.
UTMI_CLK480M	UTMI	480 MHz	Sourced from the PLL.
XTAL_CLK32K	RTC	32.768 kHz	Low-power source for real-time clock.

During reset, most clock domains are connected together and are sourced from XTAL_CLK24M, divided down to 6 MHz. Exceptions include the DC-DC converter and real-time clock, which are always clocked by their nominal sources, even in reset. Eight cycles after reset is deasserted, each clock domain is switched to its

default source. The PLL is bypassed by default, so clock domains that are sourced from it are clocked at the 24-MHz crystal rate. Some clock domains are gated by default, including FILT_CLK24M, SCLK, GPMICLK, and SPDIFCLK.

STMP36xx clocks having restricted relationships with each other are listed in Table 10. Any clock relationship not listed in the table is not restricted.

Table 10. Restricted Clock Relationships

RELATED CLOCKS	RATIO	
	MIN	MAX
HCLK/SCLK	1/4	4/1
HCLK/GPMI	1/4	4/1
CPUCLK/JTAG_TCK	12/1	none

Note: Any clock relationship not listed is not restricted.

4.4. Power Saving Features of the Clock Architecture

The STMP36xx clocking system is designed for low-power operation. Some of the low-power features include:

- Multiple clock domains allow lower performance peripherals to operate at lower clock rates.
- Most digital blocks include clock gating options to reduce power consumption when they are not used.
- Dynamic clock adjustment—Most clock domains have adjustable dividers. In most modes, the PLL speed can be adjusted from 240 MHz to 480 MHz in 4-MHz steps. This ensures that each part of the system can always run very close to the minimum frequency needed for the application.

4.5. Clock Dividers

Most of the clock domains have integer dividers. The dividers are designed to switch frequency within three of the slower clocks's periods. For example, when switching the CPUCLK divider from (480 MHz/8) to (480 MHz/32), it could take up to 200 ns before the switch is complete.

4.5.1. Automatic HCLK Divider

To save power on the very large HCLK domain, an automatic HCLK divider can be used. The divider automatically adjusts HCLK from a nominal "fast" rate to a low power "slow" rate when the CPU or other high-bandwidth users are not requesting the bus. The ratio of fast to slow rates is programmable to 2:1, 4:1 and 8:1. HCLK can switch from slow to fast within 1 "fast" clock cycle. The HCLK register has several options to select which criteria are used to put HCLK into the fast mode.

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

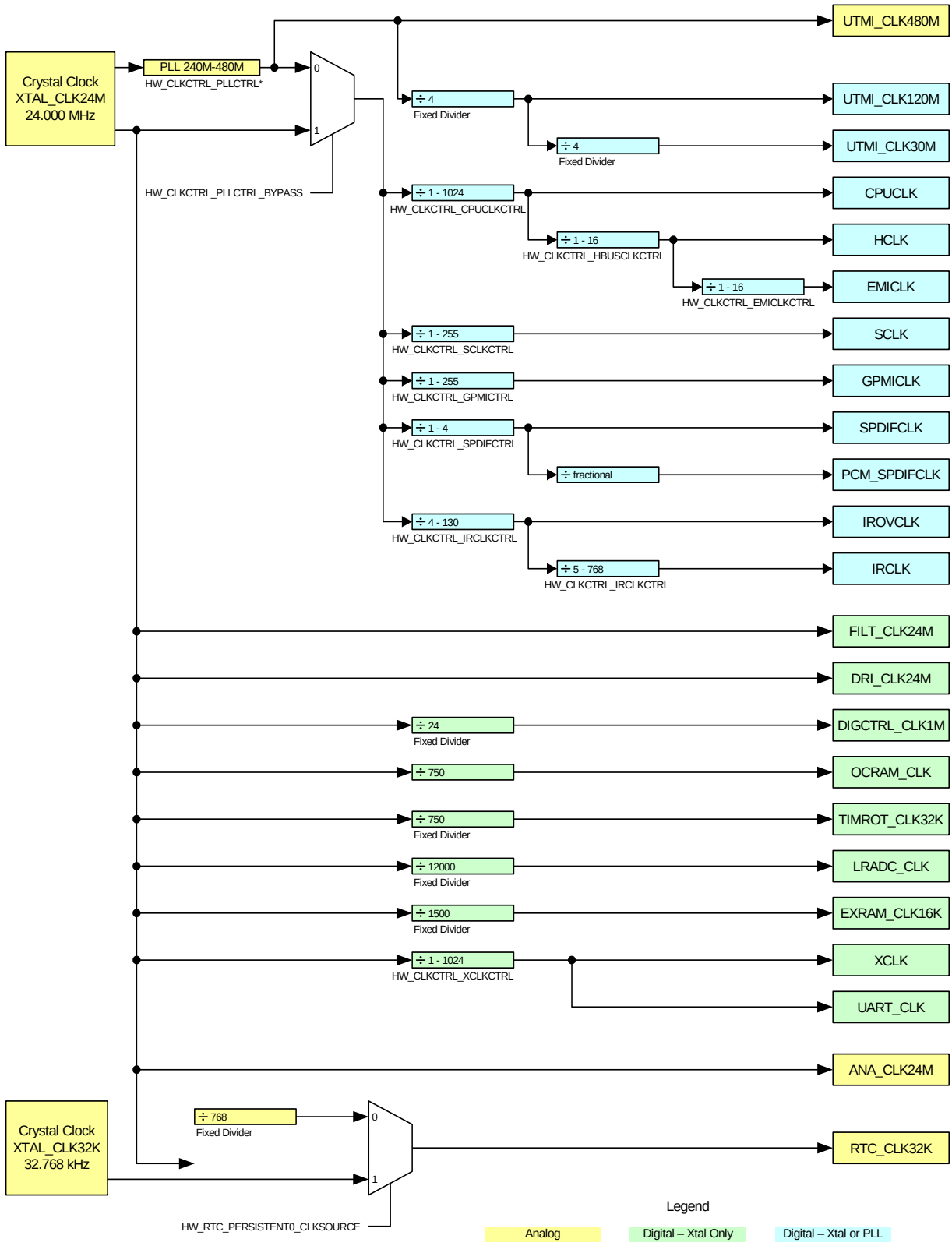


Figure 9. STMP36xx Clock Tree

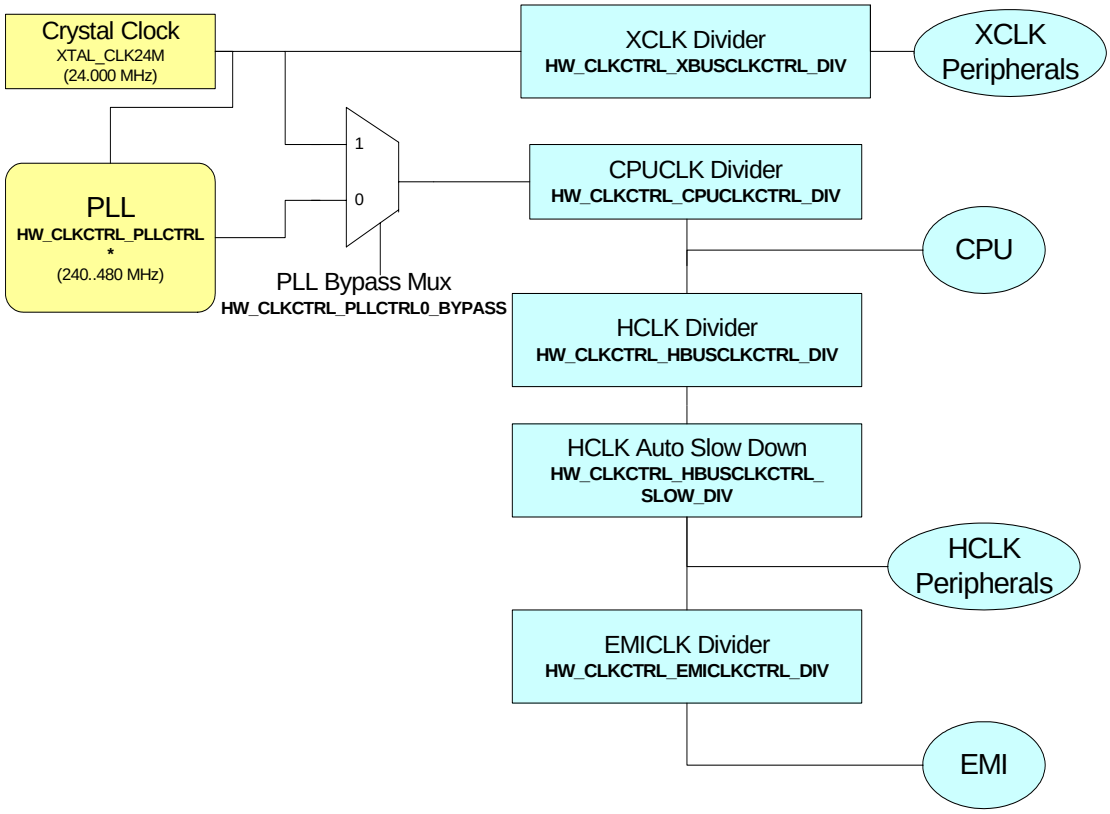


Figure 10. Detail Diagram of Key Clocks

4.6. Phase-Locked Loop (PLL)

The STMP36xx includes a 480-MHz PLL to clock the high-speed transceiver. This PLL can also be used for generating the system-wide digital clock. Figure 11 shows a block diagram of the PLL.

The STMP36xx PLL is programmable to generate a 240 to 480 MHz clock in 4-MHz steps. The PLL clock is used at high frequency by the USB. Other digital clock domains divide the PLL clock to lower frequencies. The PLL is designed for low power, low jitter, and high frequency switching speed. The frequency change time has been minimized to make dynamic clock adjustment more flexible to save power.

Most of the clock domains that are sourced from the PLL use dividers to decrease the frequency to the desired range. Typically, the PLL and dividers are set to generate a desired CPU frequency. The dividers can change very quickly, typically in just a few clock cycles. However, the PLL requires up to 10 µs to change frequency. During that time, the output of the clock divider is at a different frequency than was desired. It is typically acceptable for the frequency to be below the target for such a short period of time, but it is often unacceptable for the frequency to be above the target for even one clock cycle.

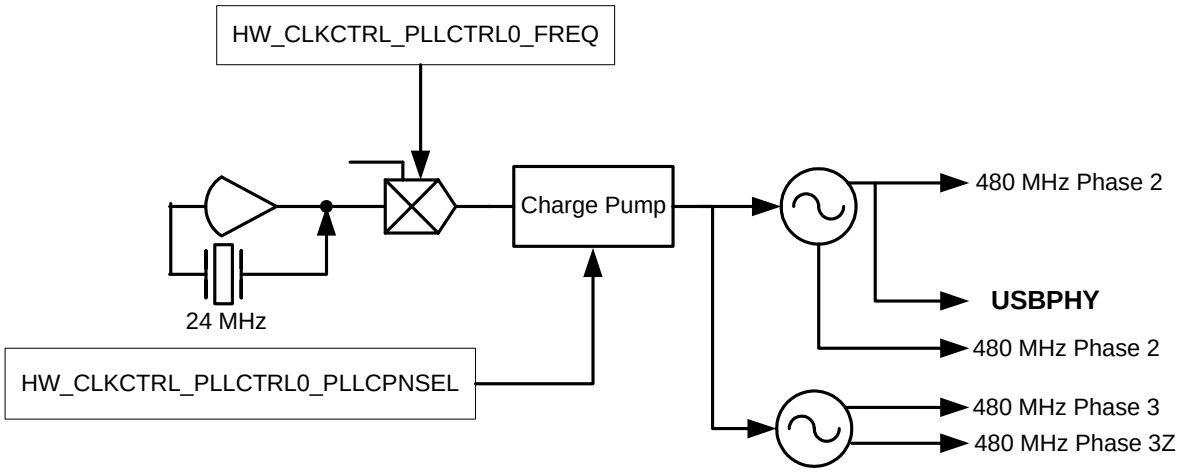


Figure 11. PLL Block Diagram

To ensure that the clocks are never faster than desired, the system must control the order of PLL and frequency divider adjustments. If the divider ratio is increasing, then the divider adjustment must be made before the PLL frequency change is complete. If the divide ratio is decreasing, then the divider adjustment is made after the PLL adjustment is complete.

The PLL has a lock indicator bit that is set when a frequency adjustment is completed. That lock bit can be used by software. However, it is not necessary for software to wait for the PLL lock bit to request a divider adjustment. The dividers can be programmed to wait for PLL lock before adjusting their settings. This feature allows software to adjust the PLL and all resulting frequencies, without waiting for the PLL to finish its adjustment. Note that the wait for PLL lock bit is edge-sensitive and takes effect only when the PLL lock transitions from 0 to 1 (i.e., if the PLL is already locked, then the wait for lock does not trigger).

Note: The PLL is not capable of operating to its maximum frequency at low operating voltages. Refer to Chapter 2, “Characteristics and Specifications” on page 39, for detailed information about the relationship between maximum PLL operating frequency and VDD voltage.

4.6.1. Frequency Program

The PLL can be programmed from 240 MHz to 480 MHz. The system uses many divided-down frequencies controlled by the CLKCTRL registers. The HW_CLKCTRL_PLLCTRL0.FREQ sets the frequency in 4-MHz increments.

4.6.2. PLL Use in USB and SPDIF Modes

To ensure proper operation and conform to industry standards, the PLL must operate at a fixed 480-MHz when USB is active. It must be a multiple of 120 MHz when SPDIF is active. The clocks that are sourced by the PLL can be adjusted using their dividers. For example, when the PLL is operating at 480 MHz, the CPU can operate at 160, 120, 96, 80, 68.5 MHz, etc.

To use the on-chip USB PHY, software must set the ENABLE_USB_CLK bit in the PLL register. This function requires extra power, so it should only be used when the

USB PHY is powered up. See Figure 12 and Figure 13 for additional detail about PHY initialization and suspend.

4.6.3. VCO and Phase Followers

The heart of the PLL is the Variable Crystal Oscillator (VCO), which can operate from 240 MHz to 480 MHz. The VCO frequency is determined by the output of the charge pump, in standard fashion. The VCO produces a 480-MHz clock for USB application and its exact out-of-phase component. In the design, these are identified as vco_clk2 and vco_clk2z. In addition, three phase followers are included to produce a precise eight-phase clock at 480 MHz. These eight phases are used in the high-speed digital receiver to operate the PLL that tracks the incoming 480Mbit/s USB receive digital stream. The vco_clk2 clock is also used as a single phase 480-MHz digital clock for various clock dividers and other circuits within the CLKCTRL. The VCO and various of its phase followers can be selectively powered down to reduce the overall energy requirements of the STMP36xx.

Note: In non-USB mode, all clocks except vco_clk2 can be gated off by HW_CLKCTRL_PLLCTRL0.EN_USB_CLKS to save power.

4.6.4. PFD and Charge Pump

The phase/frequency detector (PFD) and charge pump (CP) are used to lock the VCO to the reference oscillator. For the STMP36xx, the reference is the integrated crystal oscillator. The most common reference crystal frequencies are 24 MHz and 20 MHz. Selective power down and control of the PFD, the CP, and various loop filter parameters can be controlled in HW_CLKCTRL_PLLCTRL0**/1**. The charge pump gain (current) should be adjusted for different feedback settings; see HW_CLKCTRL0_PLLCPNSEL.

4.7. Integrated USB 2.0 PHY Initialization Flow Charts

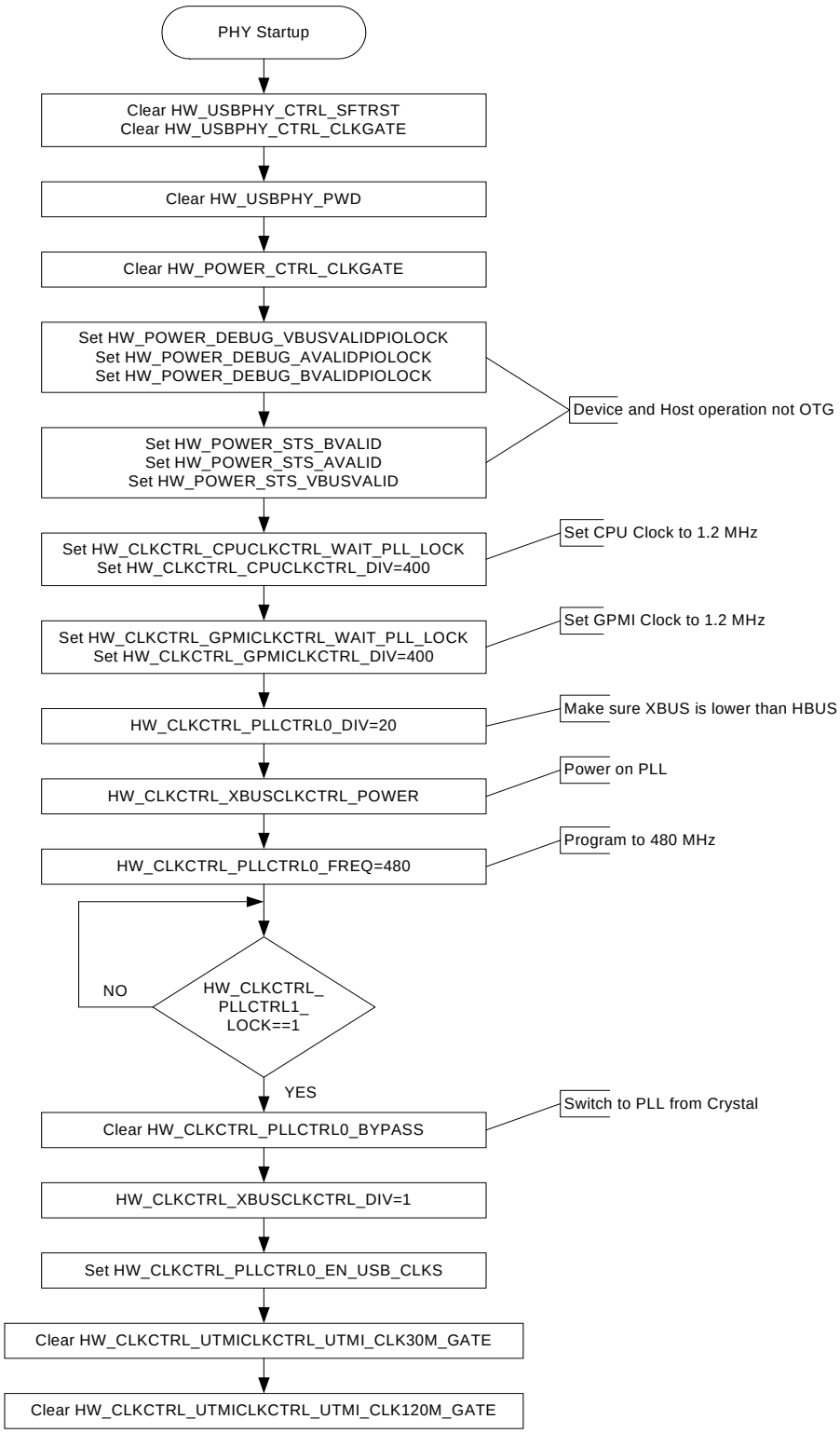


Figure 12. USB 2.0 PHY Startup Flowchart

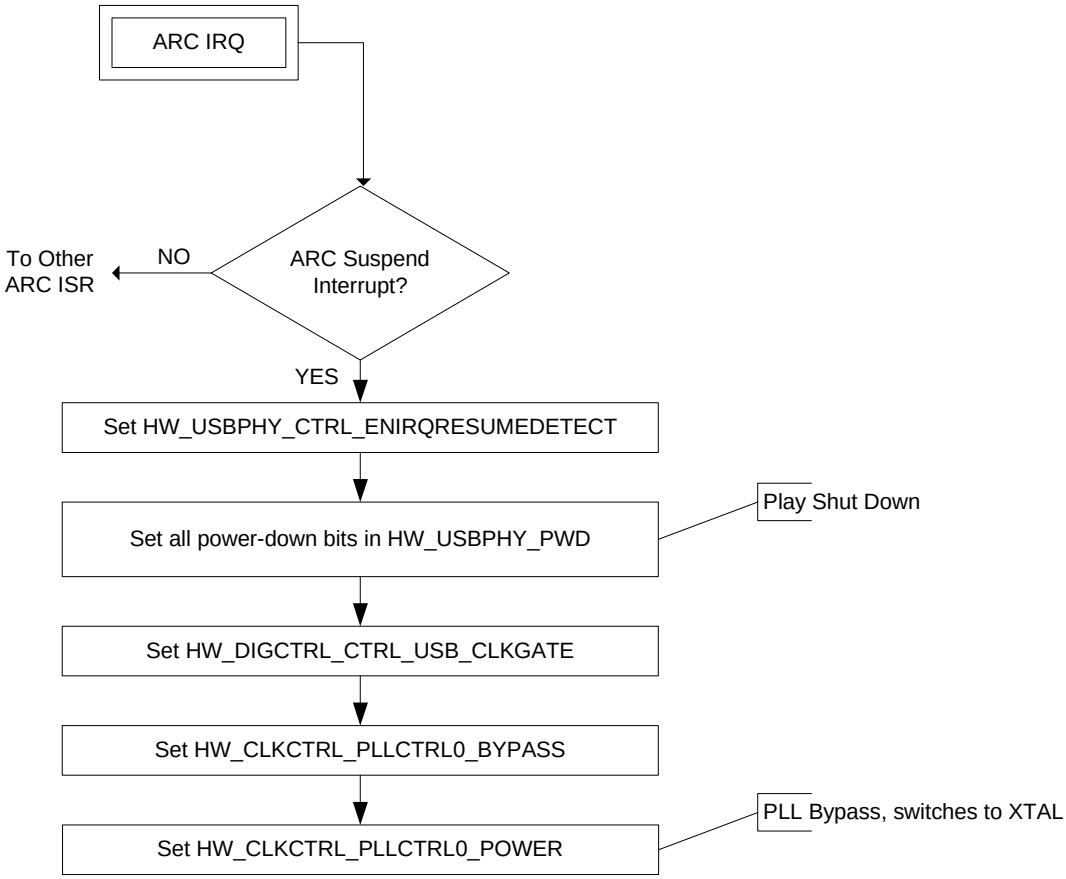


Figure 13. USB 2.0 PHY PLL Suspend Flowchart

4.8. Clocking During Reset

While the digital reset is asserted, all digital clock domains are connected to a 6-MHz clock based on XTAL_CLK24M. In this mode, the clock trees are not balanced, but the low 6-MHz rate ensures that timing is met. The reset is allowed to propagate for sixteen 6-MHz clocks. Eight 6-MHz clocks after reset is released, all clock domains revert to their defaults.

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

4.9. Programmable Registers

The following programmable registers are available to software for controlling and using the clock generation and control features of the STMP36xx.

4.9.1. PLL Control Register 0 Description

The PLL Control Register 0 programs the divide factor and sets VCO and V2I.

HW_CLKCTRL_PLLCTRL0 0x80040000
HW_CLKCTRL_PLLCTRL0_SET 0x80040004
HW_CLKCTRL_PLLCTRL0_CLR 0x80040008
HW_CLKCTRL_PLLCTRL0_TOG 0x8004000C

Table 11. HW_CLKCTRL_PLLCTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD5	PLLVCOKSTART	PLLCPSHORTLFR	PLLCPDBLIP	RSRVD4	PLLCPNSEL			RSRVD3		PLLV2ISEL		FORCE_FREQ	EN_USB_CLKS	BYPASS	POWER	RSRVD1								FREQ							

Table 12. HW_CLKCTRL_PLLCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD5**	RO	0x0	Always set to zero.
30	**PLLVCOKSTART**	RW	0x0	TEST MODE FOR SIGMATEL USE ONLY. This test bit is provided for the unlikely event that the VCO does not start oscillation. This is theoretically possible, but highly unlikely and can only happen in a noiseless system. Normally set to zero. To kick-start the VCO, perform a zero-to-one transition on this bit followed by a one-to-zero transition.
29	**PLLCPSHORTLFR**	RW	0x0	TEST MODE FOR SIGMATEL USE ONLY. This normally low test mode bit is used to short the charge pump resistor for a highly under-damped response. Set to one to short the resistor. The resistor should only be shorted in test mode.
28	**PLLCPDBLIP**	RW	0x0	TEST MODE FOR SIGMATEL USE ONLY. Set to one to double the charge pump current to speed up lock time. It can be used in conjunction with the PLLCPNSEL field to change the loop performance. At start-up time, it can be set to one to shorten the lock time. During normal operation, this should be set to zero for the lowest overall tracking jitter.
27	**RSRVD4**	RO	0x0	Always set to zero.

Table 12. HW_CLKCTRL_PLLCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26:24	**PLLCPNSEL**	RW	0x0	TEST MODE FOR SIGMATEL USE ONLY. These bits are set in conjunction with PLLCPDBLIP to maintain a constant loop-filter damping factor for the different divide ratios. They can also be used independently to speed up or slow down the activity of the PLL. DEFAULT = 0x0 Default ip current TIMES_15 = 0x2 ip current * 1.5 TIMES_075 = 0x3 ip current * 0.75 TIMES_05 = 0x4 ip current * 0.5 TIMES_04 = 0x7 ip current * 0.4
23:22	**RSRVD3**	RO	0x0	Always set to zero.
21:20	**PLLV2ISEL**	RW	0x0	These bits can be used to extend the frequency range of PLL. PLLV2ISEL transitions from 0x2 to 0x0 are recommended only when also changing FREQ from 240 to 480 MHz. Similarly, PLLV2ISEL transitions from 0x0 to 0x2 are recommended only when changing FREQ from 480 to 240 MHz. NORMAL = 0x0 Normal Range LOWER = 0x1 Lower the useful frequency range LOWEST = 0x2 Lowest useful frequency range. HIGHEST = 0x3 Highest useful frequency range
19	**FORCE_FREQ**	RW	0x0	Set this bit to one to force a write to this register to push a repeated value in the FREQ bit field out to the PLL. This allows firmware to bypass the logic that looks for a write to FREQ bit to be writing a different value than is already there. FORCE_SAME_FREQ = 0x1 force the value in the FREQ field out to the PLL, even if one is overwriting exactly the same value. HONOR_SAME_FREQ_RULE = 0x0 Honor the rule that says the FREQ field value must be over writen with a different value to force the value in the FREQ field out to the PLL.
18	**EN_USB_CLKS**	RW	0x0	0: 8-phase PLL outputs for USB PHY are powered down. If set to 1, 8-phase PLL outputs for USB PHY are powered up. The PLL must also be set to 480 MHz for USB operation. Additionally, the UTMICLK120_GATE and UTMICLK30_GATE must be deasserted to enable USB operation.
17	**BYPASS**	RW	0x1	If set to 1, PLL is bypassed and PLLCLK is 24 MHz. 0: PLLCLK is sourced from the PLL. PLL must be powered up before this bit is set.
16	**POWER**	RW	0x0	PLL Power On(0= PLL off; 1=PLL On). Allow 1 ms after turning the PLL on before enabling the PLL.
15:9	**RSRVD1**	RO	0x0	Always set to zero.
8:0	**FREQ**	RW	0x1E0	PLL output frequency in MHz. The PLL has 4-MHz steps (bits 0 and 1 are ignored).

DESCRIPTION:

The PLL Control Register 0 programs the divide factor and sets VCO and V2I. Do NOT turn off the bypass bit until the PLL has completed a lock cycle.

The PLL generates clocks from 240 MHz to 480 MHz on the PLLCLK net. The PLLCLK net can be driven either from the PLL or from the 24.0-MHz crystal oscilla-

tor. When BYPASS is set to zero, the PLLCLK net is driven by the PLL. When BYPASS is set to one, then PLLCLK is driven from the 24.0-MHz crystal oscillator.

EXAMPLE:

```
HW_CLKCTRL_PLLCTRL0_WR(BF_CLKCTRL_PLLCTRL0_FREQ(480)); // set to 480 MHz
HW_CLKCTRL_PLLCTRL0_WR(BF_CLKCTRL_PLLCTRL0_BYPASS(0)); // final enable of PLL
```

4.9.2. PLL Control Register 1 Description

The PLL Control Register 1 specifies the lock count to use for PLL stabilization.

HW_CLKCTRL_PLLCTRL1 0x80040010
HW_CLKCTRL_PLLCTRL1_SET 0x80040014
HW_CLKCTRL_PLLCTRL1_CLR 0x80040018
HW_CLKCTRL_PLLCTRL1_TOG 0x8004001C

Table 13. HW_CLKCTRL_PLLCTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
LOCK	FORCE_LOCK	RSRVD1														LOCK_COUNT															

Table 14. HW_CLKCTRL_PLLCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**LOCK**	RO	0x0	0: PLL not locked. If set to 1, PLL is locked. The PLL lock timer should be set according to changes to the FREQ or POWER registers. The timer count depends on the change.
30	**FORCE_LOCK**	RW	0x0	Set to one to start another PLL lock cycle. Note: To start another lock cycle, this must be cleared first.
29:16	**RSRVD1**	RO	0x0	Always set to zero.
15:0	**LOCK_COUNT**	RO	0x0	Reflects the number of cycles required by the PLL to lock to the newly programmed frequency. Default, power-on lock-time is 8192-cycles. Counts up from zero until the value from the internal table is reached.

DESCRIPTION:

Use this register to control the various aspects of PLL lock management.

EXAMPLE:

```
HW_CLKCTRL_PLLCTRL1_WR(BF_CLKCTRL_PLLCTRL1_LOCK_COUNT(9000));
```

4.9.3. CPU Clock Control Register Description

The CPUCLK Clock Control Register provides controls for generating the ARM CPUCLK.

HW_CLKCTRL_CPUCLKCTRL 0x80040020

Table 15. HW_CLKCTRL_CPUCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3	WAIT_PLL_LOCK	BUSY	RSRVD2																INTERRUPT_WAIT	RSRVD1		DIV									

Table 16. HW_CLKCTRL_CPUCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD3**	RO	0x0	Always set to zero.
30	**WAIT_PLL_LOCK**	RW	0x0	Wait for PLL Lock. If this is set, then new data written to the DIV field will not take effect until the PLL lock bit is set.
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28:13	**RSRVD2**	RO	0x0	Always set to zero.
12	**INTERRUPT_WAIT**	RW	0x0	Enables the gating of CPUCLK when used in conjunction with the wait for interrupt instruction (MCR).
11:10	**RSRVD1**	RO	0x0	Always set to zero.
9:0	**DIV**	RW	0x001	This field controls the CPUCLK divide ratio. CPUCLK is generated from PLLCLK through this divider. This is an integer divider. Values between 1 and 1023 are valid. Changes to the CPUCLK frequency will also affect HCLK. NOTE: PLLCLK is either sourced from the PLL at frequencies between 240 MHz and 480 MHz or from the crystal oscillator at 24.0 MHz. The divider is set to divide by 1 at power-on reset.

DESCRIPTION:

Controls for the ARM 926 clock divider.

EXAMPLE:

```
HW_CLKCTRL_CPUCLKCTRL_WR(BF_CLKCTRL_CPUCLKCTRL_DIV(12)); // 480 MHz/12 = 40 MHz
```

4.9.4. AHB, APBH Bus Clock Control Register Description

The AHB, APBH Bus Clock Control Register provides controls for HCLK generation.

HW_CLKCTRL_HBUSCLKCTRL 0x80040030

Table 17. HW_CLKCTRL_HBUSCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4	WAIT_PLL_LOCK	BUSY	RSRVD3	EMI_BUSY_FAST	APBHDMA_BUSY_FAST	APBXDMA_BUSY_FAST	TRAFFIC_JAM_FAST	TRAFFIC_FAST	CPU_DATA_FAST	CPU_INSTR_FAST	AUTO_SLOW_MODE	RSRVD2		SLOW_DIV		RSRVD1											DIV				

Table 18. HW_CLKCTRL_HBUSCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD4**	RO	0x0	Reserved
30	**WAIT_PLL_LOCK**	RW	0x0	Wait for PLL Lock. If this is set, then new data written to the DIV field will not take effect until the PLL lock bit has transitioned from cleared to set. Note that this function is edge-sensitive.
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28	**RSRVD3**	RO	0x0	Reserved
27	**EMI_BUSY_FAST**	RW	0x0	From auto-slow mode, switch to fast mode when the External Memory Interface is busy.
26	**APBHDMA_BUSY_FAST**	RW	0x0	From auto-slow mode, switch to fast mode when the APBH DMA has pending activity.
25	**APBXDMA_BUSY_FAST**	RW	0x0	From auto-slow mode, switch to fast mode when the APBX DMA has pending activity.
24	**TRAFFIC_JAM_FAST**	RW	0x0	From auto-slow mode, switch to fast mode when three or more masters are trying to use the AHB.
23	**TRAFFIC_FAST**	RW	0x0	From auto-slow mode, switch to fast mode when any master accesses the AHB.
22	**CPU_DATA_FAST**	RW	0x0	From auto-slow mode, switch to fast mode with CPU Data access to AHB.
21	**CPU_INSTR_FAST**	RW	0x0	From auto-slow mode, switch to fast mode with CPU Instruction access to AHB.
20	**AUTO_SLOW_MODE**	RW	0x0	Enable HCLK auto-slow mode. When this is set, then HCLK will run at the slow rate until one of the fast mode events has occurred.
19:18	**RSRVD2**	RO	0x0	Reserved

Table 18. HW_CLKCTRL_HBUSCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:16	**SLOW_DIV**	RW	0x0	Slow mode divide ratio. Sets the ratio of HCLK fast rate to the slow rate. 00=1, 01=2, 10 =4, 11=8 BY1 = 0x0 Slow mode divide ratio = 1 BY2 = 0x1 Slow mode divide ratio = 2 BY4 = 0x2 Slow mode divide ratio = 1 BY8 = 0x3 Slow mode divide ratio = 1
15:5	**RSRVD1**	RO	0x0	Reserved
4:0	**DIV**	RW	0x01	CPUCLK-to-HCLK divide ratio. HCLK is sourced from PLLCLK but is related to CPUCLK by the value in this divider. The HCLK frequency is dependent on the PLL, CPUCLKDIV and HCLKDIV. This is an integer divider. Values between 1 and 16 are valid. NOTE: PLLCLK is either sourced from the PLL at frequencies between 240 MHz and 480 MHz or from the crystal oscillator at 24.0 MHz. The divider is set to divide by 1 at power-on reset.

DESCRIPTION:

This register controls the clock divider that generates the HCLK, the clock used by the AHB and APBH buses.

EXAMPLE:

```
HW_CLKCTRL_HBUSCLKCTRL_WR(BF_CLKCTRL_HBUSCLKCTRL_DIV(2)); // set hclk to half the ARM clock frequency
```

4.9.5. APBX Clock Control Register Description

The APBX Clock Control Register provides control of the XCLK clock divider.

HW_CLKCTRL_XBUSCLKCTRL 0x80040040

Table 19. HW_CLKCTRL_XBUSCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BUSY	RSRVD1																					DIV									

Table 20. HW_CLKCTRL_XBUSCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
30:10	**RSRVD1**	RO	0x0	Always set to zero.
9:0	**DIV**	RW	0x001	This field controls the XCLK divide ratio. XCLK is sourced from the 24-MHz XTAL through this divider. This is an integer divider. Values between 1 and 1023 are valid.

DESCRIPTION:

This register controls the clock divider that generates the XCLK, the clock used by the APBX bus.

EXAMPLE:

```
HW_CLKCTRL_XBUSCLKCTRL_WR(BF_CLKCTRL_XBUSCLKCTRL_DIV(4)); // set APBX XBUS clock to 1/4
the 24.0MHz crystal clock frequency
```

4.9.6. XTAL Clock Control Register Description

This XCLK control register provides control of various fixed dividers sourced from the 24-MHz XTAL clock domain.

HW_CLKCTRL_XTALCLKCTRL 0x80040050

Table 21. HW_CLKCTRL_XTALCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UART_CLK_GATE	FILT_CLK24M_GATE	PWM_CLK24M_GATE	DRI_CLK24M_GATE	DIGCTRL_CLK1M_GATE	TIMROT_CLK32K_GATE	EXRAM_CLK16K_GATE	LRADC_CLK2K_GATE	RSRVD1																							

Table 22. HW_CLKCTRL_XTALCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**UART_CLK_GATE**	RW	0x0	If set to 1, variable UART_CLK is gated off.
30	**FILT_CLK24M_GATE**	RW	0x1	If set to 1, fixed 24-MHz clock for the Digital Filter is gated off.
29	**PWM_CLK24M_GATE**	RW	0x0	If set to 1, fixed 24-MHz clock for the PWM is gated off.
28	**DRI_CLK24M_GATE**	RW	0x0	If set to 1, fixed 24-MHz clock for the Digital Radio Interface (DRI) is gated off.
27	**DIGCTRL_CLK1M_GATE**	RW	0x0	If set to 1, fixed 1-MHz clock for DIGCTRL is gated off.
26	**TIMROT_CLK32K_GATE**	RW	0x0	If set to 1, fixed 32-kHz clock for the TIMROT block is gated off.
25	**EXRAM_CLK16K_GATE**	RW	0x0	If set to 1, fixed 16-kHz clock for off-chip SRAM is gated off.
24	**LRADC_CLK2K_GATE**	RW	0x0	If set to 1, fixed 2-kHz clock for LRADC is gated off.
23:0	**RSRVD1**	RO	0x0	Always set to zero.

DESCRIPTION:

This register controls various fixed-rate dividers working off the 24.0-MHz crystal clock.

EXAMPLE:

```
HW_CLKCTRL_XTALCLKCTRL_WR(BF_CLKCTRL_XTALCLKCTRL_UART_CLK_GATE(1));
```

4.9.7. On-Chip SRAM Clock Control Register Description

This register is reserved for SigmaTel use and should not be written.

HW_CLKCTRL_OCRAMCLKCTRL 0x80040060

Table 23. HW_CLKCTRL_OCRAMCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	BUSY	RSRVD1																				DIV									

Table 24. HW_CLKCTRL_OCRAMCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x0	Reserved.
30	**BUSY**	RO	0x0	Reserved.
29:10	**RSRVD1**	RO	0x0	Reserved.
9:0	**DIV**	RW	0x2EE	Reserved.

4.9.8. UTMI Clock Control Register Description

HW_CLKCTRL_UTMICLKCTRL 0x80040070

Table 25. HW_CLKCTRL_UTMICLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UTMI_CLK120M_GATE	UTMI_CLK30M_GATE	RSRVD1																													

Table 26. HW_CLKCTRL_UTMICLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**UTMI_CLK120M_GATE**	RW	0x1	UTMICLK120 Gate. If set to 1, UTMICLK120M is gated off. 0: UTMICLK120M is not gated. UTMICLK120M is a 120-MHz fixed clock that is only valid when the PLL is running at 480 MHz.
30	**UTMI_CLK30M_GATE**	RW	0x1	UTMICLK30 Gate. If set to 1, UTMICLK30 is gated off. 0: UTMICLK30 is not gated. UTMICLK30 is a 30-MHz fixed clock that is only valid when the PLL is running at 480 MHz.
29:0	**RSRVD1**	RO	0x0	Always set to zero.

DESCRIPTION:

This register controls the clock gating for the integrated USB 2.0 PHY UTMI.

EXAMPLE:

```
HW_CLKCTRL_UTMICLKCTRL_WR(BF_CLKCTRL_UTMICLKCTRL_UTMI_CLK120M_GATE(0));
HW_CLKCTRL_UTMICLKCTRL_WR(BF_CLKCTRL_UTMICLKCTRL_UTMI_CLK30M_GATE(0));
```

4.9.9. Synchronous Serial Port Clock Control Register Description

HW_CLKCTRL_SSPCLKCTRL 0x80040080

Table 27. HW_CLKCTRL_SSPCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	WAIT_PLL_LOCK	BUSY	RSRVD1																				DIV								

Table 28. HW_CLKCTRL_SSPCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	SCLK Gate. If set to 1, SCLK is gated off. 0: SCLK is not gated
30	**WAIT_PLL_LOCK**	RW	0x0	Wait for PLL Lock. If this is set, then new data written to the DIV field will not take effect until the PLL lock bit is set.
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.

Table 28. HW_CLKCTRL_SSPCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
28:9	**RSRVD1**	RO	0x0	Always set to zero.
8:0	**DIV**	RW	0x1	The synchronous serial port clock frequency is determined by dividing the PLLCLK by the value in this bit field. NOTE: PLLCLK is either sourced from the PLL at frequencies between 240 MHz and 480 MHz or from the crystal oscillator at 24.0 MHz. The divider is set to divide by 1 at power-on reset.

DESCRIPTION:

This register controls the clock divider that generates the clock for the synchronous serial port (SSP).

EXAMPLE:

```
HW_CLKCTRL_SSPCLKCTRL_WR(BF_CLKCTRL_SSPCLKCTRL_DIV(40));
```

4.9.10. General-Purpose Media Interface Clock Control Register Description

HW_CLKCTRL_GPMICLKCTRL 0x80040090

Table 29. HW_CLKCTRL_GPMICLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	WAIT_PLL_LOCK	BUSY	RSRVD1																			DIV									

Table 30. HW_CLKCTRL_GPMICLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	GPMICLK Gate. If set to 1, GPMICLK is gated off. 0: GPMICLK is not gated
30	**WAIT_PLL_LOCK**	RW	0x0	Wait for PLL Lock. If this is set, then new data written to the DIV field will not take effect until the PLL lock bit is set.
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.

Table 30. HW_CLKCTRL_GPMICLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
28:10	**RSRVD1**	RO	0x0	Always set to zero.
9:0	**DIV**	RW	0x1	The GPMI clock frequency is determined by dividing the PLLCLK by the value in this bit field. NOTE: PLLCLK is either sourced from the PLL at frequencies between 240 MHz and 480 MHz or from the crystal oscillator at 24.0 MHz. The divider is set to divide by 1 at power-on reset.

DESCRIPTION:

This register controls the divider that generates the General-Purpose Media Interface (GPMI) clock.

EXAMPLE:

```
HW_CLKCTRL_GPMICLKCTRL_WR(BF_CLKCTRL_GPMICLKCTRL_DIV(40));
```

4.9.11. SPDIF Clock Control Register Description

HW_CLKCTRL_SPDIFCLKCTRL 0x800400a0

Table 31. HW_CLKCTRL_SPDIFCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	BUSY	RSRVD1																											DIV		

Table 32. HW_CLKCTRL_SPDIFCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	SPDIFCLK Gate. If set to 1, SPDIFCLK is gated off. 0: SPDIFCLK is not gated
30	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
29:3	**RSRVD1**	RO	0x0	Always set to zero.
2:0	**DIV**	RW	0x4	This field controls the SPDIFCLK divide ratio. SPDIFCLK is sourced from the PLL through this divider. This is an integer divider. Values between 1 and 4 are valid. To meet industry standards, the SPDIFCLK must always be set to 120 MHz, i.e., at PLL=480 MHz, SPDIFCLKDIV = 4.

DESCRIPTION:

This register controls the clock divider that generates the SPDIF clock.

EXAMPLE:

```
HW_CLKCTRL_SPDIFCLKCTRL_WR(BF_CLKCTRL_SPDIFCLKCTRL_DIV(4));
```

4.9.12. EMI Clock Control Register Description

HW_CLKCTRL_EMICLKCTRL 0x800400b0

Table 33. HW_CLKCTRL_EMICLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	WAIT_PLL_LOCK	BUSY	RSRVD1																										DIV		

Table 34. HW_CLKCTRL_EMICLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	EMICLK Gate. If set to 1, EMICLK is gated off. 0: EMICLK is not gated
30	**WAIT_PLL_LOCK**	RW	0x0	Wait for PLL Lock. If this is set, then new data written to the DIV field will not take effect until the PLL lock bit is set.
29	**BUSY**	RO	0x0	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28:3	**RSRVD1**	RO	0x0	Always set to zero.
2:0	**DIV**	RW	0x1	This field controls the EMICLK divide ratio. EMICLK is sourced from HBUSCLK through this divider. This is an integer divider. Values between 1 and 7 are valid. Note that this bit cannot be changed during a transfer, or else the data in the DRAM may be corrupted.

DESCRIPTION:

This register controls the clock divider that generates the External Memory Interface (EMI) clock.

EXAMPLE:

```
HW_CLKCTRL_EMICLKCTRL_WR(BF_CLKCTRL_EMICLKCTRL_DIV(1));
```

4.9.13. IR Clock Control Register Description

HW_CLKCTRL_IRCLKCTRL 0x800400c0

Table 35. HW_CLKCTRL_IRCLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	WAIT_PLL_LOCK	AUTO_DIV	IR_BUSY	IROV_BUSY	RSRVD2			IROV_DIV								RSRVD1						IR_DIV									

Table 36. HW_CLKCTRL_IRCLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	IRCLK Gate. If set to 1, IRCLK is gated off. 0: IRCLK is not gated
30	**WAIT_PLL_LOCK**	RW	0x0	Wait for PLL Lock. If this is set, then new data written to the DIV field will not take effect until the PLL lock bit is set.
29	**AUTO_DIV**	RW	0x1	Allow hardware to automatically set the divide ratios.
28	**IR_BUSY**	RO	0x0	This read-only bit field returns a one when the IR_DIV clock divider is busy transfering a new divider value across clock domains.
27	**IROV_BUSY**	RO	0x0	This read-only bit field returns a one when the IROV_DIV clock divider is busy transfering a new divider value across clock domains.
26:25	**RSRVD2**	RO	0x0	Always set to zero.
24:16	**IROV_DIV**	RW	0x4	This field controls the IRCLK Divide ratio-1. This is an integer divider. Values between 4 and 130 are valid. This divider is used in conjunction with IR_DIV to set the final rate of the IRCLK.
15:10	**RSRVD1**	RO	0x0	Always set to zero.
9:0	**IR_DIV**	RW	0x4	This field controls the IRCLK divide-ratio-2. This is an integer divider. Values between 5 and 768 are valid. This divider is used in conjunction with IROV_DIV to set the final rate of the IRCLK.

DESCRIPTION:

This register controls the generation of both the IR clock and the IR oversample clock.

EXAMPLE:

```
HW_CLKCTRL_IRCLKCTRL_WR(BF_CLKCTRL_IRCLKCTRL_IROV_DIV(4) |
BF_CLKCTRL_IRCLKCTRL_IR_DIV(4));
```

CLKCTRL XML Revision: 1.54

5. INTERRUPT COLLECTOR

This chapter describes the interrupt control features of the STMP36xx and includes sections on interrupt nesting, FIQ generation, and CPU wait-for-interrupt mode. Table 37 lists all of the interrupt sources available on the STMP36xx. Programmable registers for interrupt generation and control are described in Section 5.7.

5.1. Overview

The ARM926 CPU core has two interrupt input lines, IRQ and FIQ. As shown in Figure 14, the Interrupt Collector (ICOLL) steers 64 interrupt sources to the two interrupt input signals on the ARM core: IRQ and FIQ. Within an individual interrupt request line, the ICOLL offers four-level priority (above base level) for each of its interrupt sources. Preemption of a lower priority interrupt by a higher priority is supported (interrupt nesting). Interrupts assigned to the same level are serviced in a strict linear priority order within level from lowest to highest interrupt source bit number.

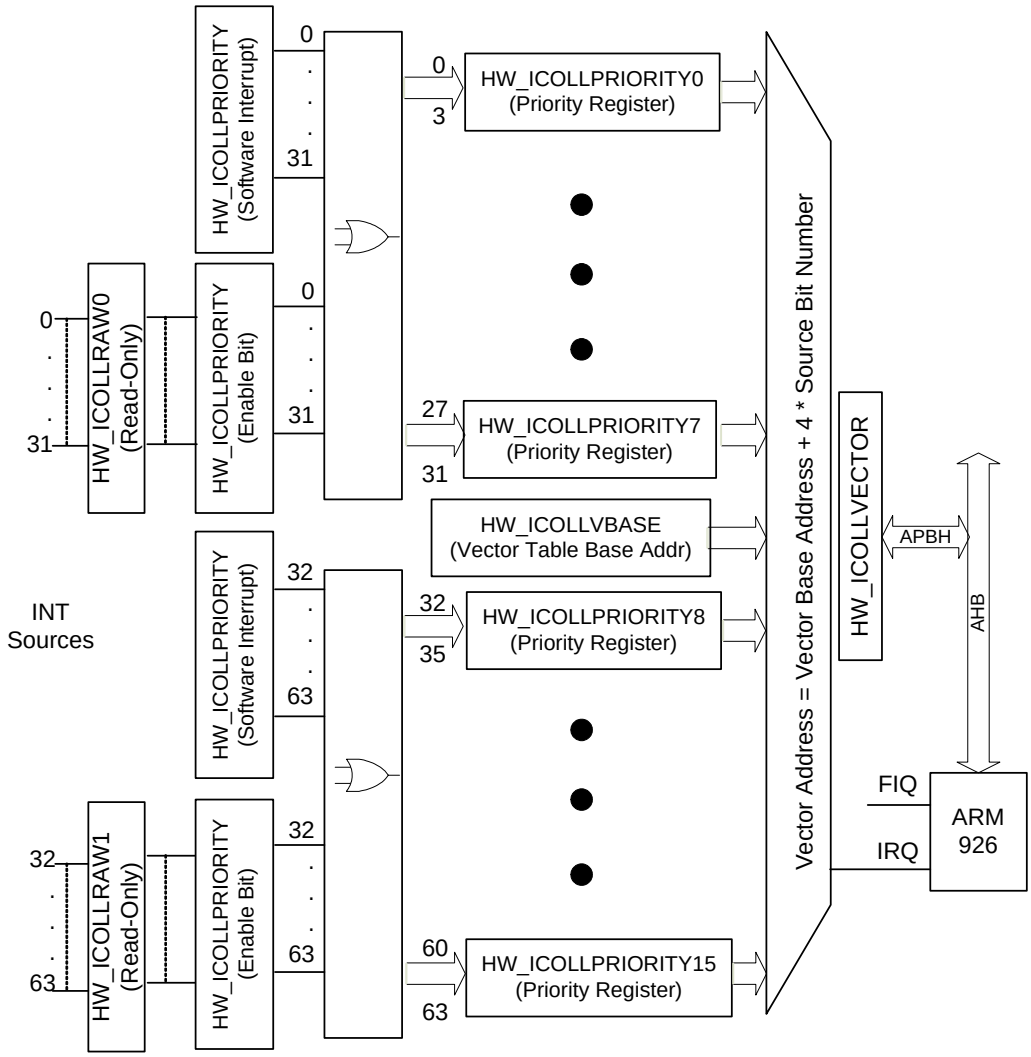


Figure 14. Interrupt Collector Diagram for IRQ Generation

FIQ interrupts are not prioritized, nor are they vectorized. Exactly four of the interrupt sources can be selected to generate the FIQ interrupt, source bits 32 through 35. If more than one is routed to the FIQ, then they must be discriminated by software. Generally, the FIQ is reserved for the exclusive use of brownout interrupts.

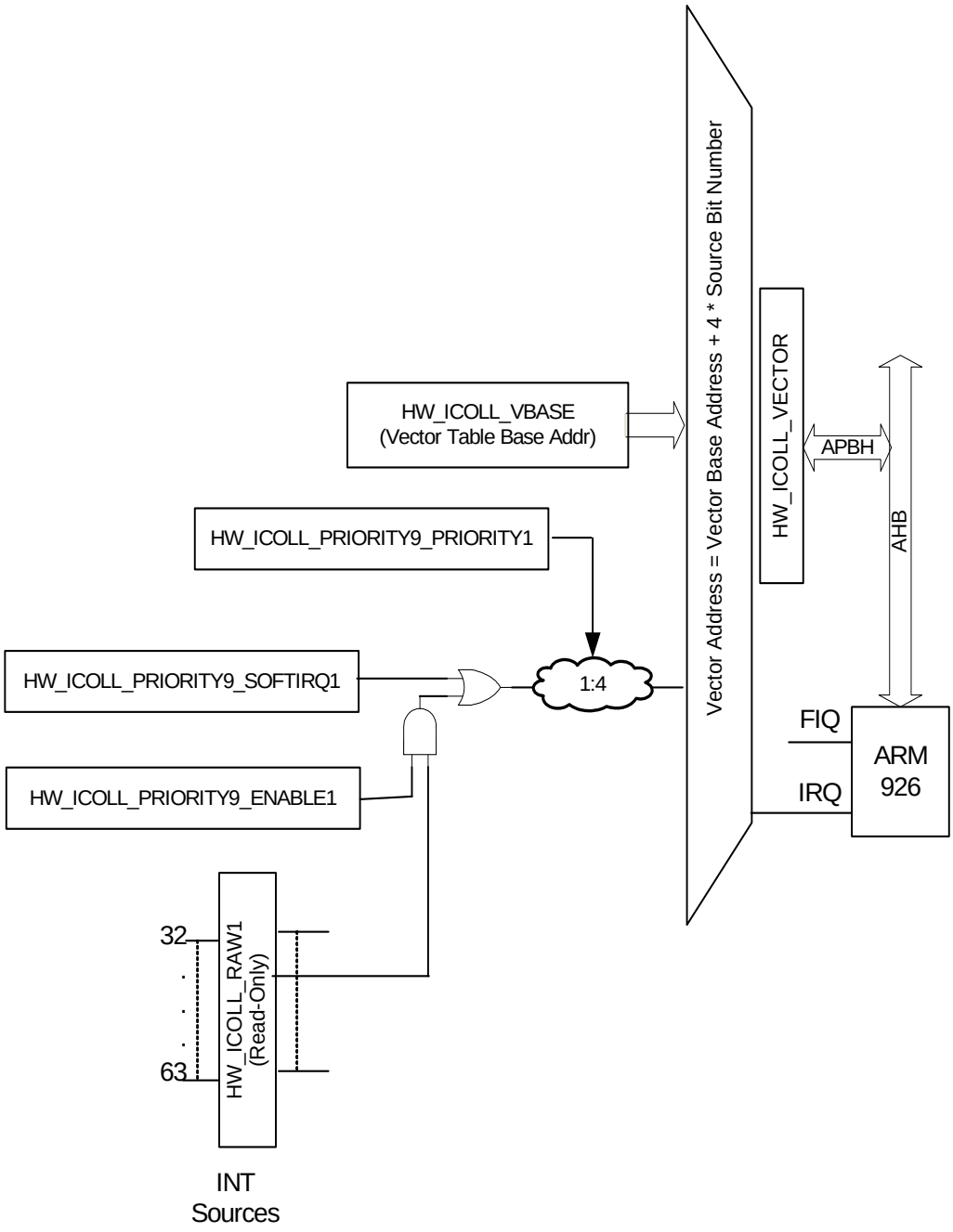


Figure 15. Interrupt Collector Bit "37" Logic

For a single interrupt source bit, there is an enable bit that gates it to the priority logic. A software interrupt bit per source bit can be used to force an interrupt at the appropriate priority level directed to the corresponding vector address. Each source can be applied to one of four interrupt levels, as shown in Figure 15.

The enable bit, the software interrupt bit, and the two-bit priority level specification for each interrupt source bit are contained in a byte in the programmable registers.

The data path for generating the vector address for the vectored interrupt portion of the interrupt collector is implemented as a multicycle path, as shown in Figure 16. The interrupt sources are continuously sampled in the holding register until one or more arrive. The FSM causes the holding register to stop sampling while a vector address is computed. Each interrupt source bit is applied to one of four levels based on the two-bit priority specification of each source bit. When the holding register "closes," there can be more than one newly arrived source bit. Thus, the source bits could be assigned such that more than one interrupt level is requesting an interrupt. The pipeline first determines the highest level requesting interrupt service. All interrupt requests on that level are presented to the linear priority encoder. The result of this stage is a six-bit number corresponding to the source bit number of the highest priority requesting an interrupt. This six-bit source number is used to compute the vector address as follows:

VectorAddress = VectorBase + (4 * SourceBitNumber)

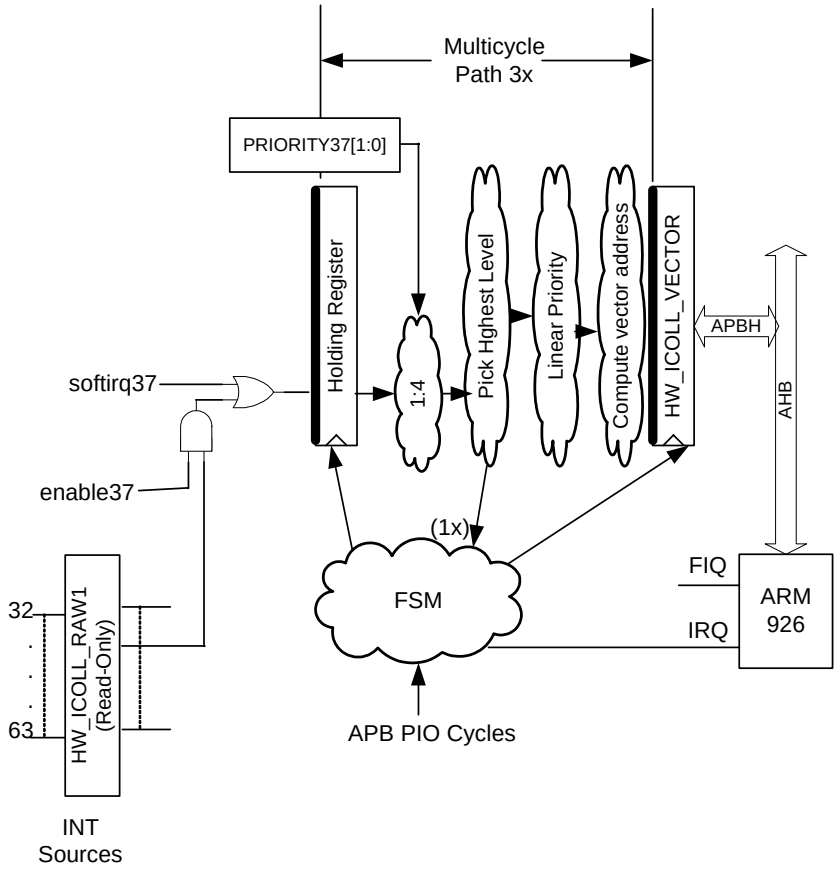


Figure 16. IRQ Control Flow

5.2. Nesting of Multi-Level IRQ Interrupts

There are a number of very important interactions between the interrupt collector's FSM and the interrupt service routine (ISR) running on the CPU. See Figure 17 for the following discussion.

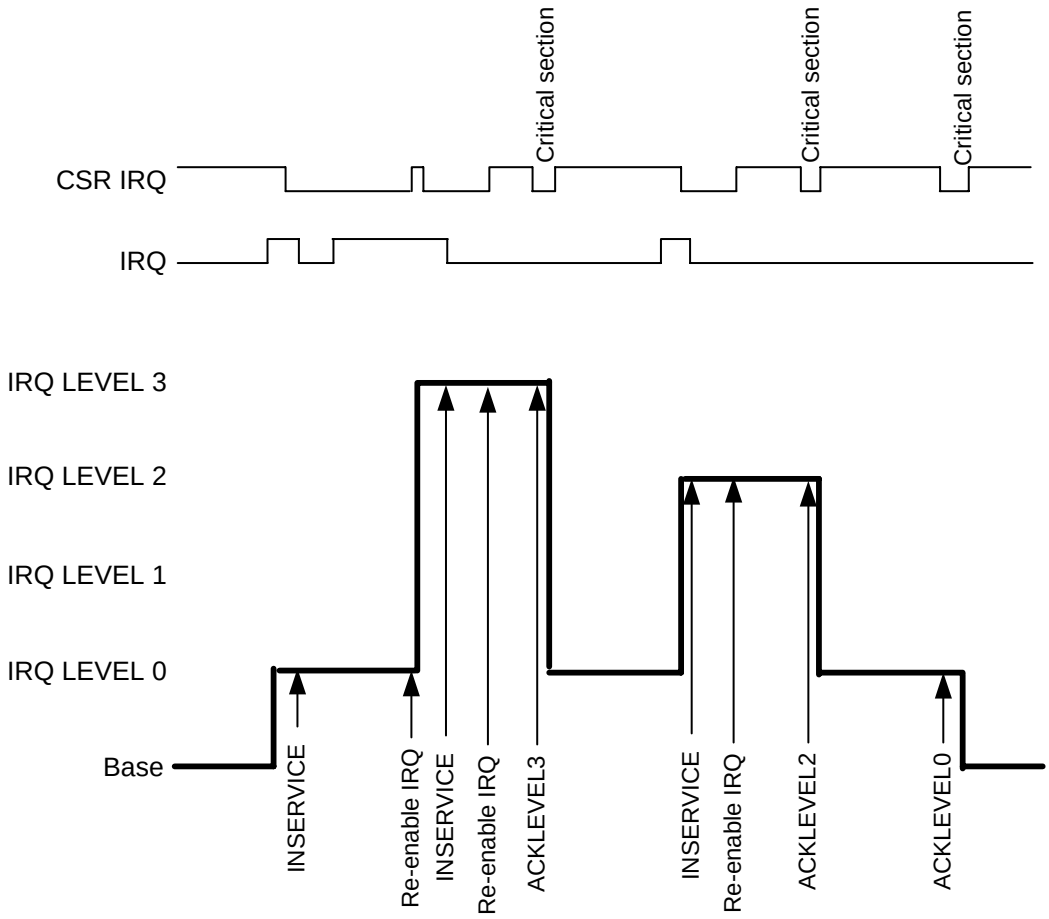


Figure 17. Nesting of Multi-Level IRQ Interrupts

As soon as the interrupt source is recognized in the holding register, the FSM delays two clocks, then grabs the vector address and asserts IRQ to the CPU. As soon as possible after the CPU enters the interrupt service routine, it must notify the interrupt collector. Software indicates the in-service state by writing to the HW_ICOLL_VECTOR register. The contents of the data bus on this write do not matter. Optionally, firmware can enable the ARM read side-effect mode. In this case, the in-service state is indicated as a side effect of having read the HW_ICOLL_VECTOR register at the exception vector (0xFFFF0018). At this point, the FSM reopens the holding register and scans for new interrupt sources. Any such IRQ sources are presented to the CPU, provided that they are at a level higher than any currently in-service level.

Whenever the ARM CPU takes an IRQ exception, it turns off the IRQ enable in the CPU status register (CSR), as shown in Figure 17. If a higher priority interrupt is pending at this point, then another IRQ exception is taken.

The example in Figure 17 shows going from the base to a level 0 ISR. When the ISR at level 0 was ready, it enabled IRQ interrupts. At this point, it nests IRQ interrupts up to a level 3 interrupt. The level 3 ISR marks its in-service state, which causes the interrupt collector to open the holding register to search for new interrupt sources. In this example, none comes in, so the level 3 ISR completes. As part of the return process, the ISR disables IRQ interrupts, then acknowledges the level 3 service state. This is accomplished by writing the level number (3 in this case) to the interrupt collector's Level Acknowledge register. The interrupt collector resets the in-service bit for level 3. If this enables an IRQ at level 3, then it asserts IRQ and goes through the nesting process again. Since IRQ exceptions are masked in the level 3 ISR, this nesting does not take place until the level 3 ISR returns from interrupt. This return automatically re-enables IRQ exceptions. At this point, another exception could occur.

Figure 17 shows a second nesting of the IRQ interrupt by the arrival of a level 2 interrupt source bit. Finally, the figure shows the point at which the level 0 ISR enters its critical section (masks IRQ) and acknowledges level 0 to the interrupt collector and returns from interrupt.

The FSM reverts to its "BASE" level state waiting for an interrupt request to arrive in the holding register. The waveform for the IRQ mask in the CPU status register (CSR) and the waveform for the IRQ input to the CPU as they relate to the interrupt collector action are shown in Figure 17.

WARNING: There is an inherent race condition between notifying the interrupt collector that an ISR has been entered and having that ISR re-enable IRQ exceptions in the CSR. The in-service notification can take a number of cycles to percolate through the write buffer, through the AHB and APB bridge and into the interrupt collector where it removes the IRQ assertion to the CPU. This ICOLL IRQ must be deasserted before the CSR IRQ on the CPU is re-enabled or the CPU will see a phantom interrupt. This is why the ARM vectored interrupt controller provides this in service notification as a read side effect of the vector address read. Alternatively, the ISR can read the interrupt collector's CSR. The value received is unimportant, but the time required to do the read ensures that the write data has arrived at the interrupt collector. If firmware uses this method, it should allow clocks after the read for the FSM and for the CPU to recognize that the IRQ has been deasserted.

5.3. FIQ Generation

Four of the interrupt source bits can be used to generate an FIQ instead of an IRQ exception. These are source bits 32 through 35, inclusive. An FIQ may be generated by one of four source bits. Figure 18 shows the FIQ sequence for interrupt source bit 33. When enabled to the FIQ, the software interrupt associated with these bits can be used to generate the FIQ from these sources for test purposes. FIQ for a given interrupt should be enabled only when the IRQ for that interrupt is disabled.

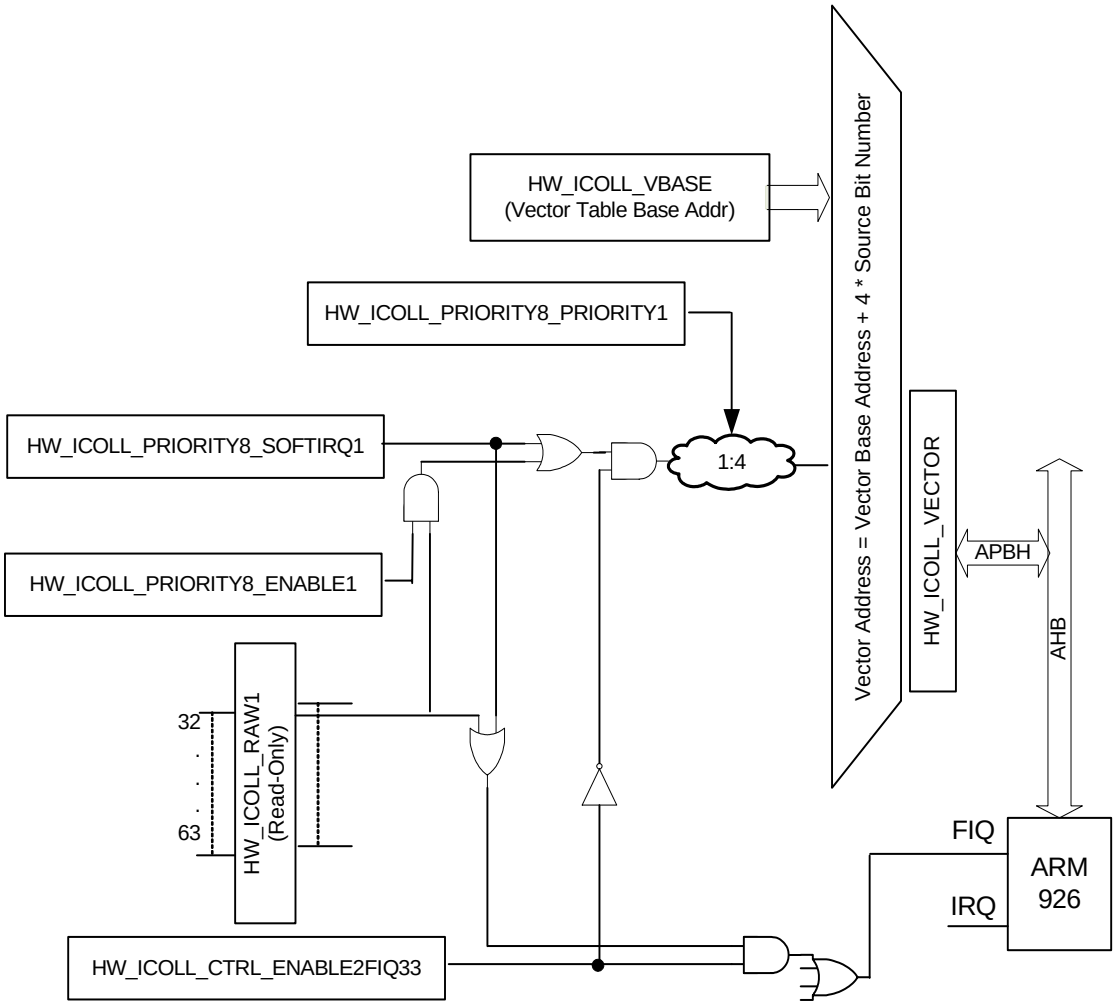


Figure 18. FIQ Generation Logic

5.4. Interrupt Sources

Table 37 lists all of the interrupt sources on the STMP36xx. Use hw_irq.h to access these bits.

Table 37. Interrupt Sources

INTERRUPT SOURCE	SRC	VECTOR	FIQ	DESCRIPTION
Debug UART	0	0x0000	NO	No DMA on the debug UART
COMMS RX	1	0x0004	NO	JTAG debug communications port
COMMS TX	2	0x0008	NO	JTAG debug communications port
VDD5V	3	0x000C	NO	IRQ on 5V connect or disconnect also OTG 4.2V
HEADPHONE_SHORT	4	0x0010	NO	Headphone short
DAC_DMA	5	0x0014	NO	DAC DMA channel
DAC_ERROR	6	0x0018	NO	DAC FIFO buffer underflow
ADC_DMA	7	0x001C	NO	ADC DMA channel
ADC_ERROR	8	0x0020	NO	ADC FIFO buffer overflow
SPDIF_DMA	9	0x0024	NO	SPDIF DMA channel
SPDIF_ERROR	10	0x0028	NO	SPDIF underflow
USB_CTRL	11	0x002C	NO	USB controller Interrupt
USB_WAKEUP	12	0x0030	NO	Also ARC core to remain suspended
GPMI_DMA	13	0x0034	NO	From DMA channel for GPMI
SSP_DMA	14	0x0038	NO	From DMA channel for SSP
SSP_ERROR	15	0x003C	NO	SSP device level error and status
GPIO0	16	0x0040	NO	GPIO bank 0 interrupt
GPIO1	17	0x0044	NO	GPIO bank 1 interrupt
GPIO2	18	0x0048	NO	GPIO bank 2 interrupt
GPIO3	19	0x004C	NO	GPIO bank 3 interrupt
ECC_DMA	20	0x0050	NO	From DMA channel for HWECC
ECC_ERROR	21	0x0054	NO	From the HWECC device itself
RTC_ALARM	22	0x0058	NO	RTC alarm event
UART_TX_DMA	23	0x005C	NO	Application UART transmitter DMAq
UART1_INTERNAL	24	0x0060	NO	Application UART internal error
UART_RX_DMA	25	0x0064	NO	Application UART receiver DMA interrupt
I2C_DMA	26	0x0068	NO	From DMA channel for I^2C
I2C_ERROR	27	0x006C	NO	From I^2C device detected errors and line conditions
TIMER0	28	0x0070	NO	TIMROT Timer0
TIMER1	29	0x0074	NO	TIMROT Timer1
TIMER2	30	0x0078	NO	TIMROT Timer2
TIMER3	31	0x007C	NO	TIMROT Timer3
BAT_BRNOUT	32	0x0080	YES	Power module battery brownout detect

Table 37. Interrupt Sources (Continued)

INTERRUPT SOURCE	SRC	VECTOR	FIQ	DESCRIPTION
VDDD_BRNOUT	33	0x0084	YES	Power module VDDD brownout detect
VDDIO_BRNOUT	34	0x0088	YES	Power module VDDIO brownout detect
VDD18_BRNOUT	35	0x008C	YES	Reserved for future use
TOUCH_IRQ	36	0x0090	NO	Touch detection
LRADC_CH0	37	0x0094	NO	Channel 0 complete
LRADC_CH1	38	0x0098	NO	Channel 1 complete
LRADC_CH2	39	0x009C	NO	Channel 2 complete
LRADC_CH3	40	0x00A0	NO	Channel 3 complete
LRADC_CH4	41	0x00A4	NO	Channel 4 complete
LRADC_CH5	42	0x00A8	NO	Channel 5 complete
LRADC_CH6	43	0x00AC	NO	Channel 6 complete
LRADC_CH7	44	0x00B0	NO	Channel 7 complete
MEMCPY_DMA_SRC	45	0x00B4	NO	From DMA channel for MEMCPY source
MEMCPY_DMA_DST	46	0x00B8	NO	From DMA channel for MEMCPY destination
LCD_DMA	47	0x00BC	NO	From DMA channel for LCD
RTC_1MSEC	48	0x00C0	NO	RTC 1-ms tick interrupt
DRI_DMA	49	0x00C4	NO	From DMA channel for DRI
DRI_ATTENTION	50	0x00C8	NO	From DRI internal error and attention IRQ
GPMI_ATTENTION	51	0x00CC	NO	From GPMI internal error and status IRQ
IR	52	0x00D0	NO	From IR (infrared) internals. Note that the IR shares DMA channels with the applications UART.
Reserved for future hardware	53–59	0X00D4–0x00E8	NO	Do not use these interrupts in STMP36xx.
SOFTWAREIRQ60–SOFTWAREIRQ63	60–63	0x00F0–0x00FC	NO	For software use.

5.5. CPU Wait-for-Interrupt Mode

To enable wait-for-interrupt mode, two distinct actions are required by the programmer.

1. Set the INTERRUPT_WAIT bit in the HW_CLKCTRL_CPUCLKCTRL register. This must be done via a RMW operation. For example:

```
uclkctrl  = HW_CLKCTRL_CPUCLKCTRL_RD();
uclkctrl |= BM_CLKCTRL_CPUCLKCTRL_INTERRUPT_WAIT;
HW_CLKCTRL_CPUCLKCTRL_WR(uclkctrl);
```

2. After setting the INTERRUPT_WAIT bit, a coprocessor instruction is required.

```
asm (
              // Note: R0 is used in the following example, but any usual
              // <Rd> register may be used.
"mov R0, 0;" // Rd SBZ (should be zero)
"mcr p15,0,r0,c7,c0,4;"//Drain write buffers, idle CPU clock & processor,
                       // and stop processor at this instruction
"nop");       // The lr sent to handler points here after RTI
```

The coprocessor instruction sequence above enables an internal gating signal. This internal signal guarantees that write buffers are drained and ensures that the processor is in an idle state. On execution of the MCR coprocessor instruction, the CPU clock is stopped and the processor halts on the instruction—waiting for an interrupt to occur.

The INTERRUPT_WAIT bit can be thought of as a Wait-for-Interrupt enable bit. Therefore, it must be set prior to execution of the MCR instruction. It is recommended that, when the Wait-for-Interrupt mode is to be used, the INTERRUPT_WAIT bit be set at initialization time and left on.

With the INTERRUPT_WAIT bit set, after execution of the MCR WFI command, the processor halts on the MCR instruction. When an interrupt or FIQ occurs, the MCR instruction completes and the IRQ or FIQ handler is entered normally. The return link that is passed to the handler is automatically adjusted by the above MCR instruction, such that a normal return from interrupt results in continuing execution at the instruction immediately following the MCR. That is, the LR will contain the address of the MCR instruction plus eight, such that a typical return from interrupt instruction (e.g., subs pc, LR, 4) will return to the instruction immediately following the MCR (the NOP in the example above).

Whenever the CPU is stopped because the clock control HW_CLKCTRL_CPUCLKCTRL_INTERRUPT_WAIT bit is set and the MCR WFI instruction is executed, the CPU stops until an interrupt occurs. The actual condition that wakes up the CPU is determined by ORing together all enabled interrupt requests including those that are directed to the FIQ CPU input. The ICOLL_BUSY output signal from the ICOLL communicates this information to the clock control. This function does not pass through the normal ICOLL state machine. It starts the CPU clock as soon as an enabled interrupt arrives.

5.6. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

5.7. Programmable Registers

The following registers provide interrupt generation and control for the STMP36xx.

5.7.1. Interrupt Collector Interrupt Vector Address Register Description

The Interrupt Collector Interrupt Vector Address Register is read by the Interrupt Service Routine (ISR) at the IRQ vector location (0xFFFF0018) using a load PC instruction. The priority logic presents the vector address of the next IRQ interrupt to be processed by the CPU. The vector address is held until acknowledged by a CPU write to the HW_ICOLL_VECTOR register.

HW_ICOLL_VECTOR 0x80000000
HW_ICOLL_VECTOR_SET 0x80000004
HW_ICOLL_VECTOR_CLR 0x80000008
HW_ICOLL_VECTOR_TOG 0x8000000C

Table 38. HW_ICOLL_VECTOR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IRQVECTOR																														RSRVD1	

Table 39. HW_ICOLL_VECTOR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**IRQVECTOR**	RW	0x0	This register presents the vector address for the interrupt currently active on the CPU IRQ input. Writing to this register notifies the interrupt collector that the interrupt service routine for the current interrupt has been entered.
1:0	**RSRVD1**	RO	0x0	Always write zeroes to this field.

DESCRIPTION:

This register mediates the vectored interrupt collectors interface with the CPU when it enteres the IRQ exception trap. The exception trap should have a LDPC instruction from this address.

EXAMPLE:

```
LDPC HW_ICOLL_VECTOR_ADDR; IRQ exception at 0xffff0018
```

5.7.2. Interrupt Collector Level Acknowledge Register Description

The Interrupt Collector Level Acknowledge Register is used by software to indicate the completion of an interrupt on a specific level.

HW_ICOLL_LEVELACK 0x80000010

Table 40. HW_ICOLL_LEVELACK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																												IRQLEVELACK			

Table 41. HW_ICOLL_LEVELACK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	**RSRVD1**	RO	0x0	Any value can be written to this bit field. Writes are ignored.
3:0	**IRQLEVELACK**	RW	0x0	This bit field is written by the processor to acknowledge the completion of an interrupt. The value written must correspond to the priority level of the completed interrupt: LEVEL0 = 0x1 Level 0 LEVEL1 = 0x2 Level 1 LEVEL2 = 0x4 Level 2 LEVEL3 = 0x8 Level 3

DESCRIPTION:

This register is written to advance the ICOLL internal IRQ state machine. It advances from an in-service on a level state to the next pending interrupt level or to the idle state. This register is written at the very end of an interrupt service routine. If nesting is used, then the CPU IRQ must be turned on before writing to this register to avoid a race condition in the CPU interrupt hardware. WARNING: The value written to the Interrupt Collector Level Acknowledge Register is decoded not binary, i.e., 8, 4, 2, 1.

EXAMPLE:

```
HW_ICOLL_LEVELACK_WR(HW_ICOLL_LEVELACK__LEVEL3);
```

5.7.3. Interrupt Collector Control Register Description

The Interrupt Collector Control Register provides overall control of interrupts being routed to the CPU. This register is not at offset zero from the block base because that location is needed for single 32-bit instructions to be placed in the exception vector location.

HW_ICOLL_CTRL 0x80000020
HW_ICOLL_CTRL_SET 0x80000024
HW_ICOLL_CTRL_CLR 0x80000028
HW_ICOLL_CTRL_TOG 0x8000002C

Table 42. HW_ICOLL_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSRVD3		ENABLE2FIQ35	ENABLE2FIQ34	ENABLE2FIQ33	ENABLE2FIQ32	RSRVD2			BYPASS_FSM	NO_NESTING	ARM_RSE_MODE	FIQ_FINAL_ENABLE	IRQ_FINAL_ENABLE	RSRVD1															

Table 43. HW_ICOLL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	When set to one, this bit causes a soft reset to the entire interrupt collector. This bit must be turned off for normal operation. RUN = 0x0 Allow the interrupt collector to operate normally. IN_RESET = 0x1 Hold the interrupt collector in its reset state.
30	**CLKGATE**	RW	0x1	When set to one, this bit causes all clocks within the interrupt collector to be gated off. WARNING: Do not set this bit at the same time as SFTRST. Doing so causes the softreset to have no effect. Setting SFTRST will cause the CLKGATE bit to set automatically four clocks later. RUN = 0x0 Enable clocks for normal operation of interrupt collector. NO_CLOCKS = 0x1 Disable clocking within the interrupt collector.
29:28	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
27	**ENABLE2FIQ35**	RW	0x0	Set this bit to one enable interrupt bit 35 as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
26	**ENABLE2FIQ34**	RW	0x0	Set this bit to one enable interrupt bit 34 as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25	**ENABLE2FIQ33**	RW	0x0	Set this bit to one enable interrupt bit 33 as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
24	**ENABLE2FIQ32**	RW	0x0	Set this bit to one enable interrupt bit 32 as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
23:21	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.

Table 43. HW_ICOLL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	**BYPASS_FSM**	RW	0x0	Set this bit to one to bypass the FSM control of the request holding register and the vector address. With this bit set to one, the vector address register is continuously updated as interrupt requests come in. Turn off all enable bits and walk a one through the software interrupts, observing the vector address changes. Set to zero for normal operation. This control is included as a test mode and is not intended for use by a real application. NORMAL = 0x0 Normal BYPASS = 0x1 No FSM handshake with CPU
19	**NO_NESTING**	RW	0x0	Set this bit to one disable interrupt level nesting, i.e., higher priority interrupt interrupting lower priority. For normal operation, set this bit to zero. NORMAL = 0x0 Normal NO_NEST = 0x1 No support for interrupt nesting
18	**ARM_RSE_MODE**	RW	0x0	Set this bit to one enable the ARM-style read side effect associated with the vector address register. In this mode, interrupt inservice is signaled by the read of the HW_ICOLL_VECTOR register to acquire the interrupt vector address. Set this bit to zero for normal operation, in which the ISR signals inservice explicitly by means of a write to the HW_ICOLL_VECTOR register. MUST_WRITE = 0x0 Must write to vector register to go in-service READ_SIDE_EFFECT = 0x1 Go in-service as a read side effect
17	**FIQ_FINAL_ENABLE**	RW	0x1	Set this bit to one to enable the final FIQ output to the CPU. Set this bit to zero for testing the interrupt collector without causing actual CPU interrupts. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
16	**IRQ_FINAL_ENABLE**	RW	0x1	Set this bit to one to enable the final IRQ output to the CPU. Set this bit to zero for testing the interrupt collector without causing actual CPU interrupts. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
15:0	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

DESCRIPTION:

This register handles the overall control of the interrupt collector, including soft reset and clock gate. In addition, it handles state machine variations such as NO_NESTING and ARM read side effect processing on the vector address register.

EXAMPLE:

```
HW_ICOLL_CTRL_CLR(BM_ICOLL_CTRL_SFTRST | BM_ICOLL_CTRL_SFTRST );
```

5.7.4. Interrupt Collector Status Register Description

The Interrupt Collector Status Register provides a read-only view into various internal states, including the vector number of the current interupt.

HW_ICOLL_STAT 0x80000030

Table 44. HW_ICOLL_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																										VECTOR_NUMBER					

Table 45. HW_ICOLL_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:6	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
5:0	**VECTOR_NUMBER**	RO	0x0	Vector number of current interrupt. Multiply by 4 and add to vector base address to obtain the value in HW_ICOLL_VECTOR.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
  if(HW_ICOLL_STAT_VECTOR_NUMBER_READ() == 0x00000017) ISR_vector_23(); // ISR for vector 23
decimal, 17 hex
```

5.7.5. Interrupt Collector Raw Interrupt Input Register 0 Description

The lower 32 interrupt source states are visible in this read-only register.

HW_ICOLL_RAW0 0x80000040
HW_ICOLL_RAW0_SET 0x80000044
HW_ICOLL_RAW0_CLR 0x80000048
HW_ICOLL_RAW0_TOG 0x8000004C

Table 46. HW_ICOLL_RAW0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RAW_IRQS																															

Table 47. HW_ICOLL_RAW0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RAW_IRQS**	RO	0x0	Read-only view of the lower 32 interrupt request bits.

DESCRIPTION:

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. Its purpose is to improve diagnostic observability.

EXAMPLE:

```
  ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

5.7.6. Interrupt Collector Raw Interrupt Input Register 1 Description

The upper 32 interrupt source states are visible in this read-only register.

HW_ICOLL_RAW1 0x80000050
HW_ICOLL_RAW1_SET 0x80000054
HW_ICOLL_RAW1_CLR 0x80000058
HW_ICOLL_RAW1_TOG 0x8000005C

Table 48. HW_ICOLL_RAW1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RAW_IRQS																															

Table 49. HW_ICOLL_RAW1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RAW_IRQS**	RO	0x0	Read-only view of the upper 32 interrupt request bits.

DESCRIPTION:

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. The purpose is to improve diagnostic observability.

EXAMPLE:

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

5.7.7. Interrupt Collector Priority Register 0 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY0 0x80000060
HW_ICOLL_PRIORITY0_SET 0x80000064
HW_ICOLL_PRIORITY0_CLR 0x80000068
HW_ICOLL_PRIORITY0_TOG 0x8000006C

Table 50. HW_ICOLL_PRIORITY0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 51. HW_ICOLL_PRIORITY0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 3 FORCE_INTERRUPT = 0x1 Force a software interrupt

Table 51. HW_ICOLL_PRIORITY0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 3. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 3. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 2. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 2. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 2. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 1. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 1. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 1. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 0. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 0. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 0. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated

interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(0,0x00000001);
```

5.7.8. Interrupt Collector Priority Register 1 Description

The Interrupt Collector Priority Register 1 provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY1 0x80000070
HW_ICOLL_PRIORITY1_SET 0x80000074
HW_ICOLL_PRIORITY1_CLR 0x80000078
HW_ICOLL_PRIORITY1_TOG 0x8000007C

Table 52. HW_ICOLL_PRIORITY1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 53. HW_ICOLL_PRIORITY1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 7. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 7. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 7. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 6. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 6. DISABLE = 0x0 Disable ENABLE = 0x1 Enable

Table 53. HW_ICOLL_PRIORITY1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 6. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 5. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 5. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 5. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 4. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 4. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 4. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(1,0x00000001);
```

5.7.9. Interrupt Collector Priority Register 2 Description

The Interrupt Collector Priority Register 2 provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY2 0x80000080
HW_ICOLL_PRIORITY2_SET 0x80000084
HW_ICOLL_PRIORITY2_CLR 0x80000088
HW_ICOLL_PRIORITY2_TOG 0x8000008C

Table 54. HW_ICOLL_PRIORITY2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 55. HW_ICOLL_PRIORITY2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 11. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 11. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 11. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 10. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 10. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 10. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 9. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 9. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 9. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

Table 55. HW_ICOLL_PRIORITY2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 8. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 8. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 8. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(2,0x00000001);
```

5.7.10. Interrupt Collector Priority Register 3 Description

The Interrupt Collector Priority Register 3 provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY3 0x80000090
HW_ICOLL_PRIORITY3_SET 0x80000094
HW_ICOLL_PRIORITY3_CLR 0x80000098
HW_ICOLL_PRIORITY3_TOG 0x8000009C

Table 56. HW_ICOLL_PRIORITY3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 57. HW_ICOLL_PRIORITY3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 15. FORCE_INTERRUPT = 0x1 Force a software interrupt

Table 57. HW_ICOLL_PRIORITY3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 15. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 15. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 14. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 14. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 14. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 13. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 13. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 13. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 12. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 12. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 12. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated

interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(3,0x00000001);
```

5.7.11. Interrupt Collector Priority Register 4 Description

The Interrupt Collector Priority Register 4 provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY4 0x800000A0
HW_ICOLL_PRIORITY4_SET 0x800000A4
HW_ICOLL_PRIORITY4_CLR 0x800000A8
HW_ICOLL_PRIORITY4_TOG 0x800000AC

Table 58. HW_ICOLL_PRIORITY4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 59. HW_ICOLL_PRIORITY4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 19. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 19. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 19. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 18. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 18. DISABLE = 0x0 Disable ENABLE = 0x1 Enable

Table 59. HW_ICOLL_PRIORITY4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 18. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 17. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 17. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 17. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 16. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 16. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 16. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(4,0x00000001);
```

5.7.12. Interrupt Collector Priority Register 5 Description

The Interrupt Collector Priority Register 5 provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY5 0x800000B0
HW_ICOLL_PRIORITY5_SET 0x800000B4
HW_ICOLL_PRIORITY5_CLR 0x800000B8
HW_ICOLL_PRIORITY5_TOG 0x800000BC

Table 60. HW_ICOLL_PRIORITY5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 61. HW_ICOLL_PRIORITY5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 23. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 23. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 23. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 22. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 22. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 22. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 21. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 21. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 21. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

Table 61. HW_ICOLL_PRIORITY5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 20. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 20. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 20. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(5,0x00000001);
```

5.7.13. Interrupt Collector Priority Register 6 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY6 0x800000C0
HW_ICOLL_PRIORITY6_SET 0x800000C4
HW_ICOLL_PRIORITY6_CLR 0x800000C8
HW_ICOLL_PRIORITY6_TOG 0x800000CC

Table 62. HW_ICOLL_PRIORITY6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 63. HW_ICOLL_PRIORITY6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 27. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 27. DISABLE = 0x0 Disable ENABLE = 0x1 Enable

Table 63. HW_ICOLL_PRIORITY6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 27. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 26. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 26. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 26. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 25. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 25. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 25. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 24. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 24. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 24. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(6,0x00000001);
```

5.7.14. Interrupt Collector Priority Register 7 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY7 0x800000D0
HW_ICOLL_PRIORITY7_SET 0x800000D4
HW_ICOLL_PRIORITY7_CLR 0x800000D8
HW_ICOLL_PRIORITY7_TOG 0x800000DC

Table 64. HW_ICOLL_PRIORITY7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 65. HW_ICOLL_PRIORITY7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 31. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 31. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 31. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 30. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 30. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 30. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.

Table 65. HW_ICOLL_PRIORITY7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 29. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 29. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 29. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 28. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 28. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 28. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(7,0x00000001);
```

5.7.15. Interrupt Collector Priority Register 8 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY8 0x800000E0
HW_ICOLL_PRIORITY8_SET 0x800000E4
HW_ICOLL_PRIORITY8_CLR 0x800000E8
HW_ICOLL_PRIORITY8_TOG 0x800000EC

Table 66. HW_ICOLL_PRIORITY8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 67. HW_ICOLL_PRIORITY8 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 35. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 35. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 35. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 34. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 34. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 34. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 33. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 33. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 33. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

Table 67. HW_ICOLL_PRIORITY8 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 32. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 32. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 32. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(8,0x00000001);
```

5.7.16. Interrupt Collector Priority Register 9 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY9 0x800000F0
HW_ICOLL_PRIORITY9_SET 0x800000F4
HW_ICOLL_PRIORITY9_CLR 0x800000F8
HW_ICOLL_PRIORITY9_TOG 0x800000FC

Table 68. HW_ICOLL_PRIORITY9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 69. HW_ICOLL_PRIORITY9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 39. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 39. DISABLE = 0x0 Disable ENABLE = 0x1 Enable

Table 69. HW_ICOLL_PRIORITY9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 39. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 38. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 38. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 38. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 37. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 37. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 37. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 36. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 36. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 36. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(9,0x00000001);
```

5.7.17. Interrupt Collector Priority Register 10 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY10 0x80000100
HW_ICOLL_PRIORITY10_SET 0x80000104
HW_ICOLL_PRIORITY10_CLR 0x80000108
HW_ICOLL_PRIORITY10_TOG 0x8000010C

Table 70. HW_ICOLL_PRIORITY10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 71. HW_ICOLL_PRIORITY10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 43. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 43. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 43. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 42. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 42. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 42. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.

Table 71. HW_ICOLL_PRIORITY10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 41. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 41. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 41. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 40. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 40. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 40. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(10,0x00000001);
```

5.7.18. Interrupt Collector Priority Register 11 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY11 0x80000110
HW_ICOLL_PRIORITY11_SET 0x80000114
HW_ICOLL_PRIORITY11_CLR 0x80000118
HW_ICOLL_PRIORITY11_TOG 0x8000011C

Table 72. HW_ICOLL_PRIORITY11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 73. HW_ICOLL_PRIORITY11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 47. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 47. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 47. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 46. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 46. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 46. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 45. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 45. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 45. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

Table 73. HW_ICOLL_PRIORITY11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 44. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 44. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 44. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(11,0x00000001);
```

5.7.19. Interrupt Collector Priority Register 12 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY12 0x80000120
HW_ICOLL_PRIORITY12_SET 0x80000124
HW_ICOLL_PRIORITY12_CLR 0x80000128
HW_ICOLL_PRIORITY12_TOG 0x8000012C

Table 74. HW_ICOLL_PRIORITY12

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 75. HW_ICOLL_PRIORITY12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 51. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 51. DISABLE = 0x0 Disable ENABLE = 0x1 Enable

Table 75. HW_ICOLL_PRIORITY12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 51. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 50. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 50. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 50. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 49. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 49. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 49. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 48. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 48. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 48. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(12,0x00000001);
```

5.7.20. Interrupt Collector Priority Register 13 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY13 0x80000130
HW_ICOLL_PRIORITY13_SET 0x80000134
HW_ICOLL_PRIORITY13_CLR 0x80000138
HW_ICOLL_PRIORITY13_TOG 0x8000013C

Table 76. HW_ICOLL_PRIORITY13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 77. HW_ICOLL_PRIORITY13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 55. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 55. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 55. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 54. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 54. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 54. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.

Table 77. HW_ICOLL_PRIORITY13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 53. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 53. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 53. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 52. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 52. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 52. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(13,0x00000001);
```

5.7.21. Interrupt Collector Priority Register 14 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY14 0x80000140
HW_ICOLL_PRIORITY14_SET 0x80000144
HW_ICOLL_PRIORITY14_CLR 0x80000148
HW_ICOLL_PRIORITY14_TOG 0x8000014C

Table 78. HW_ICOLL_PRIORITY14

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 79. HW_ICOLL_PRIORITY14 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 59. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 59. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 59. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 58. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 58. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 58. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 57. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 57. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 57. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

Table 79. HW_ICOLL_PRIORITY14 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 56. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 56. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 56. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(14,0x00000001);
```

5.7.22. Interrupt Collector Priority Register 15 Description

This register provides a mechanism to specify the priority level for four interrupt sources. It also provides an enable and software interrupt for each one.

HW_ICOLL_PRIORITY15 0x80000150
HW_ICOLL_PRIORITY15_SET 0x80000154
HW_ICOLL_PRIORITY15_CLR 0x80000158
HW_ICOLL_PRIORITY15_TOG 0x8000015C

Table 80. HW_ICOLL_PRIORITY15

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				SOFTIRQ3	ENABLE3	PRIORITY3		RSRVD3				SOFTIRQ2	ENABLE2	PRIORITY2		RSRVD2				SOFTIRQ1	ENABLE1	PRIORITY1		RSRVD1				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 81. HW_ICOLL_PRIORITY15 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Always write zeroes to this bit field.
27	**SOFTIRQ3**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 63. FORCE_INTERRUPT = 0x1 Force a software interrupt
26	**ENABLE3**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 63. DISABLE = 0x0 Disable ENABLE = 0x1 Enable

Table 81. HW_ICOLL_PRIORITY15 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:24	**PRIORITY3**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 63. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
23:20	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
19	**SOFTIRQ2**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 62. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
18	**ENABLE2**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 62. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
17:16	**PRIORITY2**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 62. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
15:12	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
11	**SOFTIRQ1**	RW	0x0	Set this bit to one to force a software interrupt. IRQ Bit 61. NO_INTERRUPT = 0x0 Turn off the software interrupt request FORCE_INTERRUPT = 0x1 Force a software interrupt
10	**ENABLE1**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 61. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
9:8	**PRIORITY1**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 61. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority
7:4	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
3	**SOFTIRQ0**	RW	0x0	Set this bit to one to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 60. FORCE_INTERRUPT = 0x1 Force a software interrupt
2	**ENABLE0**	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 60. DISABLE = 0x0 Disable ENABLE = 0x1 Enable
1:0	**PRIORITY0**	RW	0x0	Set the priority level for this bit, 0x3 is highest, 0x0 is lowest (weakest). IRQ Bit 60. LEVEL0 = 0x0 Level 0, lowest or weakest priority LEVEL1 = 0x1 Level 1 LEVEL2 = 0x2 Level 2 LEVEL3 = 0x3 Level 3, highest or strongest priority

DESCRIPTION:

This register provides a mechanism to specify the priority associated with four interrupt bits. In addition, this register controls the enable and software-generated interrupts for the four interrupt input bits. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. Always disable an interrupt prior to changing its priority.

EXAMPLE:

```
HW_ICOLL_PRIORITYn_SET(15,0x00000001);
```

5.7.23. Interrupt Collector Interrupt Vector Base Address Register Description

The Interrupt Collector Interrupt Vector Base Address Register is used by the priority logic to generate a unique vector address for each of the 64 interrupt request lines coming into the interrupt collector. The vector address is formed by multiply the interrupt bit number by 4 and adding it to the vector base address.

HW_ICOLL_VBASE 0x80000160
HW_ICOLL_VBASE_SET 0x80000164
HW_ICOLL_VBASE_CLR 0x80000168
HW_ICOLL_VBASE_TOG 0x8000016C

Table 82. HW_ICOLL_VBASE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TABLE_ADDRESS																														RSRVD1	

Table 83. HW_ICOLL_VBASE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**TABLE_ADDRESS**	RW	0x0	This bit field holds the upper 30 bits of the base address of the vector table.
1:0	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

DESCRIPTION:

This register provides a mechanism to specify the base address of the interrupt vector table. It is used in the computation of the value supplied in HW_ICOLL_VECTOR register.

EXAMPLE:

```
HW_ICOLL_VBASE_WR(pInterruptVectorTable);
```

5.7.24. Interrupt Collector Debug Register 0 Description

The contents of this register will be defined as the hardware is developed.

HW_ICOLL_DEBUG 0x80000170
HW_ICOLL_DEBUG_SET 0x80000174
HW_ICOLL_DEBUG_CLR 0x80000178
HW_ICOLL_DEBUG_TOG 0x8000017C

Table 84. HW_ICOLL_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INSERVICE				LEVEL_REQUESTS				REQUESTS_BY_LEVEL				RSRVD2		FIQ	IRQ	RSRVD1						VECTOR_FSM									

Table 85. HW_ICOLL_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**INSERVICE**	RO	0x0	Read-only view of the Inservice bits used for nesting IRQs. LEVEL0 = 0x1 LEVEL0 LEVEL1 = 0x2 LEVEL1 LEVEL2 = 0x4 LEVEL2 LEVEL3 = 0x8 LEVEL3
27:24	**LEVEL_REQUESTS**	RO	0x0	Read-only view of the requsts by priority level for the current IRQ. LEVEL0 = 0x1 LEVEL0 LEVEL1 = 0x2 LEVEL1 LEVEL2 = 0x4 LEVEL2 LEVEL3 = 0x8 LEVEL3
23:20	**REQUESTS_BY_LEVEL**	RO	0x0	Read-only view of the requsts by priority level for the current IRQ. LEVEL0 = 0x1 LEVEL0 LEVEL1 = 0x2 LEVEL1 LEVEL2 = 0x4 LEVEL2 LEVEL3 = 0x8 LEVEL3
19:18	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
17	**FIQ**	RO	0x0	Read-only view of the FIQ output to the CPU. NO_FIQ_REQUESTED = 0x0 No FIQ Requested FIQ_REQUESTED = 0x1 FIQ Requested
16	**IRQ**	RO	0x0	Read-only view of the FIQ output to the CPU. NO_IRQ_REQUESTED = 0x0 No IRQ Requested IRQ_REQUESTED = 0x1 IRQ Requested
15:10	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
9:0	**VECTOR_FSM**	RO	0x0	Empty description. FSM_IDLE = 0x000 FSM_IDLE FSM_MULTICYCLE1 = 0x001 FSM_MULTICYCLE1 FSM_MULTICYCLE2 = 0x002 FSM_MULTICYCLE2 FSM_PENDING = 0x004 FSM_PENDING FSM_MULTICYCLE3 = 0x008 FSM_MULTICYCLE3 FSM_MULTICYCLE4 = 0x010 FSM_MULTICYCLE4 FSM_ISR_RUNNING1 = 0x020 FSM_ISR_RUNNING1 FSM_ISR_RUNNING2 = 0x040 FSM_ISR_RUNNING2 FSM_ISR_RUNNING3 = 0x080 FSM_ISR_RUNNING3 FSM_MULTICYCLE5 = 0x100 FSM_MULTICYCLE5 FSM_MULTICYCLE6 = 0x200 FSM_MULTICYCLE6

DESCRIPTION:

This register provides diagnostic visibility into the IRQ request state machine and its various inputs.

EXAMPLE:

```
 if (BF_RD(ICOLL_DEBUG, LEVEL_REQUESTS) != HW_ICOLL_DEBUG_LEVEL_REQUESTS__LEVEL3)
 Error();
 TPRINTF(TP_MED, ("ICOLL INSERVICE = 0x%x
INSERVICE)));
 TPRINTF(TP_MED, ("ICOLL STATE = 0x%x
VECTOR_FSM)));
```

5.7.25. Interrupt Collector Debug Read Register 0 Description

This register always returns a known read value for debug purposes.

HW_ICOLL_DBGREAD0 0x80000180
HW_ICOLL_DBGREAD0_SET 0x80000184
HW_ICOLL_DBGREAD0_CLR 0x80000188
HW_ICOLL_DBGREAD0_TOG 0x8000018C

Table 86. HW_ICOLL_DBGREAD0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 87. HW_ICOLL_DBGREAD0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0xECA94567	Fixed read-only value.

DESCRIPTION:

This register is used to test the read mux paths on the APBH.

EXAMPLE:

```
if (HW_ICOLL_DBGREADn_RD(0) != 0xECA94567)
Error();
```

5.7.26. Interrupt Collector Debug Read Register 1 Description

This register always returns a known read value for debug purposes.

HW_ICOLL_DBGREAD1 0x80000190
HW_ICOLL_DBGREAD1_SET 0x80000194
HW_ICOLL_DBGREAD1_CLR 0x80000198
HW_ICOLL_DBGREAD1_TOG 0x8000019C

Table 88. HW_ICOLL_DBGREAD1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 89. HW_ICOLL_DBGREAD1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0x1356DA98	Fixed read-only value.

DESCRIPTION:

This register is used to test the read mux paths on the APBH.

EXAMPLE:

```
if (HW_ICOLL_DBGREADn_RD(1) != 0x1356DA98)
```

```
Error();
```

5.7.27. Interrupt Collector Debug Flag Register Description

The Interrupt Collector Debug Flag Register is used to post diagnostic state into simulation.

HW_ICOLL_DBGFLAG 0x800001A0
HW_ICOLL_DBGFLAG_SET 0x800001A4
HW_ICOLL_DBGFLAG_CLR 0x800001A8
HW_ICOLL_DBGFLAG_TOG 0x800001AC

Table 90. HW_ICOLL_DBGFLAG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																FLAG															

Table 91. HW_ICOLL_DBGFLAG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
15:0	**FLAG**	RW	0x0	This debug facility is probably temporary.

DESCRIPTION:

This register provides a posting register to synchronize C program execution and the internal simulation environment.

EXAMPLE:

```
BF_WR(ICOLL_DBGFLAG, FLAG, 3);
// ... do some diagnostic action
BF_WR(ICOLL_DBGFLAG, FLAG, 4);
// ... do some more diagnostic actions
BF_WR(ICOLL_DBGFLAG, FLAG, 5);
```

5.7.28. Interrupt Collector Debug Read Request Register 0 Description

The Interrupt Collector Debug Read Request Register 0 provides a read-only view into the low 32 bits of the request holding register.

HW_ICOLL_DBGREQUEST0 0x800001B0
HW_ICOLL_DBGREQUEST0_SET 0x800001B4
HW_ICOLL_DBGREQUEST0_CLR 0x800001B8
HW_ICOLL_DBGREQUEST0_TOG 0x800001BC

Table 92. HW_ICOLL_DBGREQUEST0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 93. HW_ICOLL_DBGREQUEST0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	Low 32 bits of the request holding register.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
if (HW_ICOLL_DBGREQUESTn_RD(0) != 0x00000000)
Error();
```

5.7.29. Interrupt Collector Debug Read Request Register 1 Description

The Interrupt Collector Debug Read Request Register 1 provides a read-only view into the high 32 bits of the request holding register.

HW_ICOLL_DBGREQUEST1 0x800001C0
HW_ICOLL_DBGREQUEST1_SET 0x800001C4
HW_ICOLL_DBGREQUEST1_CLR 0x800001C8
HW_ICOLL_DBGREQUEST1_TOG 0x800001CC

Table 94. HW_ICOLL_DBGREQUEST1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 95. HW_ICOLL_DBGREQUEST1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RO	0x0	High 32 bits of the request holding register.

DESCRIPTION:

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE:

```
if (HW_ICOLL_DBGREQUESTn_RD(n) != 0x00000000)
Error();
```

ICOLL XML Revision: 1.51

6. DEFAULT FIRST-LEVEL PAGE TABLE FOR ARM926 MMU

This chapter describes the default first-level page table for the ARM926 MMU.

6.1. Overview

The STMP36xx contains a compact hardware implementation of a default 16-Kbyte first-level page table for the MMU. This area-efficient implementation allows a cost-effective alternative to allocating 16 Kbytes of on-chip SRAM to hold this extremely sparse table. This is particularly important for applications that do not include external SDRAM. The default page table begins at address 0x800C0000 and runs through 0x800C3FFF, as shown in Figure 19. Firmware can point the ARM926 MMU's Translation Base Address Register to this default first-level page table by loading it with 0x800C0000.

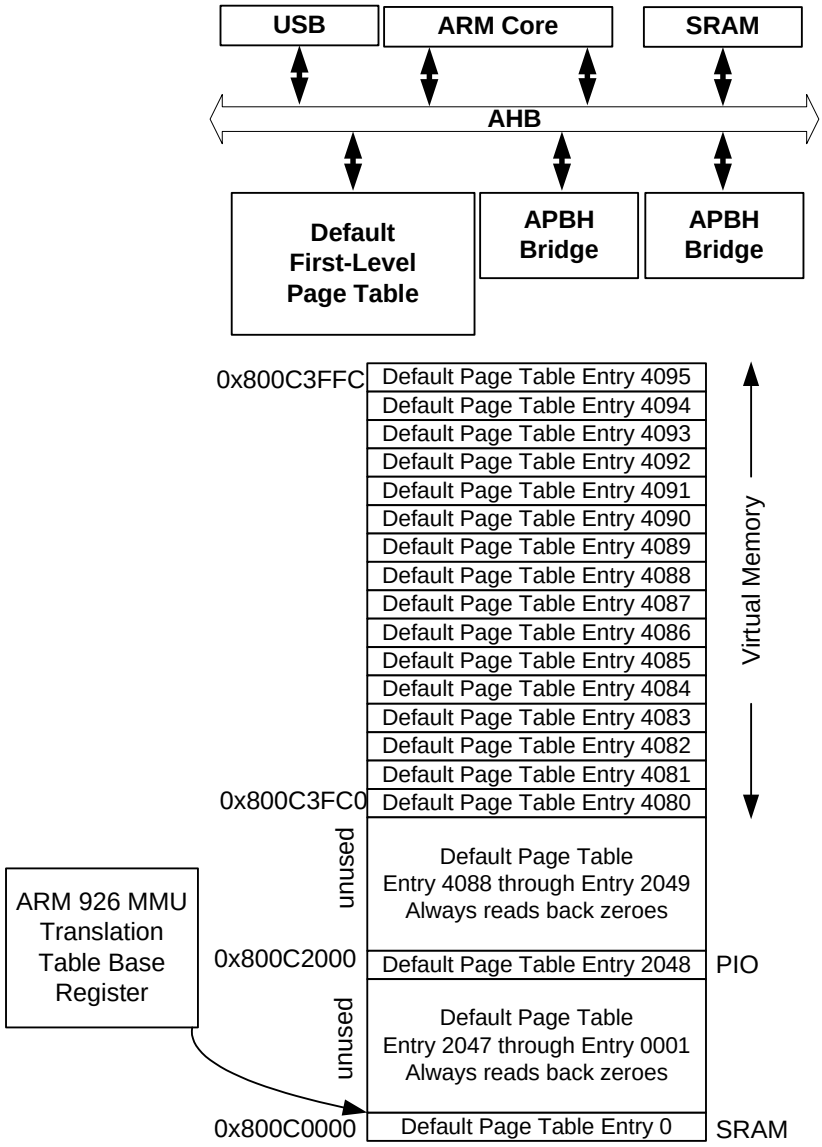


Figure 19. Default First-Level Page Table (DFLPT) Block Diagram

Table 96. Default First-Level Page Table

VIRTUAL ADDRESS	FLPT ENTRY #	DFLPT AHB ADDRESS	COARSE SECONDARY PAGE TABLE POINTER	FINE SECONDARY PAGE TABLE POINTER	USAGE, DOMAIN AND AP VALUES
0xFFFXXXXX	4095	0x800C3FFC	0x0003FC00	0x0003F000	Can select all four first-level descriptor options. Default is V==R section covering ROM at 0xFFFF0000. Domain, AP and CB can be specified for the V==R section.
0xFFEXXXXX	4094	0x800C3FF8	0x0003F800	0x0003E000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFFDXXXXX	4093	0x800C3FF4	0x0003F400	0x0003D000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFFCXXXXX	4092	0x800C3FF0	0x0003F000	0x0003C000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFFBXXXXX	4091	0x800C3FEC	0x0003EC00	0x0003B000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFFAXXXXX	4090	0x800C3FE8	0x0003E800	0x0003A000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF9XXXXX	4089	0x800C3FE4	0x0003E400	0x00039000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF8XXXXX	4088	0x800C3FE0	0x0003E000	0x00038000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF7XXXXX	4087	0x800C3FDC	0x0003DC00	0x00037000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF6XXXXX	4086	0x800C3FD8	0x0003D800	0x00036000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF5XXXXX	4085	0x800C3FD4	0x0003D400	0x00035000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.

Table 96. Default First-Level Page Table (Continued)

VIRTUAL ADDRESS	FLPT ENTRY #	DFLPT AHB ADDRESS	COARSE SECONDARY PAGE TABLE POINTER	FINE SECONDARY PAGE TABLE POINTER	USAGE, DOMAIN AND AP VALUES
0xFF4XXXXX	4084	0x800C3FD0	0x0003D000	0x00034000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF3XXXXX	4083	0x800C3FCC	0x0003CC00	0x00033000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF2XXXXX	4082	0x800C3FC8	0x0003C800	0x00032000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF1XXXXX	4081	0x800C3FC4	0x0003C400	0x00031000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFF0XXXXX	4080	0x800C3FC0	0x0003C000	0x00030000	Can select, unavailable, pointer to coarse secondary page table, pointer to fine secondary page table. Sections are unavailable.
0xFEFXXXXX - 0x801XXXXX	4079 - 2049	0x800C3FBC through 0x800C2004	0x00000000	0x00000000	These entries are NEVER available and always return zeroes when read.
0x800XXXXX	2048 (0x800)	0x800C2000	Never points to secondary page table	Never points to secondary page table	Always available, V==R section covering PIO registers at 0x800XXXXX. Domain, AP and CB can be specified.
0x7FFXXXXX - 0x001XXXXX	2047-1	0x800C1FFC through 0x800C0004	0x00000000	0x00000000	These entries are NEVER available and always return zeroes when read.
0x000XXXXX	0	0x800C0000	Never points to secondary page table	Never points to secondary page table	Can select unavailable or V==R section at 0x00000000. Domain, AP and CB bits not available. This is the power on default.

6.2. 16-Megabyte Page-Mapped Virtual Memory (0xFFXXXXXX)

There are 16 1-Mbyte entries in the default first-level page table that can point to second-level page tables. This makes them available for use in paged virtual memory applications. Each time an entry is enabled as a pointer to second-level page table, it consumes either a 1-Kbyte or 4-Kbyte chunk of on-chip SRAM at a hard-wired location in the SRAM. For example, entry 4095 points to the top-most 1-Kbyte or 4-Kbyte block of on-chip SRAM. Most of the entries return 0x00000000. This is true for entries 0001 through 2047 and entries 2049 through 4093.

6.2.1. Default First-Level Page Table Entry 4095

The last entry in the default first-level page table is designed to allow the 1-Mbyte region containing the on-chip ROM image to be mapped “virtual equal real.” This allows the ROM to be accessed directly when the MMU is first turned on. Alternatively, it can be set to point to either a coarse or fine second-level page table. Only a subset of the 32 bits are actually writable. The pointers are hardwired to the top of on-chip SRAM and depend on the setting of bits [1:0]. At power on, it reads back 0xFFF00C12.

DFLPT_ENTRY4095 0x800C3FFC

Table 97. First-Level Page Table Entry 4095 (0xFFF00000 -0xFFFFFFFF) @ 0x800C3FFC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0x00000000																															
Pointer to Coarse Page Table @ 0x0003FC00 (this bit field reads back 0x0003F)																AP =11		0		DOMAIN				1		CB =00				01	
Virtual == Real Section 0xFFF0000																AP				DOMAIN						C	B			10	
Pointer to Fine Page Table @ 0x0003F000 (this bit field reads back 0x0003F)																AP =00				DOMAIN						CB =00				11	

.

Table 98. DFLPT_ENTRY4095 Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	**POINTER**	RO	0xFFF00	For section, points to 0xFFFXXXXX. For fine page table, points to 0x0003F000 For coarse page table, points to 0x0003FC00
11:10	**AP**	RW	0x3	For section, set to 0x3, allowing ALL accesses or other value as desired. For coarse page table, always set to 0x3. For fine page table, always set to 0x0
9	**ALWAYS_ZERO**	RO	0x0	Always reads back a zero.
8:5	**DOMAIN**	RW	0x0	For section, set as desired. For secondary page pointers, set to zero.
4	**ALWAYS_ONE**	RO	0x1	Always reads back a one, as required in 926 TRM
3	**CACHE**	RW	0x0	For section, set to desired calculability. For secondary page table pointers, set to zero.
2	**BUFFER**	RW	0x0	For section, set to desired buffer ability. For secondary page table pointers, set to zero.
1:0	**FIRST_LEVEL**	RW	0x2	For sections, set to 0x2. For coarse page table pointer, set to 0x1. For fine page table pointer, set to 0x3 To mark the regions unavailable, set to 0x0.

6.2.2. Default First-Level Page Table Entries 4094–4080

In a similar fashion, 15 additional 1-Mbyte regions at the top of virtual memory can be enabled to point to second-level page tables. Again, these registers only have a subset of the bits that can be written. For each first-level page table entry, there is a definite hardwired address that is pointed to when the entry is enabled as a coarse or fine page table pointer.

DFLPT_ENTRY4094 0x800C3FF8
DFLPT_ENTRY4093 0x800C3FF4
DFLPT_ENTRY4092 0x800C3FF0
DFLPT_ENTRY4091 0x800C3FEC
DFLPT_ENTRY4090 0x800C3FE8
DFLPT_ENTRY4089 0x800C3FE4
DFLPT_ENTRY4088 0x800C3FE0
DFLPT_ENTRY4087 0x800C3FDC
DFLPT_ENTRY4086 0x800C3FD8
DFLPT_ENTRY4085 0x800C3FD4
DFLPT_ENTRY4084 0x800C3FD0
DFLPT_ENTRY4083 0x800C3FCC
DFLPT_ENTRY4082 0x800C3FC8
DFLPT_ENTRY4081 0x800C3FC4
DFLPT_ENTRY4080 0x800C3FC0

Table 99. First-Level Page Table Entry 4094–Entry 4080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0x00000000																															
Bits [31:9] of Coarse Page Table, see Table 96																							**DOMAIN**				**100**			**01**	
Section cannot be specified, will revert to unavailable (0x00000000)																														**10**	
Bits [31:9] of Fine Page Table, see Table 96																							**DOMAIN**				**100**			**11**	

Table 100. DFLPT_ENTRY4094–DFLPT_ENTRY4080 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:9	**POINTER**	RO	0x0	Bits [31:9] of Coarse or Fine Page Table power, depending on first-level setting.
8:5	**DOMAIN**	RW	0x0	Set as desired.
4:2	**SPECIAL_ARM_VALUE**	RO	0x0	For abort mode (FIRST_LEVEL==00), this field returns a value of 0x0. For coarse page tables (FIRST_LEVEL == 01) or fine page tables, (FIRST_LEVEL == 11) the field returns a value of 0x4.
1:0	**FIRST_LEVEL**	RW	0x0	Set to 0x0 to mark the regions as unavailable (register will read all zeros) Set to 0x1 for coarse page table pointer Set to 0x2, same effect as 0x0 Set to 0x3 for fine page table pointer

6.2.3. Default First-Level Page Table PIO Register Map Entry 2048

The 1-Mbyte PIO region at physical address 0x800XXXXX is mapped “virtual equal real” by the default first-level page table ENTRY_2048, as shown below.

DFLPT_ENTRY_2048 0x800C2000

Table 101. First-Level Page Table Entry 2048 (0x80000000 -0x800FFFFF) @ 0x800C2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Virtual == Real Section, i.e., 0x80000																				AP		0	DOMAIN				1	C	B	10	

Table 102. DFLPT_ENTRY2048 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	**POINTER**	RO	0x80000	This section points to 0x80000000 and is always available.
11:10	**AP**	RW	0x3	Initially set to 0x3 for allowing ALL accesses, set to other values as desired.
9	**ALWAYS_ZERO**	RO	0x0	Always reads back a zero.
8:5	**DOMAIN**	RW	0x0	Set as desired.
4	**ALWAYS_ONE**	RO	0x1	Always reads back a one, as required in 926 TRM
3	**CACHE**	RW	0x0	Set to desired cachability.
2	**BUFFER**	RW	0x0	Set to desired bufferability.
1:0	**FIRST_LEVEL**	RO	0x2	Always reads back 0x2 for section descriptor.

6.2.4. Default First-Level Page Table Entry 0000 V==R SRAM Access

Finally, the SRAM at physical address 0x0000000 can be mapped “virtual equal real”, if desired. When enabled, this location reads back as a first-level page table section descriptor with a 32-bit value of 0x00000C12. When disabled, it reads back 0x0000000 to abort any access attempts.

DFLPT_ENTRY_0000 0x800C0000

Table 103. First-Level Page Table Entry 0000 (0x00000000 -0x000FFFFF) @ 0x800C0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0x00000000																															
Virtual == Real Section, i.e., 0x00000C12																														**SECTION**	**0**

Table 104. DFLPT_ENTRY0000 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**POINTER**	RO	0x0x0000304	For section, points to 0x000000000. with AP==3, DOMAIN==0, always_one, Non cached, Non-Buffered. Otherwise unavailable (returns all zeroes).
1	**SECTION**	RW	0x1	Set to one to enable SRAM at address 0X00000000. If set to zero, the address returns 0X00000009.
0	**ALWAYS_ZERO**	RO	0x0	Always reads back a zero.

7. DIGITAL CONTROL AND ON-CHIP RAM

This chapter describes the digital control block and the on-chip RAM features of the STMP36xx. It includes sections on controlling the SRAM, ROM, performance monitors, high-entropy pseudo-random number seed, and free-running microseconds counter. Programmable registers for the block are described in Section 7.6.

7.1. Overview

The digital control block provides overall control of various items within the top digital block of the chip, including the on-chip RAM controls, default page-table controls, and HCLK performance counter, as shown in Figure 20.

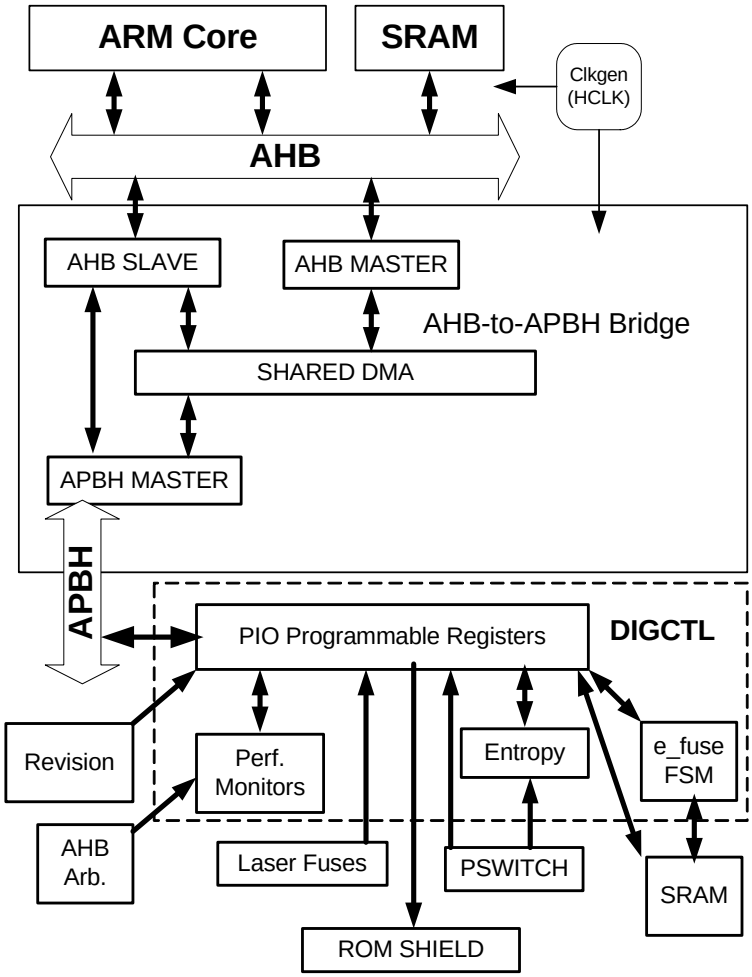


Figure 20. Digital Control (DIGCTL) Block Diagram

The on-chip RAM is constructed from an array of six-transistor dynamic RAM bit cells. The repair functions of this SRAM are controlled by registers in the DIGCTL block.

7.2. SRAM Controls

The on-chip RAM is based on a six-transistor dynamic RAM cell. It is implemented in four segments of 64 Kbytes each (4 by 16Kx32), as shown in Figure 21.

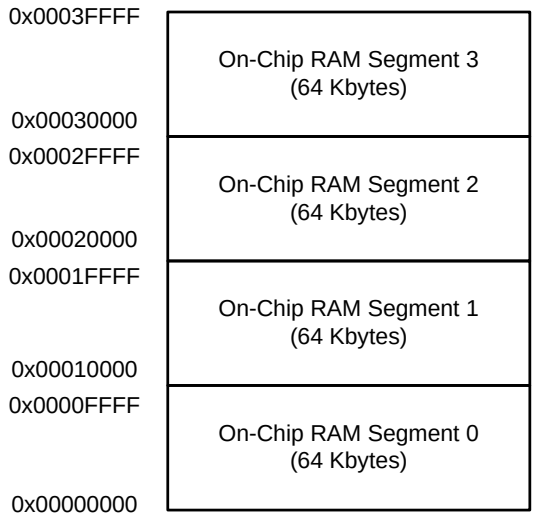


Figure 21. On-Chip RAM Partitioning

A 32-bit AHB address is converted to an SRAM macrocell address, as shown in Table 105.

Table 105. On-Chip RAM Address Bits (within Macrocell)

AHB ADDR BITS	MACRO CELL BITS	USAGE	DESCRIPTION
71:16	15:14	**SEGMENT ADDRESS**	Selects one 64-Kbyte segment
15:11	13:9	**ROW ADDRESS**	Selects a row in the array
10:7	8:5	**COLUMN ADDRESS**	Selects a column in the array
6:2	4:0	**BANK ADDRESS**	Selects a bank in the array

Accessing on-chip RAM over the bus requires only one initial wait state for arbitration.

The on-chip RAM includes some redundancy for RAM repair. Each segment contains two spare columns that can be substituted for failures. The macrocell contains eight 7-bit e_fuse registers that control the repair circuitry. The macrocell documentation refers to these as e_fuse registers, because it was originally designed to work with electric fused repair information. In this application, the repair information is stored in conventional flip-flops by firmware. The interface to the e_fuse repair registers is serial. A state machine in the DIGCTL block shifts the 56 bits of repair data into the on-chip RAM macrocell. Software runs the BIST algorithm hardware and determines the proper corrections. It loads the repair information into the HW_DIGCTL_RAMREPAIR0 and HW_DIGCTL_RAMREPAIR1 registers and sets HW_DIGCTL_RAMCTRL_REPAIR_TRANSMIT. Software must wait until

HW_DIGCTL_RAMCTRL_REPAIR_STATUS returns to zero before attempting to use the on-chip RAM. See Figure 22.

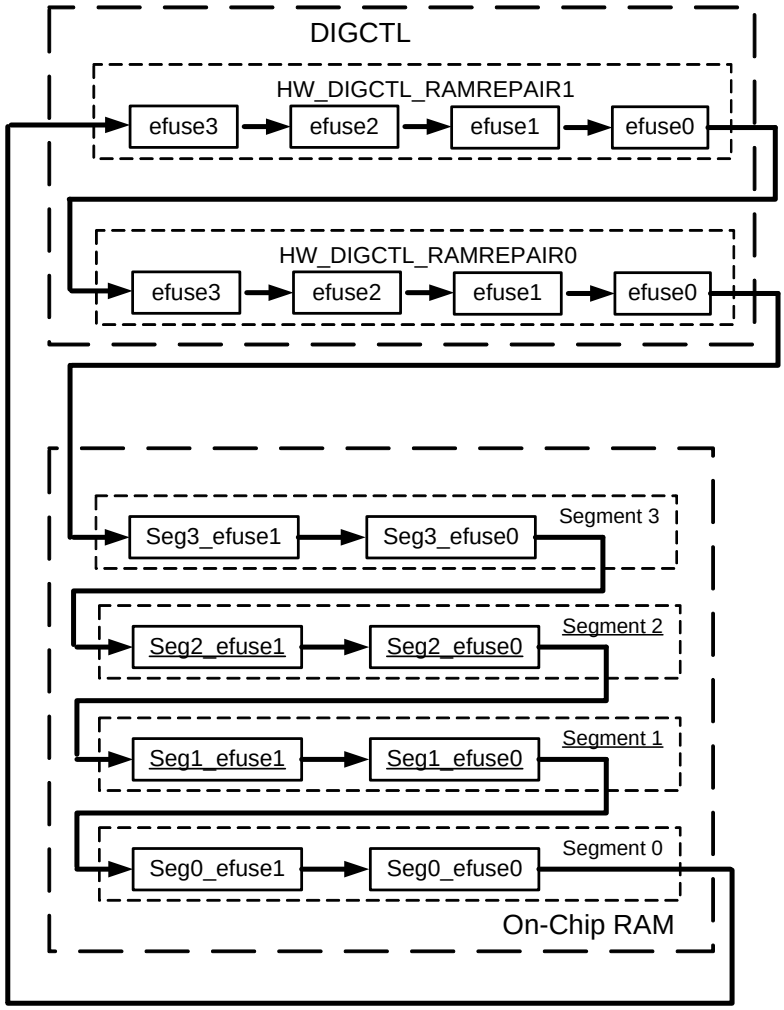


Figure 22. On-Chip RAM E_Fuse Control

To substitute a spare column, a 7-bit e_fuse register is loaded with the defective bank and column number and the valid is set, as shown in Table 106. Once this information has been shifted into the on-chip RAM macrocell, then the redundant column is used for subsequent accesses.

Table 106. E_Fuse Control for One 64-Kbyte Bank of On-Chip RAM

BITS	LABEL	DEFINITION
6	**E_FUSE_VALID**	Set to one to mark a valid E_FUSE entry.
5:1	**DEFECTIVE_BANK**	Address[4:0] of the defective locations.
0	**DEFECTIVE_COLUMN**	Address[5] of the defective locations.

7.2.1. SRAM BIST Control

The SRAM has a Built-In Self-Test (BIST) engine that tests RAM using algorithms defined by Mosys, the supplier for the IP. The BIST performs a 10N test using row fast addressing, followed by a 10N test in column fast addressing, followed by a retention test using row fast addressing. The 10N tests use 0 and F data, and the retention test uses 5 and A data.

The SRAM has four blocks, each of which contain 32 banks and one redundant bank. The redundant bank can be used to repair the RAM. There are two sub-blocks in each block that can be switched in for repair, for a total of eight values that can be shifted into the RAM to repair eight sub-blocks.

The SRAM BIST engine tests the RAM and stores information on eight unique failing addresses that can be used for the repair, two for each block in registers in the DIGCTL block. These are the HW_DIGCTL_1TBIST_REPAIR0 and HW_DIGCTL_1TBIST_REPAIR1 registers. This data can be transferred directly to the HW_DIGCTL_RAMREPAIR0 and HW_DIGCLT_RAMREPAIR1 registers and transferred to the SRAM for the repair.

The SRAM BIST operation is started by setting the BIST_START bit in the HW_DIGCTL_1TBIST_CSR register. This starts the BIST operation, and, when completed, the BIST_DONE signal in this register is set. This register also contains the results of the BIST in the BIST_PASS and BIST_FAIL bits.

Additional information on the fails is provided in the bits shown in Table 107.

Table 107. BIST Fail Table

BIT	DESCRIPTION
FAIL_BLOCK_0_0	Set for the second fail in block 0
FAIL_BLOCK_1_0	Set for the first fail in block 1
FAIL_BLOCK_1_1	Set for the second fail in block 1
FAIL_BLOCK_2_0	Set for the first fail in block 2
FAIL_BLOCK_2_1	Set for the second fail in block 2
FAIL_BLOCK_3_0	Set for the first fail in block 3
FAIL_BLOCK_3_1	Set for the second fail in block 3

The 14 status registers containing fail information are listed in Table 108.

Table 108. BIST Fail Register Information

REGISTER	INFORMATION
HW_DIGCTL_1TBIST_STATUS0	Contains fail data for fail1 of block 0
HW_DIGCTL_1TBIST_STATUS1	Contains fail data for fail2 of block 0
HW_DIGCTL_1TBIST_STATUS2	Contains fail data for fail1 of block 1
HW_DIGCTL_1TBIST_STATUS3	Contains fail data for fail2 of block 1
HW_DIGCTL_1TBIST_STATUS4	Contains fail data for fail1 of block 2
HW_DIGCTL_1TBIST_STATUS5	Contains fail data for fail2 of block 2
HW_DIGCTL_1TBIST_STATUS6	Contains fail data for fail1 of block 3
HW_DIGCTL_1TBIST_STATUS7	Contains fail data for fail2 of block 3

Table 108. BIST Fail Register Information (Continued)

REGISTER	INFORMATION
HW_DIGCTL_1TBIST_STATUS8	Contains fail address of fail 1 and 2 for block 0
HW_DIGCTL_1TBIST_STATUS9	Contains fail address of fail 1 and 2 for block 1
HW_DIGCTL_1TBIST_STATUS10	Contains fail address of fail 1 and 2 for block 1
HW_DIGCTL_1TBIST_STATUS11	Contains fail address of fail 1 and 2 for block 3
HW_DIGCTL_1TBIST_STATUS12	Contains the state in which fail occurred for fails 1 and 2 of blocks 0 and 1
HW_DIGCTL_1TBIST_STATUS13	Contains the state in which fail occurred for fails 1 and 2 of blocks 2 and 3

This data can be used for debug and analysis.

7.3. ROM Controls

The on-chip ROM contains a shielded 2-Kbyte area, 0xFFFF0800–0xFFFF0FFF, that is used to hold various decryption and authentication keys used by the boot loader to certify a legal boot image into the trust zone. This area can be *shielded* from view by writing to the HW_DIGCTL_ROMSHIELD_WRITE_ONCE bit. This shields the keys from further reading. The bit is a write-once operation, i.e., the ROM cannot be unshielded until the next chip-wide reset event.

NOTE: The shield is also raised, automatically, when the JTAG debugger is detected, as evidenced by a number of JTAG clock rising-edges being detected.

7.4. Miscellaneous Controls

The digital control block also contains a number of other miscellaneous functions, as detailed in this section.

7.4.1. Performance Monitoring

The digital control block contains several registers for performance monitoring, including HW_DIGCTL_HCLKCOUNT, which counts HCLK rising edges. This register counts at a variable rate as the HW_CLKCTRL_HBUSCLKCTRL_AUTO_SLOW _DOWN is enabled.

The HW_DIGCTL_AHBSTALLED and HW_DIGCTL_AHBCYCLES registers can be used to measure AHB bus utilization. The stalled register counts all cycles in which any device has an outstanding and unfulfilled bus operation in flight. The cycles register counts the number of data transfer cycles. Subtract cycles from stalls to determine under utilized bus cycles. These counters can be used to tune the performance of the HCLK frequency for specific activities. In addition, these monitors can be focus on specific masters. See the HW_DIGCTL_CTRL _MASTER_SELECT bit description, for example.

7.4.2. High-Entropy PRN Seed

A 32-bit entropy register begins running a pseudo-random number algorithm from the time reset is removed until the PSWITCH is released by the user. This high-entropy value can be used as the seed for other pseudo-random number generators.

7.4.3. Write-Once Register

A 32-bit write-once register holds a runtime-derived locked seed. Once written, it cannot be changed until the next chip wide reset event. The contents of this register are frequently derived from the entropy register.

7.4.4. Microseconds Counter

A 32-bit free-running microseconds counter provides fine-grain real-time control. Its period is determined by dividing the 24.0-MHz crystal oscillator by 24. Thus, its frequency does not change as HCLK, XCLK, and the processor clock frequency are changed.

7.5. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

7.6. Programmable Registers

The following registers provide control of all programmable elements of the digital control block.

7.6.1. DIGCTL Control Register Description

The DIGCTL Control Register provides overall control of various functions throughout the digital portion of the chip.

HW_DIGCTL_CTRL 0x8001C000
HW_DIGCTL_CTRL_SET 0x8001C004
HW_DIGCTL_CTRL_CLR 0x8001C008
HW_DIGCTL_CTRL_TOG 0x8001C00C

Table 109. HW_DIGCTL_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3				MASTER_SELECT				RSVD2			USB_TESTMODE	ANALOG_TESTMODE	DIGITAL_TESTMODE	UTMI_TESTMODE	UART_LOOPBACK	RSVD1												DEBUG_DISABLE	USB_CLKGATE	JTAG_SHIELD	PACKAGE_SENSE_ENABLE

Table 110. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD3**	RO	0x0	Always write zeroes to this bit field.
28:24	**MASTER_SELECT**	RW	0x0	Set various bits of this bit field to one to enable performance monitoring in the AHB arbiter for the corresponding AHB master. ARM_I = 0x01 Select ARM I Master. ARM_D = 0x02 Select ARM D Master. APBH = 0x04 Select APBH DMA Master. APBX = 0x08 Select APBX DMA Master. USB = 0x10 Select USB Master.
23:21	**RSVD2**	RO	0x0	Always write zeroes to this bit field.
20	**USB_TESTMODE**	RW	0x0	Reserved. Always write a 0 to this bit field.
19	**ANALOG_TESTMODE**	RW	0x0	Reserved. Always write a 0 to this bit field.
18	**DIGITAL_TESTMODE**	RW	0x0	Reserved. Always write a 0 to this bit field.
17	**UTMI_TESTMODE**	RW	0x0	Reserved. Always write a 0 to this bit field.
16	**UART_LOOPBACK**	RW	0x0	Set this bit to one to loop the two UARTs back on themselves in a null modem configuration. NORMAL = 0x0 No loopback. LOOPIT = 0x1 Loop the debug UART and the application UART together.
15:4	**RSVD1**	RO	0x0	Always write zeroes to this bit field.
3	**DEBUG_DISABLE**	RW	0x0	Set this bit to one to disable the ARM core's debug logic (for power savings). This bit must remain zero following power-on reset for normal JTAG debugger operation of the ARM core. When set to one, it gates off the clocks to the ARM core's debug logic. Once this bit is set, the part must undergo a power-on reset to re-enable debug operation. Manually clearing this bit via a write after it has been set produces unknown results.
2	**USB_CLKGATE**	RW	0x1	This bit must be set to zero for normal operation of the USB controller. When set to one, it gates off the clocks to the USB controller. RUN = 0x0 Allow USB to operate normally. NO_CLKS = 0x1 Do not clock USB gates in order to minimize power consumption.
1	**JTAG_SHIELD**	RW	0x1	This bit is set to one by laser fuse to disable the JTAG debugger during boot ROM execution. It is set to zero at the end of boot ROM execution, just before branching to the loaded code. NORMAL = 0x0 JTAG debugger enabled. SHIELDS_UP = 0x1 JTAG debugger disabled.
0	**PACKAGE_SENSE_ENABLE**	RW	0x0	Set this bit to one to enable the pullup resistor on the package-type sense pad. This pad is floating in 100-pin packages; therefore turning on the pullup will cause the PACKAGE_TYPE sensor to read back a one. The pad is bonded to ground in 169-pin packages so that a zero is read back by the sensor. DISABLE = 0x0 Disable the package-sense pullup resistor. ENABLE = 0x1 Enable the package-sense pullup resistor.

DESCRIPTION:

This register controls various functions throughout the digital portion of the chip.

EXAMPLE:

```
HW_DIGCTL_CTRL_CLR(BM_DIGCTL_CTRL_USB_CLKGATE); // enable USB clock
```

7.6.2. DIGCTL Status Register Description

The DIGCTL Status Register reports status for the digital control block.

HW_DIGCTL_STATUS 0x8001C010
HW_DIGCTL_STATUS_SET 0x8001C014
HW_DIGCTL_STATUS_CLR 0x8001C018
HW_DIGCTL_STATUS_TOG 0x8001C01C

Table 111. HW_DIGCTL_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ROM_KEYS_PRESENT	RSVD1																								JTAG_SHIELD_DEFAULT	ROM_SHIELDED	JTAG_IN_USE	PSWITCH		PACKAGE_TYPE	WRITTEN

Table 112. HW_DIGCTL_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**ROM_KEYS_PRESENT**	RO	0x1	This read-only bit field returns a one if the ROM Key Set is available. Otherwise, it returns a zero.
30:7	**RSVD1**	RO	0x0	Reserved.
6	**JTAG_SHIELD_DEFAULT**	RO	0x0	This read-only bit is a one if the JTAG shield default all layer change bit is a one.
5	**ROM_SHIELDED**	RO	0x0	This read-only bit is a one if the ROM shield is raised so that the last 2K bytes cannnot be read.
4	**JTAG_IN_USE**	RO	0x0	This read-only bit is a one if JTAG debugger usage has been detected.
3:2	**PSWITCH**	RO	0x0	These read-only bits reflect the current state of the pswitch comparators.
1	**PACKAGE_TYPE**	RO	0x0	This read-only bit returns a one in 100-pin packages. It reads back a zero in 169-pin packages.
0	**WRITTEN**	RO	0x0	Set to one by any successful write to the HW_WRITEONCE register.

DESCRIPTION:

The status register provies a read-only view to various input conditions and internal states.

EXAMPLE:

```
if(HW_DIGCTL_STATUS.PACKAGE_TYPE)
```

```
{
// do 100-pin package things
}
```

7.6.3. Free-Running HCLK Counter Register Description

This free-running counter is available for performance metrics.

HW_DIGCTL_HCLKCOUNT 0x8001C020
HW_DIGCTL_HCLKCOUNT_SET 0x8001C024
HW_DIGCTL_HCLKCOUNT_CLR 0x8001C028
HW_DIGCTL_HCLKCOUNT_TOG 0x8001C02C

Table 113. HW_DIGCTL_HCLKCOUNT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 114. HW_DIGCTL_HCLKCOUNT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RO	0x0	This counter counts up from reset using HCLK.

DESCRIPTION:

This counter increments once per HCLK rising edge.

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT;
// Do something you want timed here
EndTime = HW_DIGCTL_HCLKCOUNT;
Duration = EndTime - StartTime; // make sure to handle rollover in a real application
```

7.6.4. On-Chip RAM Control Register Description

The On-Chip RAM Control Register holds on-chip SRAM control bit fields.

HW_DIGCTL_RAMCTRL 0x8001C030
HW_DIGCTL_RAMCTRL_SET 0x8001C034
HW_DIGCTL_RAMCTRL_CLR 0x8001C038
HW_DIGCTL_RAMCTRL_TOG 0x8001C03C

Table 115. HW_DIGCTL_RAMCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD5	TEST_MARGIN			PWDN_BANKS				RSVD4	TEMP_SENSOR			RSVD3	TEST_TEMP_COMP			RSVD2	SHIFT_COUNT							FLIP_CLK	RSVD1			OVER_RIDE_TEMP	REF_CLK_GATE	REPAIR_STATUS	REPAIR_TRANSMIT

Table 116. HW_DIGCTL_RAMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD5**	RO	0x0	Reserved.
30:28	**TEST_MARGIN**	RW	0x0	Set these bits to various test margin levels to the SRAM TLS bits. NORMAL = 0x0 Normal Operation. LEVEL1 = 0x1 Test Mode level 1. LEVEL2 = 0x2 Test Mode level 2. LEVEL3 = 0x3 Test Mode level 3. LEVEL4 = 0x4 Test Mode level 4. LEVEL5 = 0x5 Test Mode level 5. LEVEL6 = 0x6 Test Mode level 6. LEVEL7 = 0x7 Test Mode level 7.
27:24	**PWDN_BANKS**	RW	0x0	Powers down the SRAM banks. Each bit powers down 64KB of SRAM. PWDN_BANK3 = 0x8 Set to one to power down bank3, i.e., 0x00030000 through 0x0003FFFF. PWDN_BANK2 = 0x4 Set to one to power down bank2, i.e., 0x00020000 through 0x0002FFFF. PWDN_BANK1 = 0x2 Set to one to power down bank1, i.e., 0x00010000 through 0x0001FFFF. PWDN_BANK0 = 0x1 Set to one to power down bank0, i.e., 0x00000000 through 0x0000FFFF.
23	**RSVD4**	RO	0x0	Reserved.
22:20	**TEMP_SENSOR**	RO	0x7	Three-bit temperature code from the on-chip temperature sensor. This value can be automatically copied into TEST_TEMP_COMP
19	**RSVD3**	RO	0x0	Reserved.
18:16	**TEST_TEMP_COMP**	RW	0x7	Temperature compensation for RAM repair. 0=Normal Mode (default). During RAM test and repair, the die temperature must be written to this field. Temperature is determined using on-chip temperature sensor (see LRADC). LOW_TEMP = 0x1 Temperature less than 15C. RANGE_A = 0x2 Temperature 15C to 25C. RANGE_B = 0x3 Temperature 25C to 35C. RANGE_C = 0x4 Temperature 35C to 45C. RANGE_D = 0x5 Temperature 45C to 55C. RANGE_E = 0x6 Temperature 55C to 70C. RANGE_F = 0x7 Temperature great than 70C.
15	**RSVD2**	RO	0x0	Reserved.
14:8	**SHIFT_COUNT**	RO	0x0	This read-only bit field reads back the state of the shift counter. The LSB toggles to generate an efuse_clk.
7	**FLIP_CLK**	RW	0x0	Use the opposite edge for efuse_clk. NORMAL = 0x0 Normal rising edge. INVERT = 0x1 Inverted, i.e., falling edge.
6:4	**RSVD1**	RO	0x0	Reserved.
3	**OVER_RIDE_TEMP**	RW	0x0	Normally, the three-bit hardware temperature sensor value is copied into the TEST_TEMP_CODE register automatically. Set to one to override the copying of the on-chip temperature sensor value into the TEST_TEMP_COMP bit field. The value in TEST_TEMP_COMP always drives the temperature inputs to the on-chip RAM. NORMAL = 0x0 Normal operation, provide hardware temperature sensor value to on-chip RAM (default). OVER_RIDE = 0x1 Firmware-supplied value in TEST_TEMP_COMP is not modified by hardware.

Table 116. HW_DIGCTL_RAMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**REF_CLK_GATE**	RW	0x0	Gate the 32-kHz Reference Clock. This should be left at 0. NORMAL = 0x0 Normal operation, provide reference clock to the macro (default). OFF = 0x1 Turn off the refresh clock.
1	**REPAIR_STATUS**	RO	0x0	SRAM Repair Transmission in Progress. Do not access the SRAM while this bit is set. IDLE = 0x0 E_fuse transfer complete. BUSY = 0x1 E_fuse transer in progress.
0	**REPAIR_TRANSMIT**	RW	0x0	Transmit Repair Data to On-Chip RAM. Serially sends the RAM repair data to the on-chip RAM. The on-chip RAM should not be accessed while the repair data is being transmitted. IDLE = 0x0 No transfer. SEND = 0x1 Send E_fuse data serially to the on-chip RAM.

DESCRIPTION:

This register controls various parts of the on-chip RAM, including the repair state machine that shifts the repair configuration data into the SRAM macro-cell.

EXAMPLE:

```
HW_DIGCTL_RAMCTRL_SET(BM_DIGCTL_RAMCTRL_REPAIR_TRANSMIT); // Start the efuse state machine
```

7.6.5. On-Chip RAM Repair Data 0 Register Description

The On-Chip RAM Repair Data 0 Register holds repair data for the on-chip SRAM.

HW_DIGCTL_RAMREPAIR0 0x8001C040
HW_DIGCTL_RAMREPAIR0_SET 0x8001C044
HW_DIGCTL_RAMREPAIR0_CLR 0x8001C048
HW_DIGCTL_RAMREPAIR0_TOG 0x8001C04C

Table 117. HW_DIGCTL_RAMREPAIR0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4	EFUSE3							RSVD3	EFUSE2							RSVD2	EFUSE1							RSVD1	EFUSE0						

Table 118. HW_DIGCTL_RAMREPAIR0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD4**	RO	0x0	Reserved, always set to zero.
30:24	**EFUSE3**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 1 efuse 1. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.
23	**RSVD3**	RO	0x0	Reserved, always set to zero.

Table 118. HW_DIGCTL_RAMREPAIR0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22:16	**EFUSE2**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 1 efuse 0. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.
15	**RSVD2**	RO	0x0	Reserved, always set to zero.
14:8	**EFUSE1**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 0 efuse 1. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.
7	**RSVD1**	RO	0x0	Reserved, always set to zero.
6:0	**EFUSE0**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 0 efuse 0. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.

DESCRIPTION:

This register contains the efuse repair configuration information that can be shifted into the lower two 64-Kbyte banks of the on-chip RAM.

EXAMPLE:

```
HW_DIGCTL_RAMREPAIR0.EFUSE0= 0x2A; // read modify write is ok
```

7.6.6. On-Chip RAM Repair Data 1 Register Description

The On-Chip RAM Repair Data 1 Register holds repair data for the on-chip SRAM

HW_DIGCTL_RAMREPAIR1 0x8001C050
HW_DIGCTL_RAMREPAIR1_SET 0x8001C054
HW_DIGCTL_RAMREPAIR1_CLR 0x8001C058
HW_DIGCTL_RAMREPAIR1_TOG 0x8001C05C

Table 119. HW_DIGCTL_RAMREPAIR1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4	EFUSE3							RSVD3	EFUSE2							RSVD2	EFUSE1							RSVD1	EFUSE0						

Table 120. HW_DIGCTL_RAMREPAIR1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD4**	RO	0x0	Reserved, always set to zero.
30:24	**EFUSE3**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 3 efuse 1. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.
23	**RSVD3**	RO	0x0	Reserved, always set to zero.
22:16	**EFUSE2**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 3 efuse 0. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.
15	**RSVD2**	RO	0x0	Reserved, always set to zero.
14:8	**EFUSE1**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 2 efuse 1. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.
7	**RSVD1**	RO	0x0	Reserved, always set to zero.
6:0	**EFUSE0**	RW	0x0	SRAM Repair efuse register bits 6 through 0 for segment 2 efuse 0. This data and the DATA field in SRAM_REPAIR1 are shifted to the SRAM controller when the REPAIR_TRANSMIT bit in SRAM_CTRL is set.

DESCRIPTION:

This register contains the efuse repair configuration information that can be shifted into the upper two 64-Kbyte banks of the on-chip RAM.

EXAMPLE:

```
HW_DIGCTL_RAMREPAIR1.EFUSE0= 0x37; // read modify write is ok
```

7.6.7. Software Write-Once Register Description

The Software Write Once Register hold the value used in software certification management.

HW_DIGCTL_WRITEONCE 0x8001C060

Table 121. HW_DIGCTL_WRITEONCE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 122. HW_DIGCTL_WRITEONCE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0xA5A5A5A5	This field can be written only one time. The contents are not used by hardware.

DESCRIPTION:

This register is used to hold a portion of a certificate that is not mutable after software initialization.

EXAMPLE:

```
HW_DIGCTL_WRITEONCE.U = my_certificate;
```

7.6.8. AHB Transfer Count Register Description

The AHB Transfer Count Register counts the number of transfers made on the AHB.

HW_DIGCTL_AHBCYCLES 0x8001C070

Table 123. HW_DIGCTL_AHBCYCLES

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 124. HW_DIGCTL_AHBCYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

DESCRIPTION:

This counter increments on AHB cycles when the arbiter sees HREADY and a master has an active HTRANS code in process, i.e., a clock in which data was read or written between a master and a slave. It ignores cycles in which a master was granted access but the slave was not ready. The master selects in HW_DIGCTL_CTRL_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_AHBCYCLES.COUNT less than 1000000)
{
// wait for a specific number of xfers
}
ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

7.6.9. AHB Performance Metric for Stalled Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

HW_DIGCTL_AHBSTALLED 0x8001C080

Table 125. HW_DIGCTL_AHBSTALLED

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 126. HW_DIGCTL_AHBSTALLED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x0	This field counts the number of AHB cycles in which a master was stalled.

DESCRIPTION:

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_CTRL_MASTER_SELECT are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberStalledCycles = HW_DIGCTL_AHBSTALLED_COUNT_RD();
```

7.6.10. Entropy Register Description

The Entropy register is a read-only test value register.

HW_DIGCTL_ENTROPY 0x8001C090

Table 127. HW_DIGCTL_ENTROPY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 128. HW_DIGCTL_ENTROPY Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0x0	This read-only bit field always reads back the results of an entropy calculation. It is used to randomize the seeds for random number generators.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
while(HW_DIGCTL_STATUS.PSWITCH != 0)
{
//wait for pswitch to go away
}
HW_DIGCTL_WRITEONCE.BITS = rand(HW_DIGCTL_ENTROPY.VALUE);
```

7.6.11. Digital Control ROM Shield Read Enable Register Description

The Digital Control ROM Shield Read Enable Register is a write-once register for disabling key set reads from the on-chip ROM.

HW_DIGCTL_ROMSHIELD 0x8001C0A0

Table 129. HW_DIGCTL_ROMSHIELD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																															WRITE_ONCE

Table 130. HW_DIGCTL_ROMSHIELD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:1	**RSVD1**	RO	0x0	Always write zeroes to this bit field.
0	**WRITE_ONCE**	RW	0x0	Set this bit to one to disable reading the boot loader encryption keys, which occupy ROM addresses 0xFFFF0800 - 0xFFFF0FFF. This bit can written once only. It is written from within ROM code, so that access to the boot loader keys are denied to all loaded code. Any attempt to read this shielded memory will return 0xBEBEBEBE (which represents a debugger breakpoint).

DESCRIPTION:

The ROM Shield is raised before the boot loader loads external code. It is also automatically raised when the JTAG debugger is detected.

EXAMPLE:

```
HW_DIGCTL_ROMSHIELD_SET(BM_DIGCTL_ROMSHIELD_SET_WRITE_ONCE);
```

7.6.12. Digital Control Microseconds Counter Register Description

The Digital Control Microseconds Counter Register is a read-only test value register.

HW_DIGCTL_MICROSECONDS 0x8001C0B0
HW_DIGCTL_MICROSECONDS_SET 0x8001C0B4
HW_DIGCTL_MICROSECONDS_CLR 0x8001C0B8
HW_DIGCTL_MICROSECONDS_TOG 0x8001C0BC

Table 131. HW_DIGCTL_MICROSECONDS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 132. HW_DIGCTL_MICROSECONDS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RW	0x0	This register maintains a 32-bit counter that increments at a one-microsecond rate. The 1-MHz clock driving this counter is derived from the 24.0-MHz crystal osillator. The count value is not preserved over power downs. The 32-bit value wraps in less than two hours. Note that the Digital Control Microseconds Counter Register does not reliably increment when HCLK is set to less than 3 MHz.

DESCRIPTION:

This fixed-rate timer always increments at 24.0 MHz divided by 24 or 1.0 MHz. It does not generate an interrupt.

EXAMPLE:

```
StartTime = HW_DIGCTL_MICROSECONDS_RD();
EndTime = HW_DIGCTL_MICROSECONDS_RD();
ElapsedTime = StartTime - EndTime; // WARNING, handle rollover in real software
```

7.6.13. Digital Control Debug Read Test Register Description

The Digital Control Debug Read Test Register is a read-only test value register.

HW_DIGCTL_DBGRD 0x8001C0C0

Table 133. HW_DIGCTL_DBGRD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COMPLEMENT																															

Table 134. HW_DIGCTL_DBGRD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COMPLEMENT**	RO	0x789ABCDE	This read-only bit field always reads back the ones complement of the value in HW_DIGCTL_DBG.

DESCRIPTION:

This register is used for debugging purposes.

EXAMPLE:

```
debug_value = HW_DIGCTL_DBGRD_RD();
```

7.6.14. Digital Control Debug Register Description

The Digital Control Debug Register is a read-only test value register.

HW_DIGCTL_DBG 0x8001C0D0

Table 135. HW_DIGCTL_DBG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 136. HW_DIGCTL_DBG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RO	0x87654321	This read-only bit field always reads back the fixed value 0x87654321.

DESCRIPTION:

This register is used for debugging purposes.

EXAMPLE:

```
debug_value = HW_DIGCTL_DBG_RD();
```

7.6.15. SRAM BIST Control and Status Register Description

The SRAM BIST Control and Status Register provides overall control of the integrated BIST engine.

HW_DIGCTL_1TRAM_BIST_CSR 0x8001C0E0
HW_DIGCTL_1TRAM_BIST_CSR_SET 0x8001C0E4
HW_DIGCTL_1TRAM_BIST_CSR_CLR 0x8001C0E8
HW_DIGCTL_1TRAM_BIST_CSR_TOG 0x8001C0EC

Table 137. HW_DIGCTL_1TRAM_BIST_CSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																												FAIL	PASS	DONE	START

Table 138. HW_DIGCTL_1TRAM_BIST_CSR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	**RSVD0**	RO	0x0	Reserved
3	**FAIL**	RO	0x0	BIST has FAILED
2	**PASS**	RO	0x0	BIST has PASSED
1	**DONE**	RO	0x0	BIST has completed
0	**START**	RW	0x0	Initiate BIST of internal memory when high

DESCRIPTION:

This register is used to start off the BIST operation on two RAMS in the DMA block. The status signals are returned after the BIST operation is completed to this register.

EXAMPLE:

```
To start the BIST operation, set HW_DIGCTL_1TRAM_BIST_CSR = 0x00000001.
After the BIST is completed and the test passes, the contents of
HW_DIGCTL_1TRAM_BIST_CSR will be 0x00000007, as the DONE and PASS flags
will be set.
```

7.6.16. SRAM BIST Repair Register 0 Description

The SRAM BIST Repair Register 0 contains the repair data for blocks 0 and 1

HW_DIGCTL_1TRAM_BIST_REPAIR0 0x8001C0F0

HW_DIGCTL_1TRAM_BIST_REPAIR0_SET 0x8001C0F4
HW_DIGCTL_1TRAM_BIST_REPAIR0_CLR 0x8001C0F8
HW_DIGCTL_1TRAM_BIST_REPAIR0_TOG 0x8001C0FC

Table 139. HW_DIGCTL_1TRAM_BIST_REPAIR0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																															

Table 140. HW_DIGCTL_1TRAM_BIST_REPAIR0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RSVD0**	RO	0x0	This bit will always be set to 0

DESCRIPTION:

This register contains the repair information for blocks 0 and 1 for the SRAM. It needs to be just loaded into the RAM repair register 0.

EXAMPLE:

```
  Register will contain 0x00000000 for no repair.
  Register will contain 0x00000042 for a repair on bank address 1 and col lsb 0, for block 0.
  Register will contain 0x00004942 for a repair to bank address 4 and col lsb 1, for block 0
in addition.
```

7.6.17. SRAM BIST Repair Register 1 Description

The SRAM BIST Repair Register 1 contains the repair data for blocks 2 and 3

HW_DIGCTL_1TRAM_BIST_REPAIR1 0x8001C100
HW_DIGCTL_1TRAM_BIST_REPAIR1_SET 0x8001C104
HW_DIGCTL_1TRAM_BIST_REPAIR1_CLR 0x8001C108
HW_DIGCTL_1TRAM_BIST_REPAIR1_TOG 0x8001C10C

Table 141. HW_DIGCTL_1TRAM_BIST_REPAIR1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																															

Table 142. HW_DIGCTL_1TRAM_BIST_REPAIR1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RSVD0**	RO	0x0	This bit will always be set to 0

DESCRIPTION:

This register contains the repair information for blocks 2 and 3 for the SRAM. It needs to be just loaded into the RAM repair register 1.

EXAMPLE:

```
  Register will contain 0x00000000 for no repair.
  Register will contain 0x00000042 for a repair on bank address 1 and col LSB 0, for block 2.
  Register will contain 0x00004942 for a repair to bank address 4 and col LSB 1, for block 2
in addition.
```

7.6.18. SRAM Status Register 0 Description

The SRAM Status Register 0 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS0 0x8001C110
HW_DIGCTL_1TRAM_STATUS0_SET 0x8001C114
HW_DIGCTL_1TRAM_STATUS0_CLR 0x8001C118
HW_DIGCTL_1TRAM_STATUS0_TOG 0x8001C11C

Table 143. HW_DIGCTL_1TRAM_STATUS0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA00																															

Table 144. HW_DIGCTL_1TRAM_STATUS0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA00**	RO	0x0	This read-only bit field will contain the fail data for the first fail in block 0.

DESCRIPTION:

This register will contain fail data for the first fail in block 0.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS0_RD();
```

7.6.19. SRAM Status Register 1 Description

The SRAM Status Register 1 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS1 0x8001C120
HW_DIGCTL_1TRAM_STATUS1_SET 0x8001C124
HW_DIGCTL_1TRAM_STATUS1_CLR 0x8001C128
HW_DIGCTL_1TRAM_STATUS1_TOG 0x8001C12C

Table 145. HW_DIGCTL_1TRAM_STATUS1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA01																															

Table 146. HW_DIGCTL_1TRAM_STATUS1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA01**	RO	0x0	This read-only bit field will contain the fail data for the second fail in block 0.

DESCRIPTION:

This register will contain fail data for the second fail in block 0.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS1_RD();
```

7.6.20. SRAM Status Register 2 Description

SRAM Status Register 2 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS2 0x8001C130
HW_DIGCTL_1TRAM_STATUS2_SET 0x8001C134
HW_DIGCTL_1TRAM_STATUS2_CLR 0x8001C138
HW_DIGCTL_1TRAM_STATUS2_TOG 0x8001C13C

Table 147. HW_DIGCTL_1TRAM_STATUS2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA10																															

Table 148. HW_DIGCTL_1TRAM_STATUS2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA10**	RO	0x0	This read-only bit field will contain the fail data for the first fail in block 1.

DESCRIPTION:

This register will contain fail data for the first fail in block 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS2_RD();
```

7.6.21. SRAM Status Register 3 Description

RAM Status Register 3 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS3 0x8001C140
HW_DIGCTL_1TRAM_STATUS3_SET 0x8001C144
HW_DIGCTL_1TRAM_STATUS3_CLR 0x8001C148
HW_DIGCTL_1TRAM_STATUS3_TOG 0x8001C14C

Table 149. HW_DIGCTL_1TRAM_STATUS3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA11																															

Table 150. HW_DIGCTL_1TRAM_STATUS3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA11**	RO	0x0	This read-only bit field will contain the fail data for the second fail in block 1.

DESCRIPTION:

This register will contain fail data for the second fail in block 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS3_RD();
```

7.6.22. SRAM Status Register 4 Description

SRAM Status Register 4 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS4 0x8001C150
HW_DIGCTL_1TRAM_STATUS4_SET 0x8001C154
HW_DIGCTL_1TRAM_STATUS4_CLR 0x8001C158
HW_DIGCTL_1TRAM_STATUS4_TOG 0x8001C15C

Table 151. HW_DIGCTL_1TRAM_STATUS4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA20																															

Table 152. HW_DIGCTL_1TRAM_STATUS4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA20**	RO	0x0	This read-only bit field will contain the fail data for the first fail in block 2.

DESCRIPTION:

This register will contain fail data for the first fail in block 2.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS4_RD();
```

7.6.23. SRAM Status Register 5 Description

SRAM Status Register 5 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS5 0x8001C160
HW_DIGCTL_1TRAM_STATUS5_SET 0x8001C164
HW_DIGCTL_1TRAM_STATUS5_CLR 0x8001C168
HW_DIGCTL_1TRAM_STATUS5_TOG 0x8001C16C

Table 153. HW_DIGCTL_1TRAM_STATUS5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA21																															

Table 154. HW_DIGCTL_1TRAM_STATUS5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA21**	RO	0x0	This read-only bit field will contain the fail data for the second fail in block 2.

DESCRIPTION:

This register will contain fail data for the second fail in block 2.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS5_RD();
```

7.6.24. SRAM Status Register 6 Description

SRAM Status Register 6 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS6 0x8001C170
HW_DIGCTL_1TRAM_STATUS6_SET 0x8001C174
HW_DIGCTL_1TRAM_STATUS6_CLR 0x8001C178
HW_DIGCTL_1TRAM_STATUS6_TOG 0x8001C17C

Table 155. HW_DIGCTL_1TRAM_STATUS6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA30																															

Table 156. HW_DIGCTL_1TRAM_STATUS6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA30**	RO	0x0	This read-only bit field will contain the fail data for the first fail in block 3.

DESCRIPTION:

This register will contain fail data for the first fail in block 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS6_RD();
```

7.6.25. SRAM Status Register 7 Description

SRAM Status Register 7 is a read-only fail data register.

HW_DIGCTL_1TRAM_STATUS7 0x8001C180
HW_DIGCTL_1TRAM_STATUS7_SET 0x8001C184
HW_DIGCTL_1TRAM_STATUS7_CLR 0x8001C188
HW_DIGCTL_1TRAM_STATUS7_TOG 0x8001C18C

Table 157. HW_DIGCTL_1TRAM_STATUS7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILDATA31																															

Table 158. HW_DIGCTL_1TRAM_STATUS7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FAILDATA31**	RO	0x0	This read-only bit field will contain the fail data for the second fail in block 3.

DESCRIPTION:

This register will contain fail data for the second fail in block 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS7_RD();
```

7.6.26. SRAM Status Register 8 Description

SRAM Status Register 8 is a read-only fail address register.

HW_DIGCTL_1TRAM_STATUS8 0x8001C190
HW_DIGCTL_1TRAM_STATUS8_SET 0x8001C194
HW_DIGCTL_1TRAM_STATUS8_CLR 0x8001C198
HW_DIGCTL_1TRAM_STATUS8_TOG 0x8001C19C

Table 159. HW_DIGCTL_1TRAM_STATUS8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILADDR01																FAILADDR00															

Table 160. HW_DIGCTL_1TRAM_STATUS8 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**FAILADDR01**	RO	0x0	This read-only bit field will contain the failing address for the second fail in block 0.
15:0	**FAILADDR00**	RO	0x0	This read-only bit field will contain the failing address for the first fail in block 0.

DESCRIPTION:

This register will contain fail data for the first and second failures in block 0.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS8_RD();
```

7.6.27. SRAM Status Register 9 Description

SRAM Status Register 9 is a read-only fail address register.

HW_DIGCTL_1TRAM_STATUS9 0x8001C1A0
HW_DIGCTL_1TRAM_STATUS9_SET 0x8001C1A4
HW_DIGCTL_1TRAM_STATUS9_CLR 0x8001C1A8
HW_DIGCTL_1TRAM_STATUS9_TOG 0x8001C1AC

Table 161. HW_DIGCTL_1TRAM_STATUS9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILADDR11																FAILADDR10															

Table 162. HW_DIGCTL_1TRAM_STATUS9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**FAILADDR11**	RO	0x0	This read-only bit field will contain the failing address for the second fail in block 1.
15:0	**FAILADDR10**	RO	0x0	This read-only bit field will contain the failing address for the first fail in block 1.

DESCRIPTION:

This register will contain fail data for the first second failures in block 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS9_RD();
```

7.6.28. SRAM Status Register 10 Description

SRAM Status Register 10 is a read-only fail address register.

HW_DIGCTL_1TRAM_STATUS10 0x8001C1B0
HW_DIGCTL_1TRAM_STATUS10_SET 0x8001C1B4
HW_DIGCTL_1TRAM_STATUS10_CLR 0x8001C1B8
HW_DIGCTL_1TRAM_STATUS10_TOG 0x8001C1BC

Table 163. HW_DIGCTL_1TRAM_STATUS10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILADDR21																FAILADDR20															

Table 164. HW_DIGCTL_1TRAM_STATUS10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**FAILADDR21**	RO	0x0	This read-only bit field will contain the failing address for the second fail in block 2.
15:0	**FAILADDR20**	RO	0x0	This read-only bit field will contain the failing address for the first fail in block 2.

DESCRIPTION:

This register will contain fail data for the first and second failures in block 2.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS10_RD();
```

7.6.29. SRAM Status Register 11 Description

SRAM Status Register 11 is a read-only fail address register.

HW_DIGCTL_1TRAM_STATUS11 0x8001C1C0
HW_DIGCTL_1TRAM_STATUS11_SET 0x8001C1C4
HW_DIGCTL_1TRAM_STATUS11_CLR 0x8001C1C8

HW_DIGCTL_1TRAM_STATUS11_TOG 0x8001C1CC

Table 165. HW_DIGCTL_1TRAM_STATUS11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FAILADDR31																FAILADDR30															

Table 166. HW_DIGCTL_1TRAM_STATUS11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**FAILADDR31**	RO	0x0	This read-only bit field will contain the failing address for the second fail in block 3.
15:0	**FAILADDR30**	RO	0x0	This read-only bit field will contain the failing address for the first fail in block 3.

DESCRIPTION:

This register will contain fail data for the first and second failures in block 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS11_RD();
```

7.6.30. SRAM Status Register 12 Description

SRAM Status Register 12 is a read-only fail state register.

HW_DIGCTL_1TRAM_STATUS12 0x8001C1D0
HW_DIGCTL_1TRAM_STATUS12_SET 0x8001C1D4
HW_DIGCTL_1TRAM_STATUS12_CLR 0x8001C1D8
HW_DIGCTL_1TRAM_STATUS12_TOG 0x8001C1DC

Table 167. HW_DIGCTL_1TRAM_STATUS12

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3				FAILSTATE11				RSVD2				FAILSTATE10				RSVD1				FAILSTATE01				RSVD0				FAILSTATE00			

Table 168. HW_DIGCTL_1TRAM_STATUS12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD3**	RO	0x0	This field is unused.
28:24	**FAILSTATE11**	RO	0x0	This read-only bit field will contain the failing state for the second fail in block 1.
23:21	**RSVD2**	RO	0x0	This field is unused.

Table 168. HW_DIGCTL_1TRAM_STATUS12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20:16	**FAILSTATE10**	RO	0x0	This read-only bit field will contain the failing state for the first fail in block 1.
15:13	**RSVD1**	RO	0x0	This field is unused.
12:8	**FAILSTATE01**	RO	0x0	This read-only bit field will contain the failing state for the second fail in block 0.
7:5	**RSVD0**	RO	0x0	This field is unused.
4:0	**FAILSTATE00**	RO	0x0	This read-only bit field will contain the failing state for the first fail in block 0.

DESCRIPTION:

This register will contain fail data for the first and second failures in blocks 0 and 1.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS12_RD();
```

7.6.31. SRAM Status Register 13 Description

SRAM Status Register 13 is a read-only fail state register.

HW_DIGCTL_1TRAM_STATUS13 0x8001C1E0
HW_DIGCTL_1TRAM_STATUS13_SET 0x8001C1E4
HW_DIGCTL_1TRAM_STATUS13_CLR 0x8001C1E8
HW_DIGCTL_1TRAM_STATUS13_TOG 0x8001C1EC

Table 169. HW_DIGCTL_1TRAM_STATUS13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3				FAILSTATE31				RSVD2				FAILSTATE30				RSVD1				FAILSTATE21				RSVD0				FAILSTATE20			

Table 170. HW_DIGCTL_1TRAM_STATUS13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD3**	RO	0x0	This field is unused.
28:24	**FAILSTATE31**	RO	0x0	This read-only bit field will contain the failing state for the second fail in block 3.
23:21	**RSVD2**	RO	0x0	This field is unused.
20:16	**FAILSTATE30**	RO	0x0	This read-only bit field will contain the failing state for the first fail in block 3.
15:13	**RSVD1**	RO	0x0	This field is unused.
12:8	**FAILSTATE21**	RO	0x0	This read-only bit field will contain the failing state for the second fail in block 2.

Table 170. HW_DIGCTL_1TRAM_STATUS13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:5	**RSVD0**	RO	0x0	This field is unused.
4:0	**FAILSTATE20**	RO	0x0	This read-only bit field will contain the failing state for the first fail in block 2.

DESCRIPTION:

This register will contain fail data for the first and second failures in blocks 2 and 3.

EXAMPLE:

```
fail_data = HW_DIGCTL_1TRAM_STATUS0_RD();
```

7.6.32. Digital Control Scratch Register 0 Description

Scratch register 0.

HW_DIGCTL_SCRATCH0 0x8001C290

Table 171. HW_DIGCTL_SCRATCH0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PTR																															

Table 172. HW_DIGCTL_SCRATCH0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**PTR**	RW	0x0	Scratch pad register.

DESCRIPTION:

Scratch Pad Register 0.

EXAMPLE:

```
scratch_pad = (*void)HW_DIGCTL_SCRATCH0.PTR;
```

7.6.33. Digital Control Scratch Register 1 Description

Scratch register 1.

HW_DIGCTL_SCRATCH1 0x8001C2A0

Table 173. HW_DIGCTL_SCRATCH1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PTR																															

Table 174. HW_DIGCTL_SCRATCH1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**PTR**	RW	0x0	Scratch pad register.

DESCRIPTION:

Scratch Pad Register 1.

EXAMPLE:

```
scratch_pad = (*void)HW_DIGCTL_SCRATCH1.PTR;
```

7.6.34. Digital Control ARM Cache Register Description

Cache RAM controls.

HW_DIGCTL_ARMCACHE 0x8001C2B0

Table 175. HW_DIGCTL_ARMCACHE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2																						CACHE_SS		RSVD1		DTAG_SS		RSVD0		ITAG_SS	

Table 176. HW_DIGCTL_ARMCACHE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:10	**RSVD2**	RO	0x0	Reserved.
9:8	**CACHE_SS**	RW	0x1	Timing Control for 512x32x2 RAMs (Cache).
7:6	**RSVD1**	RO	0x0	Reserved.
5:4	**DTAG_SS**	RW	0x1	Timing Control for 128x22x4 RAM (DTAG).
3:2	**RSVD0**	RO	0x0	Reserved.
1:0	**ITAG_SS**	RW	0x1	Timing Control for 64x22x4 RAM (ITAG).

DESCRIPTION:

ARM Cache Control Register.

EXAMPLE:

```
cache_timing = HW_DIGCTL_ARMCACHE.CACHE_SS;
```

7.6.35. SigmaTel Copyright Identifier Register Description

Read-only SigmaTel Copyright Identifier Register.

HW_DIGCTL_SGTL 0x8001C300

Table 177. HW_DIGCTL_SGTL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COPYRIGHT																															

Table 178. HW_DIGCTL_SGTL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COPYRIGHT**	RO	0x6d676953	This read-only bit field contains the four bytes of the SigmaTel Copyright Identification String.

DESCRIPTION:

This register provides read-only access to the zero-terminated twelve-byte SigmaTel copyright identification string. This register behaves somewhat differently from all other APB registers in that it provides different read-back values at its four successive SCT bus addresses. The following binary values are read back at 0x8001C300, 0x8001C304, and 0x8001C308 respectively:

0x6d676953 M,G,I,S at 0x8001C300

0x6c655461 L,E,T,A at 0x8001C304

0x00AEA92d 0x00, Registered Trademark Symbol;, ©, hyphen at 0x8001C308

0x00AEA92d 0xBA, 0xD0, I, S at 0x8001C30C

The debugger does a string compare on these 12 successive little endian bytes. Any chip that reads back these values is either a SigmaTel chip or it is a competitors chip that is violating SigmaTel registered trademarks and or copyrights.

EXAMPLE:

```
printf("%s", (char *)HW_DIGCTL_SGTL_ADDR);
```

7.6.36. Digital Control Chip Revision Register Description

Read-only chip revision register.

HW_DIGCTL_CHIPID 0x8001C310

Table 179. HW_DIGCTL_CHIPID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PRODUCT_CODE																RSVD1								REVISION							

Table 180. HW_DIGCTL_CHIPID Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**PRODUCT_CODE**	RO	0x3600	This read-only bit field always reads back the chip ID. The lower eight bits of the chip ID are constructed from laser fuse values.
15:8	**RSVD1**	RO	0x00	Always write zeroes to this bit field.
7:0	**REVISION**	RO	0x00	This read-only bit field always reads back the mask revision level of the chip. TA1 = 0x0 TA2 = 0x1 TA3/TA3A/1.3 = 0x2 1.4 = 0x3 1.5 = 0x4 B1/2.0 = 0x5

DESCRIPTION:

Chip Identification Register.

EXAMPLE:

```
FormatAndPrintChipID(HW_DIGCTL_CHIPID_PRODUCT_CODE,HW_DIGCTL_CHIPID_REVISION );
```

DIGCTL XML Revision: 1.79

8. USB HIGH-SPEED ON-THE-GO (HOST/DEVICE) CONTROLLER

This chapter describes the USB high-speed On-the-Go controller included on the STMP36xx. It includes sections on the PIO, DMA, and UTMI interfaces, along with USB controller flowcharts. Descriptions for programmable registers mentioned in this chapter can be found in Section 4.9 on page 56, Section 7.6 on page 128, Section 9.6 on page 169, and Section 31.8 on page 759.

8.1. Overview

The STMP36xx includes a Universal Serial Bus (USB) version 2.0 controller capable of operating as either a USB device or a USB host, as shown in Figure 23. In addition, it contains supporting circuitry for USB On-the-Go (OTG). The USB controller is used to download digital music data or program code into external memory and to upload voice recordings from memory to the PC. Program updates can also be loaded into the flash memory area using the USB interface.

As a host controller, it can enumerate and control USB devices attached to it. Using the OTG features, it can negotiate with another OTG system to be either the host or the device in a peer connection.

The USB controller operates either in full-speed mode or high-speed mode.

Refer to the USB Implementer's Forum website www.usb.org for detailed specifications and information on the USB protocol, timing and electrical characteristics.

The USB 2.0 controller comprises both a programmed I/O (PIO) interface and a DMA interface. Both of these interfaces, as implemented in the ARC (TransDimension) High-Speed USB core, are designed to meet an ARM Ltd. AMBA Hardware Bus (AHB). The AHB is used by the USB controller as a slave (PIO register accesses) and as a master (DMA memory accesses).

The USB 2.0 PHY is fully integrated on-chip and is described in Chapter 9, beginning on page 161. The PHY is controlled over the APBX peripheral bus.

8.2. USB Controller Core

The USB controller is an instantiation of the ARC USB controller core. This proprietary core, the intellectual property it represents, and the copyrighted documentation for the core are the property of ARC International. For detailed information about the controller core, refer to the *TD243 USB Host/Peripheral/OTG Controller Datasheet* (http://www.transdimension.com/downloads/index.html).

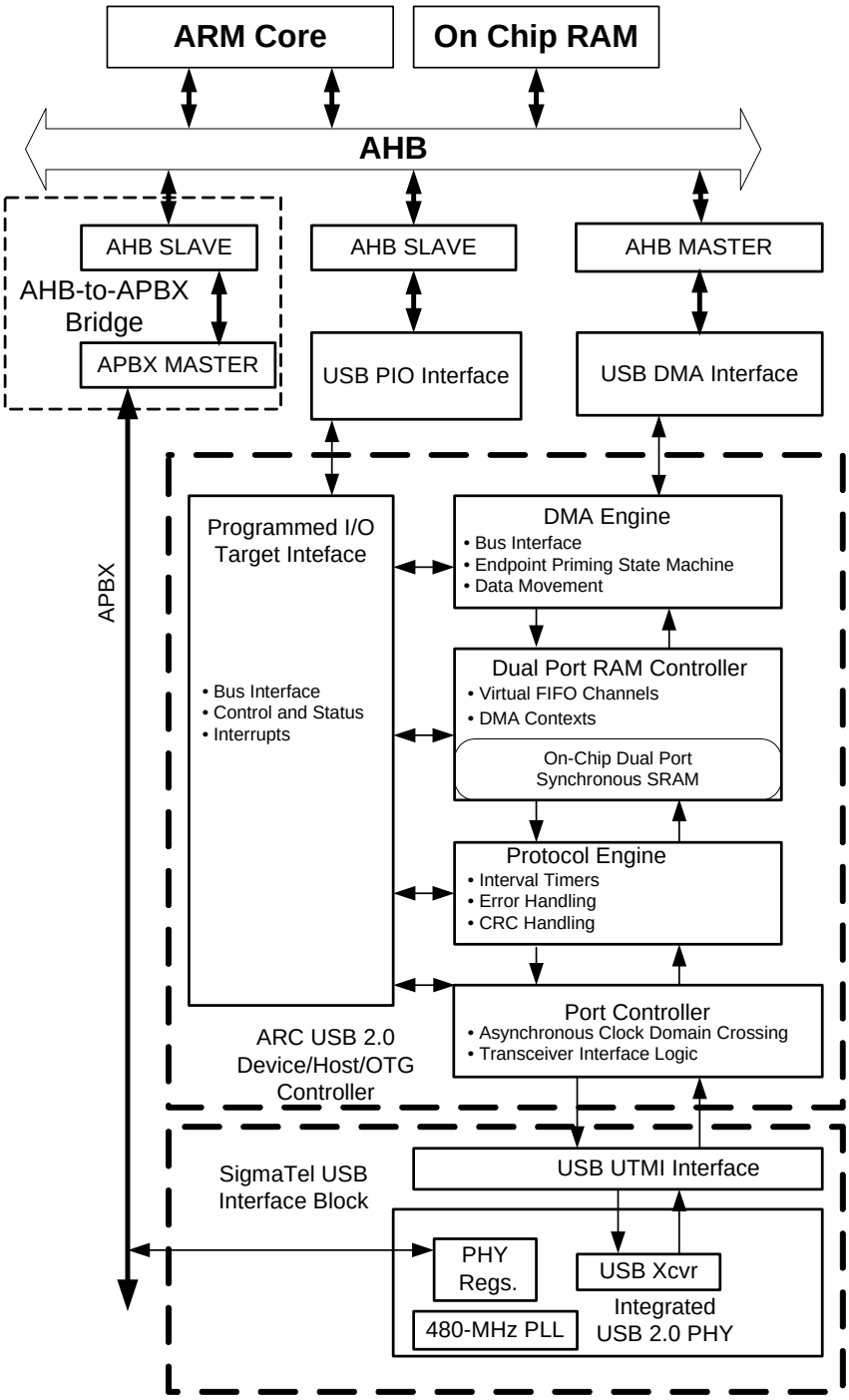


Figure 23. USB 2.0 Device Controller Block Diagram

8.3. USB Programmed I/O (PIO) Target Interface

The PIO interface is on an AHB slave of the TDI controller. It allows the ARM processor to access the configuration, control, and status registers. There are identification registers for hardware configuration parameters and operational registers for control and status.

8.4. USB DMA Interface

The DMA is a master AHB interface that allows USB data to be transferred to/from the system memory. The data in memory is structured to implement a software framework supported by the controller. For a device controller, this structure is a link-list interface that consists of queue heads and pointers that are transfer descriptors. The queue head is where transfers are managed. It has status information and location of the data buffers. The hardware controller's PIO registers enable the entire data structure, and once USB data is transferred in between the host, the status of the transfer is updated in the queue head, with minimal latency to the system.

For a host controller, there is also a link-list interface. It consists of a periodic frame list and pointers to transfer descriptors. The period frame list is a schedule of transfers. The frame list points to the data buffers through the transfer descriptors. The hardware controller's PIO registers enable the data structure and manage the transfers within a USB frame. The period frame list works as a sliding window of host transfers over time. As each transfer is completed, the status information is updated in the frame list.

For high-speed USB transmissions, use on-chip RAM (OC-RAM) as the data source instead of SDRAM. SDRAM does not have the bandwidth needed to supply the USB DMA engine at the required rate of 60 Mbytes per second (480 MBits/s USB high-speed bit rate). For full-speed transmissions, the STMP36xx has the bandwidth to handle the data buffers in SDRAM. However, the queue heads (dQH, as described in the ARC manual) must be placed in OC-RAM. A design limitation on burst size does not allow the queue heads to be placed in SDRAM.

8.5. USB UTMI Interface

The SigmaTel-developed test mode logic allows the integrated USB PHY to be exported for standalone use. In addition, the test modes include both digital and analog loopback tests.

8.5.1. Exporting the PHY

The STMP36xx USB PHY interface can be configured to be a standalone USB PHY. In this mode, the UTMI interface is exposed on the pins. This mode only supports two 16-bit unidirectional data buses.

8.5.2. Digital/Analog Loopback Test Mode

Since the UTMI has to operate at high frequencies (480 MHz), it has a capacity to self-test. A pseudo-random number generator transmits data to the receive path, and data is compared for validity. In the digital loopback, the data transfer only resides in the UTMI. It checks for sync, EOP, and bit-stuffing generation and data integrity. The analog loopback is the same as the digital loopback, but involves the analog PHY. This allows for checking of the high-speed (HS) and full-speed (FS) comparators and transmitters.

8.6. USB Controller Flowcharts

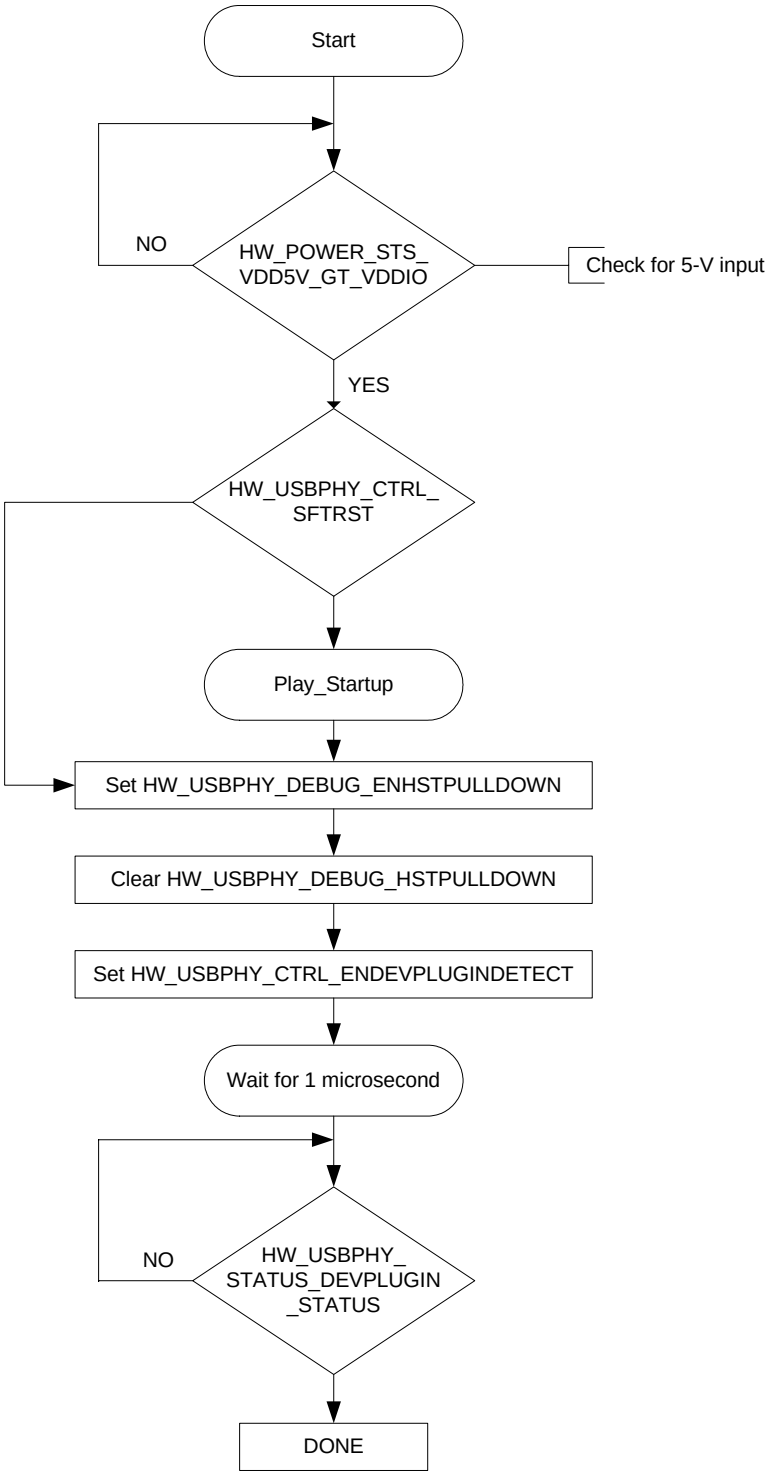


Figure 24. USB 2.0 Check_USB_Plugged_In Flowchart

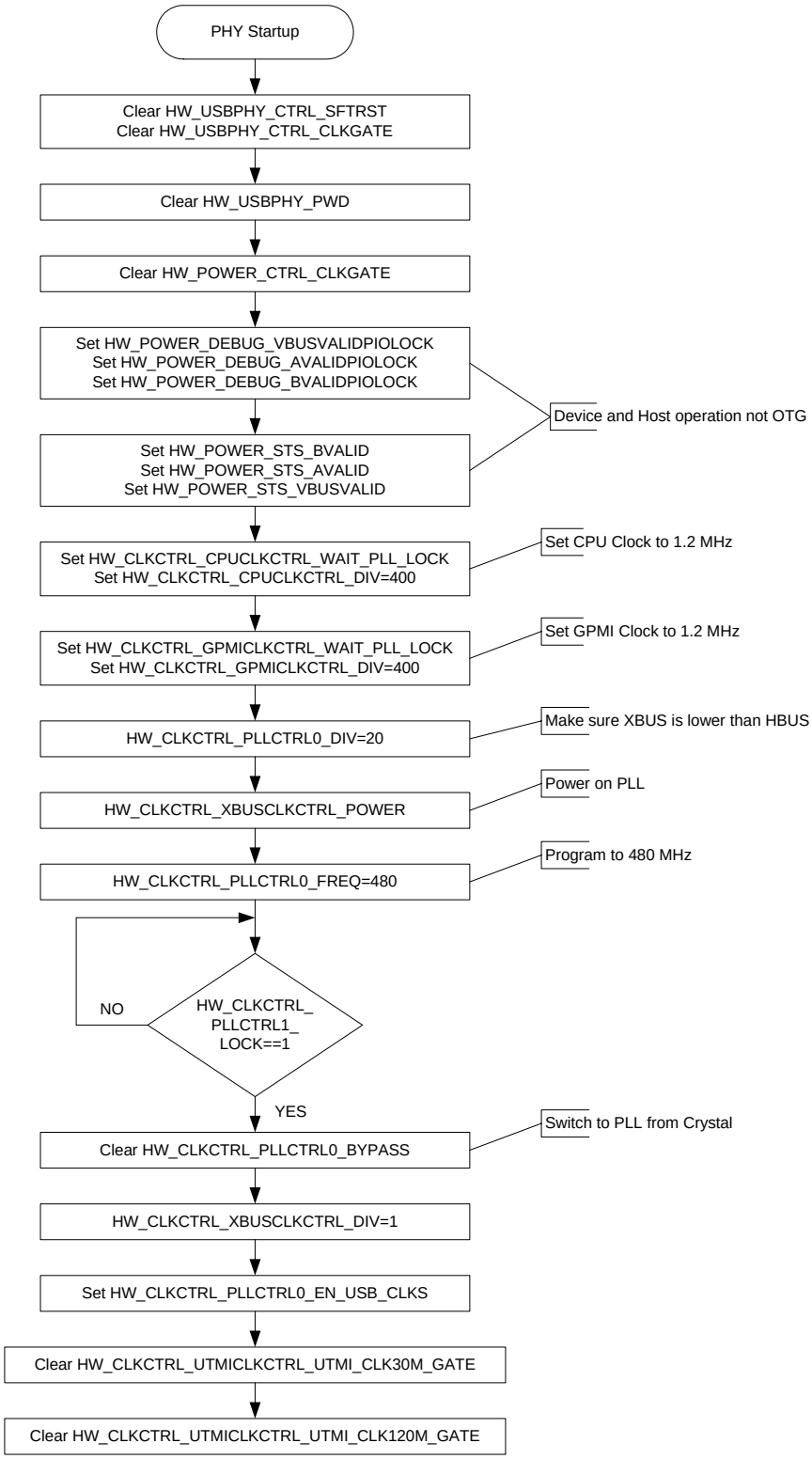


Figure 25. USB 2.0 USB PHY Startup Flowchart

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

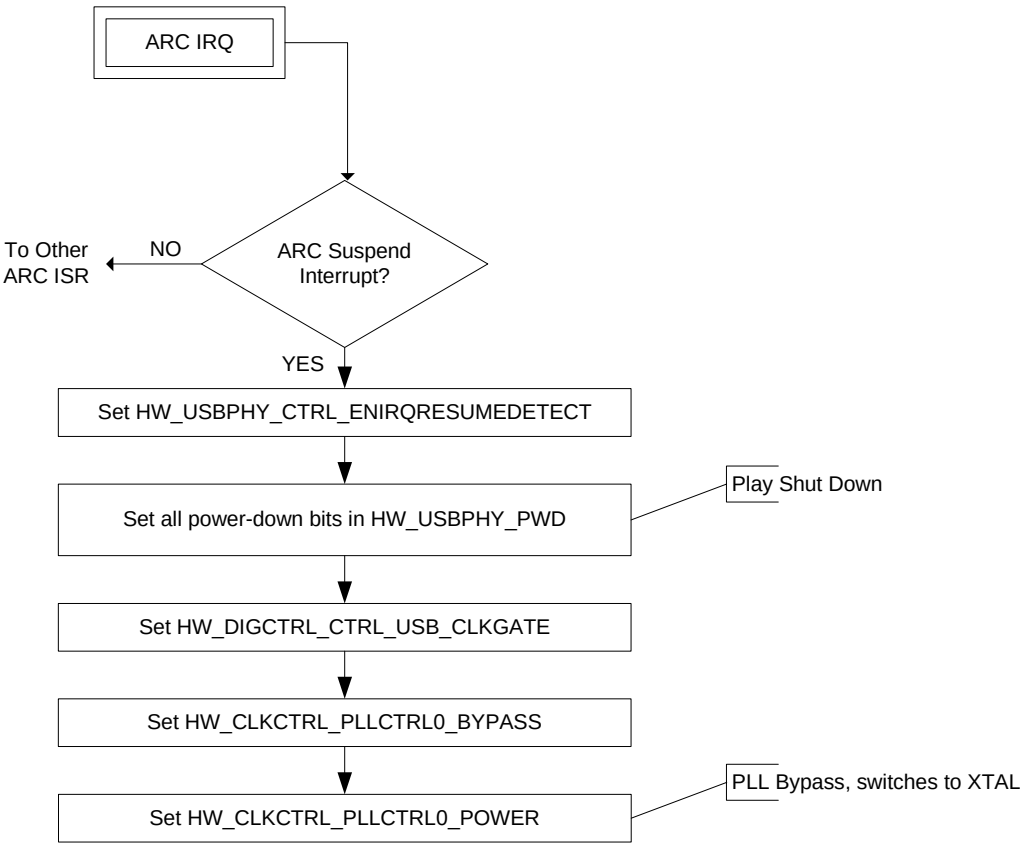


Figure 26. USB 2.0 PHY PLL Suspend Flowchart

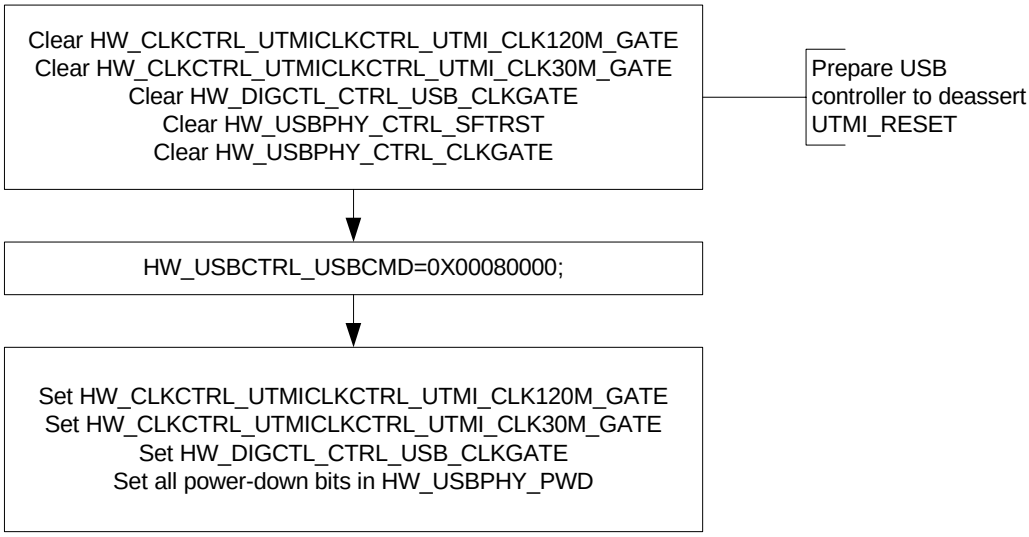


Figure 27. UTMI Powerdown

9. INTEGRATED USB 2.0 PHY

This chapter describes the integrated USB 2.0 full-speed and high-speed PHY available on the STMP36xx. It includes sections on external signals, the UTMI and digital circuits, and the analog transceiver. Programmable registers are described in Section 9.6.

9.1. Overview

The STMP36xx contains an integrated USB 2.0 PHY macrocell capable of connecting to PC host systems at the USB full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s. See Figure 28 for a block diagram of the PHY. The integrated PHY provides a standard UTMI interface. This allows the STMP36xx to alternatively connect to an external UTMI-compliant USB 2.0 controller chip.

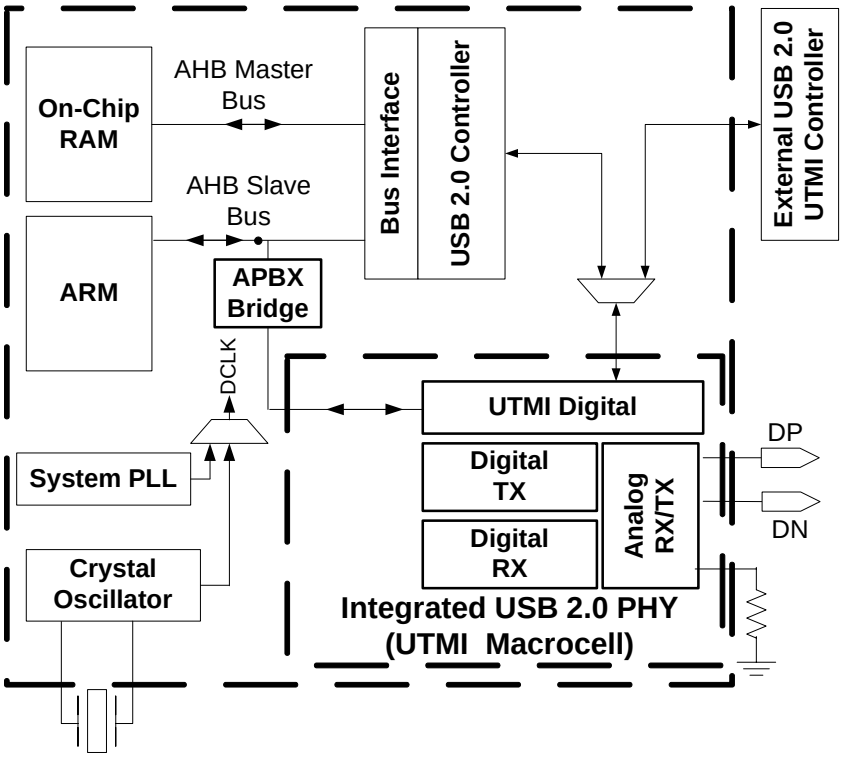


Figure 28. USB 2.0 PHY Block Diagram

The following subsections describe the external interfaces, internal interfaces, major blocks, and programable registers that comprise the integrated USB 2.0 PHY.

9.2. External Signals

- **DP, DN**: These pins connect directly to a USB device connector.
- **Precision Calibration Resistor**: This pin connects a 620Ω ± 1% resistor to ground. The key on-chip resistor—the 45Ω high-speed termination resistor—contains digitally controlled trimming and calibration circuits to match its impedance to the external precision resistor for USB 2.0 specification

compliance in device operation. A 15KΩ pulldown on the DP and DN pins allows for compliance in host operation.

9.3. UTMI and Digital Circuits

The UTMI provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz. There are four parts to the UTMI/digital circuits block: the UTMI block, the digital transmitter, the digital receiver, and the programmable registers block.

9.3.1. UTMI Block

This block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection. The PLL supplies a 120-MHz signal to all of the digital logic. The UTMI block does a final divide by four to develop the 30-MHz clock used in the interface.

9.3.2. Digital Transmitter Block

The digital transmitter block receives the 16-bit transmit data from the USB controller, and handles the tx_valid, tx_validh and tx_ready handshake. In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

9.3.3. Digital Receiver Block

The digital receiver block receives the raw serial bitstream either from the HS differential transceiver or from the FS differential transceiver. The HS input goes to a very fast DLL that uses one of eight identically spaced phases of the 480-MHz clock to pick a sample point. As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480-Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, etc.).

9.3.4. Programmable Registers Block

The PHY contains four 24 bit programmable registers, which are fully described in Section 9.6.

9.4. Analog Transceiver

The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 29.

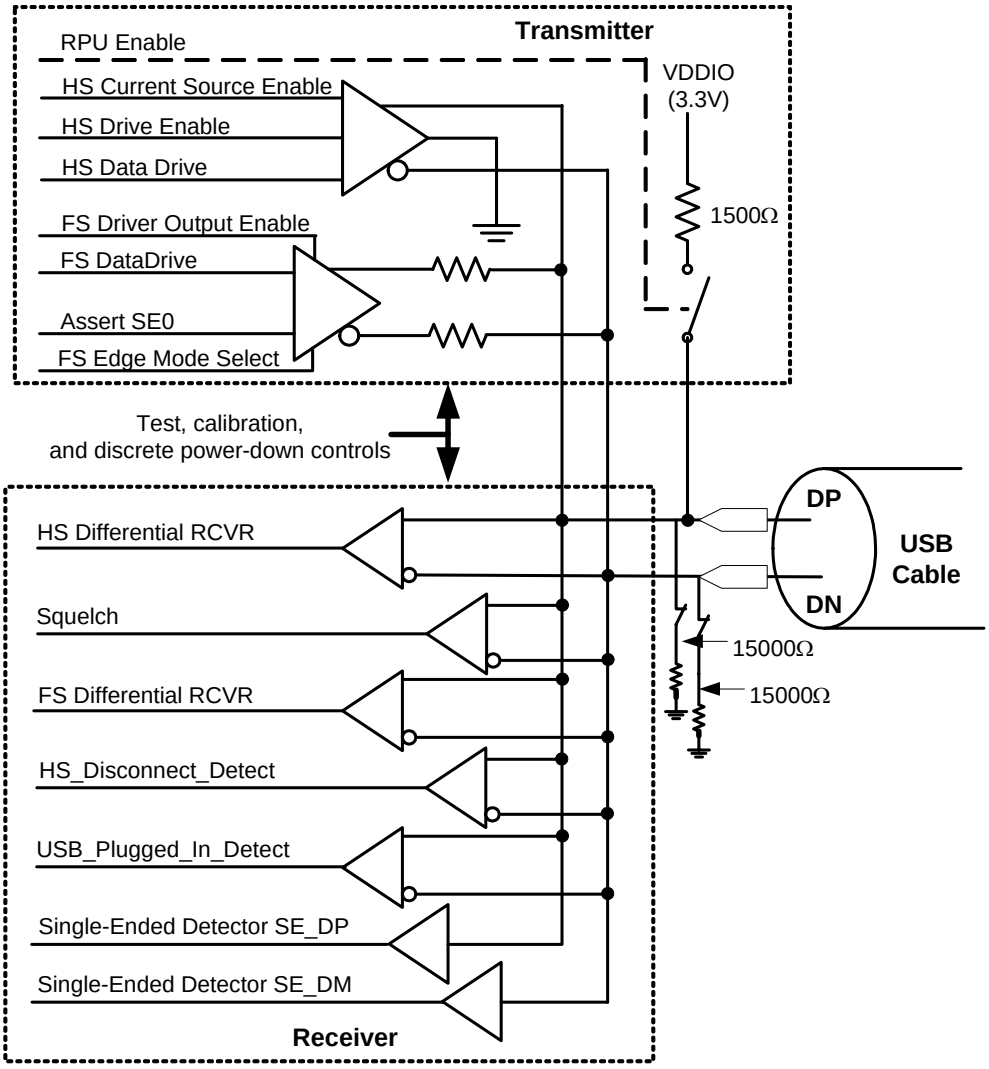


Figure 29. USB 2.0 PHY Analog Transceiver Block Diagram

9.4.1. Analog Receiver

The analog receiver comprises five differential receivers and two single-ended receivers, described below.

9.4.1.1. HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is a one if the differential signal is greater than a 0-V threshold. Its output is zero otherwise. Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual

45Ω terminations found on each leg of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

9.4.1.2. Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator. Its output is a one if the differential magnitude is less than a nominal 100-mV threshold. Its output is zero otherwise. Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

9.4.1.3. FS Differential Receiver

The full-speed differential receiver is both a differential analog receiver and threshold comparator. The crossover voltage falls between 1.3 V and 2.0 V. Its output is a one when the D_P line is above the crossover point and the D_N line is below the crossover point.

9.4.1.4. HS Disconnect Detector

This host-side function is not used in STMP36xx applications, but is included to make a complete UTMI macrocell. It is a differential analog receiver and threshold comparator. Its output is a one if the differential magnitude is greater than a nominal 575-mV threshold. Its output is zero, otherwise.

9.4.1.5. USB Plugged-In Detector

The USB plugged-in detector looks for both D_P and D_N to be high. There is a pair of large on-chip pullup resistors (200KΩ) that hold both D_P and D_N high when the USB cable is not attached. The USB plugged-in detector signals a zero in this case.

When in device mode, the host/hub interface that is *upstream* from the STMP36xx contains a 15KΩ pulldown resistor that easily overrides the 200KΩ pullup. When plugged in, at least one signal in the pair will be low, which will force the plugged-in detector's output high.

9.4.1.6. Single Ended D_P Receiver

The single ended D_P receiver output is high whenever the D_P input is above its nominal 1.8-V threshold.

9.4.1.7. Single Ended D_N Receiver

The single ended D_N receiver output is high whenever the D_N input is above its nominal 1.8-V threshold.

9.4.2. Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable1.5KΩ pullup resistor.

9.4.2.1. Switchable High-Speed 45Ω Termination Resistors

High-speed current mode differential signaling requires good 90Ω differential termination at each end of the USB cable. This results from switching in 45Ω terminating resistors from each signal line to ground at each end of the cable. Because each signal is parallel terminated with 45Ω at each end, each driver sees a 22.5Ω load.

This is much too low of a load impedance for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure 30. The HW_USBPHYTX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45Ω terminator on the D_P signal. The calibration operation is described in Section 9.4.2.6.

9.4.2.2. Full-Speed Differential Driver

The full-speed differential drivers are essentially “open drain” low-impedance pull-down devices that are switched in a differential mode for full-speed signaling, i.e., either one or the other device is turned on to signal the “J” state or the “K” state. These drivers are both turned on, simultaneously, for high-speed signaling. This has the effect of switching in both 45Ω terminating resistors. The tx_fs_hiz signal originates in the digital transmitter section. The hs_term signal that also controls these drivers comes from the UTMI.

9.4.2.3. High-Speed Differential Driver

The high-speed differential driver receives a 17.78-mA current from the constant current source and essentially steers it down either the D_P signal or the D_N signal or alternatively to ground. This current will produce approximately a 400-mV drop across the 22.5Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78-mA current source is referenced back to the integrated voltage-band-gap circuit. The Iref, IBias, and V to I circuits are shared with the integrated battery charger.

9.4.2.4. Switchable 1.5KΩ DP Pullup Resistor

The STMP36xx contains a switchable 1.5KΩ pullup resistor on the D_P signal. This resistor is switched on to tell the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until processor software enables the announcement of a full-speed device.

9.4.2.5. Switchable 15KΩ DP Pulldown Resistor

The STMP36xx contains a switchable 15KΩ pulldown resistor on both D_P and D_N signals. This is used in host mode to tell the device controller that a host is present.

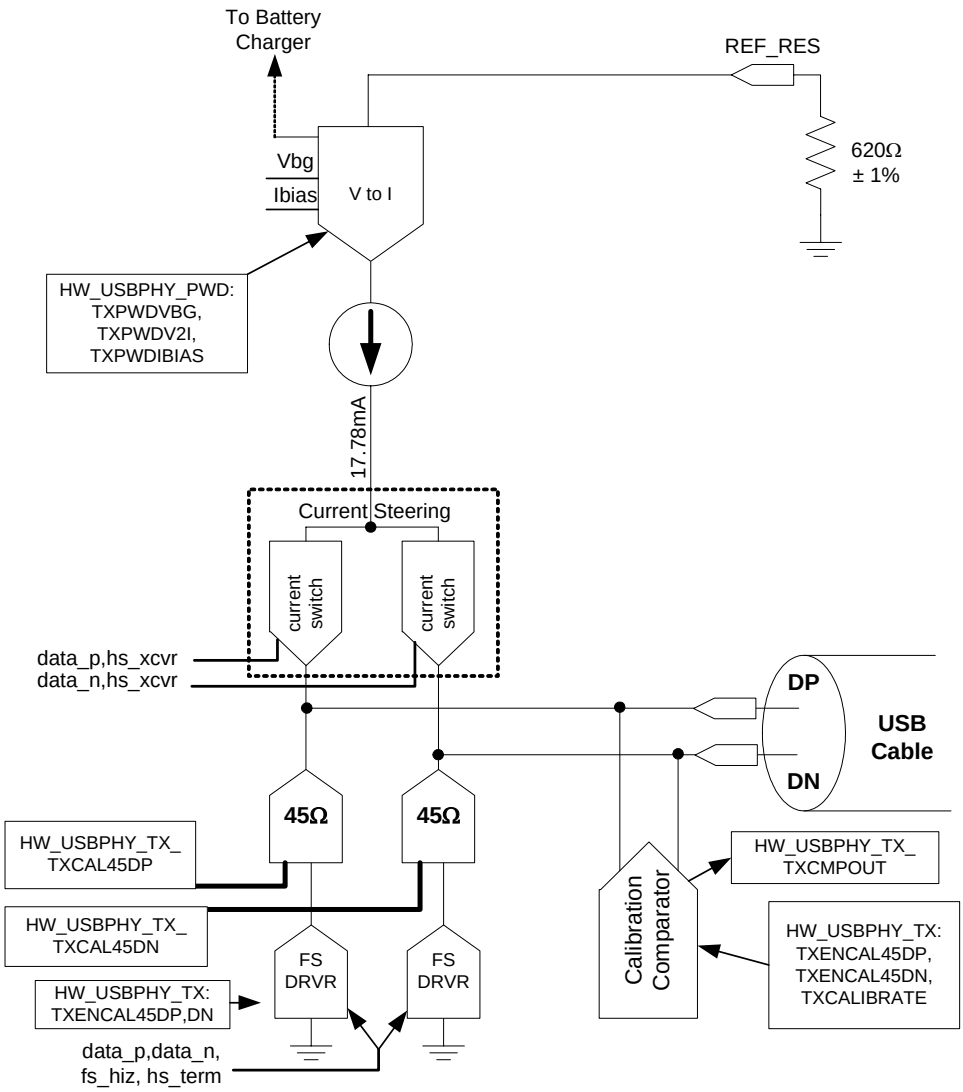


Figure 30. USB 2.0 PHY Transmitter Block Diagram

Table 181 summarizes the response of the PHY analog transmitter to various states of UTMI input and key transmit/receive state machine states.

Table 181. USB PHY Terminator States

UTMI OPMODE	UTMI TERM	UTMI XCVR	T/R	FUNCTION	45 Ω HIZ	1500Ω HIZ	
00=Normal	0	0	X	HS	0	1	
	1	1	T	FS	0	0	
	1	1	R	FS	1	0	**SUSPEND**
	1	0	R	CHIRP	1	0	
	1	0	T	CHIRP	1	0	
	0	1	X	DISCONNECT	1	1	
01=NoDrive	0	0	T	HS	1	1	
	0	0	R	HS	1	1	
	1	1	X	FS	1	1	
	1	0	X	CHIRP	1	1	
	0	1	X	DISCONNECT	1	1	**POR**
10=NoNRZI NoBitStuff	0	0	X	HS	0	1	
	1	1	T	FS	0	0	
	1	1	R	FS	1	0	
	1	0	R	CHIRP	1	0	
	1	0	T	CHIRP	1	0	
	0	1	X	DISCONNECT	1	1	
11= Invalid	0	0	T	HS	1	1	
	0	0	R	HS	1	1	
	1	1	X	FS	1	1	
	1	0	X	CHIRP	1	1	
	0	1	X	DISCONNECT	1	1	

9.4.2.6. Resistor Calibration Mode

The analog transmitter section includes a calibration comparator that can monitor the D_P or D_N voltage, as desired. Setting HW_USBPHY_TX_TXENCAL45DP selects the D_P signal. The comparator output is visible in HW_USBPHY_TX_TXCMPOUT_STATUS. To calibrate the 45Ω D_P terminator, first set the field HW_USBPHY_TX_TXCAL45DP to all ones.

The flowchart in Figure 31 shows how to search for the proper trimming resistor to calibrate the D_P terminator. In general, follow these steps to perform a similar operation.

- Put the chip into termination resistor calibration mode for the D_P terminator.
- Start with the largest value of trimming select, i.e., all ones.
- Make several precise minimum delay calculations to allow the mixed signal components to stabilize.
- The comparator output is sampled and then checked. If the comparator has not tripped, reduce the value of the trimming select field and try again. Repeat these steps until the trip point is reached.

While the flowchart in Figure 31 shows how to calibrate the D_P terminator, calibration of the D_N terminator is accomplished in a similar manner, substituting *DN bit fields for *DP bit fields.

Note: The TXCAL45DP and TXCAL45DN bit fields are represented in the register description for HW_USBPHY_TX as 5-bit fields. However, the flowchart indicates and uses only four bits, which is correct for the STMP36xx. For resistor calibration on the STMP36xx, the most significant bit (5th bit) should always be 0, else an aliasing effect will occur.

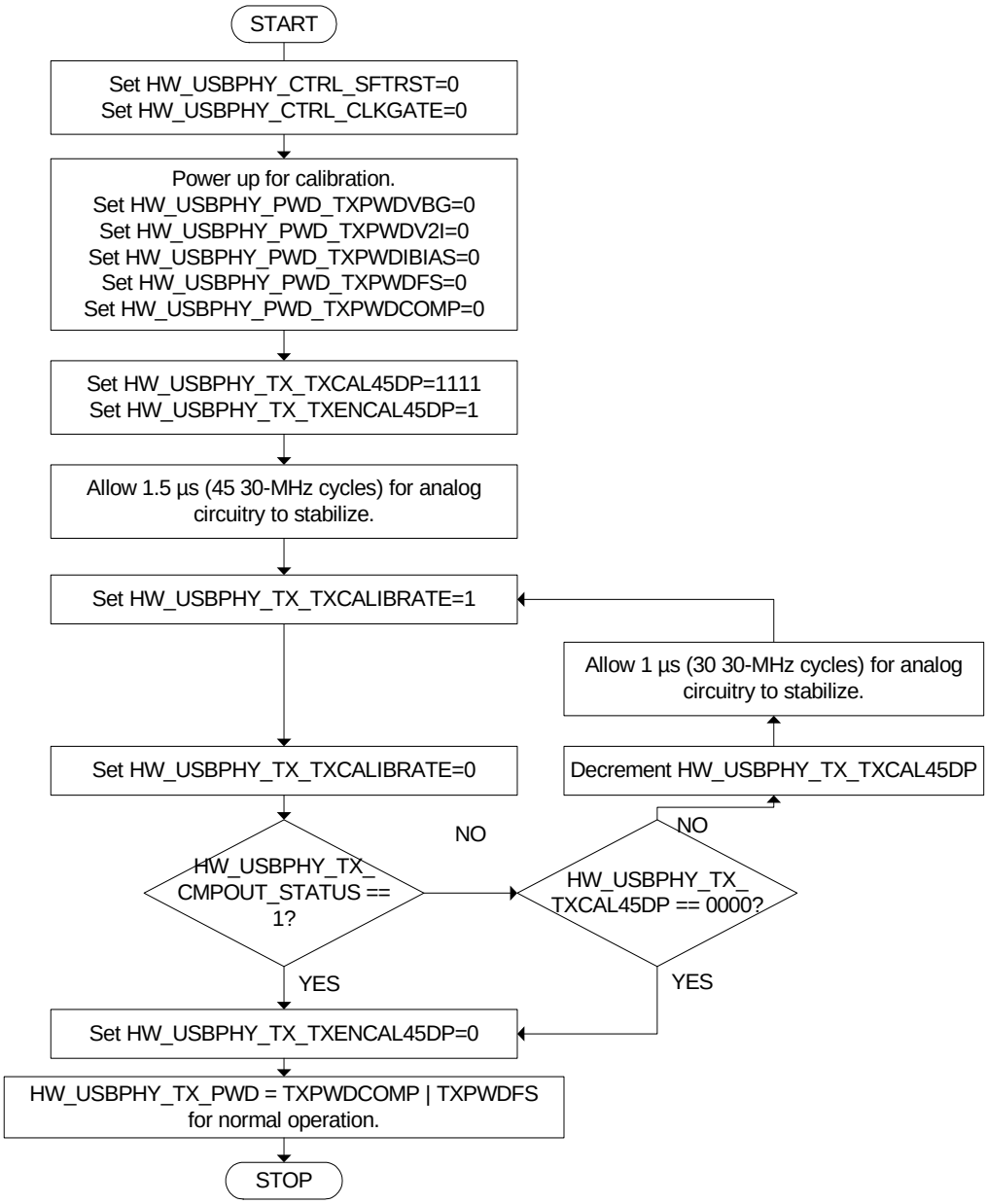


Figure 31. 45Ω Calibration Flowchart

9.5. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

9.6. Programmable Registers

The USB 2.0 integrated PHY contains the following directly programmable registers.

9.6.1. USB PHY Power-Down Register Description

The USB PHY Power-Down Register provides overall control of the PHY power state.

HW_USBPHY_PWD 0x8007c000
HW_USBPHY_PWD_SET 0x8007c004
HW_USBPHY_PWD_CLR 0x8007c008
HW_USBPHY_PWD_TOG 0x8007c00C

Table 182. HW_USBPHY_PWD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3											RXPWDRX	RXPWDDIFF	RXPWD1PT1	RXPWDENV	RSRVD2		TXPWDCOMP	TXPWDVBG	TXPWDV2I	TXPWDIBIAS	TXPWDFS	RSRVD1						RSRVD0			

Table 183. HW_USBPHY_PWD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSRVD3**	RO	0x0	Reserved
20	**RXPWDRX**	RW	0x1	Set to one to power down the entire USB PHY receiver block, except for the full-speed differential receiver. Set to zero for normal operation.
19	**RXPWDDIFF**	RW	0x1	Set to one to power down the USB high-speed differential receiver. Set to zero for normal operation.
18	**RXPWD1PT1**	RW	0x1	Set to one to power down the USB full-speed differential receiver. Set to zero for normal operation.
17	**RXPWDENV**	RW	0x1	Set to one to power down the USB high-speed receiver envelope detector (squelch signal). Set to zero for normal operation.
16:15	**RSRVD2**	RO	0x0	Reserved
14	**TXPWDCOMP**	RW	0x1	Set to one to power down the USB PHY transmit calibration comparator. Set to zero during calibration, and set to one after calibration is complete.

Table 183. HW_USBPHY_PWD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
13	**TXPWDVBG**	RW	0x1	Set to one to power down the USB PHY transmit voltage bandgap buffer amplifier, as well as the V-to-I converter and the current mirror. Note that these circuits are shared with the battery charge circuit. Setting this to one will not power down these circuits, unless the corresponding bit in the battery charger is also set for power down. Set to zero for normal operation and for calibration.
12	**TXPWDV2I**	RW	0x1	Set to one to power down the USB PHY transmit V-to-I converter and the current mirror. Note these circuits are shared with the battery charge circuit. Setting this to one will not power down these circuits, unless the corresponding bit in the battery charger is also set for power down. Set to zero for normal operation and for calibration.
11	**TXPWDIBIAS**	RW	0x1	Set to one to power down the USB PHY current bias block for the transmitter. This bit should only be set when the USB is in suspend mode. This effectively powers down the entire USB transmit path. Note these circuits are shared with the battery charge circuit. Setting this to one will not power down these circuits, unless the corresponding bit in the battery charger is also set for power down. Set to zero for normal operation and for calibration.
10	**TXPWDFS**	RW	0x1	Set to one to power down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-Z output. Set to zero during calibration and set to one after calibration is complete.
9:5	**RSRVD1**	RO	0x0	Reserved
4:0	**RSRVD0**	RO	0x0	Reserved

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.2. USB PHY Transmitter Control Register Description

The USB PHY Transmitter Control Register handles the transmit calibration controls and other transmit controls.

HW_USBPHY_TX 0x8007c010
HW_USBPHY_TX_SET 0x8007c014
HW_USBPHY_TX_CLR 0x8007c018
HW_USBPHY_TX_TOG 0x8007c01C

Table 184. HW_USBPHY_TX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD6								TXCMPOUT_STATUS	RSRVD5	TXENCAL45DP	TXCAL45DP					RSRVD3		TXENCAL45DN	TXCAL45DN					TXCALIBRATE	RSRVD1						

Table 185. HW_USBPHY_TX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD6**	RO	0x0	Reserved
23	**TXCMPOUT_STATUS**	RW	0x0	The calibration comparator output is latched to this bit. This bit should be erased for every new calibration. This bit can be set to zero or set to one by normal write from the CPU. In addition, it is loaded with the state of the calibration comparator's output whenever HW_USBPHY_TXCALIBRATE is set to one. It continously copies the comparator to this bit as long TXCALIBRATE is set to one, i.e., when TXCALIBRATE is one, writing to this bit has no effect.
22	**RSRVD5**	RO	0x0	Reserved
21	**TXENCAL45DP**	RW	0x0	Set to one for time during compare of the 45-Ohm DP termination resistor to the reference resistor. This bit should be set to one each time a new value of HW_USBPHY_TX_TXCAL45DP is set in order to compare the resulting resistance. NOTE: Only one of the following bits can be set to one for any calibration operation: HW_USBPHY_TX_TXENCAL45DN or HW_USBPHY_TX_TXENCAL45DP. Set to zero when the DP calibration is completed. The result of the comparison can be seen in HW_USBPHY_TX_TXCMPOUT.
20:16	**TXCAL45DP**	RW	0x6	Decode to select a 45-Ohm resistance to the DP output pin. Maximum resistance = 0000. Resistance is centered by design at 0110. Perform calibration routine by initially setting to 1111 and counting down until comparator trips. Note that while this field is 5 bits, the msb is not used on the STMP360xx and should be 0.
15:14	**RSRVD3**	RO	0x0	Reserved

Table 185. HW_USBPHY_TX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
13	**TXENCAL45DN**	RW	0x0	Set to one for time during compare of the 45-Ohm DN termination resistor to the reference resistor. This bit should be set to one each time a new value of HW_USBPHY_TX_TXCAL45DP is set in order to compare the resulting resistance. NOTE: Only one of the following bits can be set to one for any calibration operation: HW_USBPHY_TX_TXENCAL45DN or HW_USBPHY_TX_TXENCAL45DP. Set to zero when the DP calibration is completed. The result of the comparison can be seen in HW_USBPHY_TX_TXCMPOUT.
12:8	**TXCAL45DN**	RW	0x6	Decode to select a 45 Ohm resistance to the DN output pin. Maximum resistance = 0000. Resistance is centered by design at 0110. Perform calibration routine by initially setting to 1111 and counting down until comparator trips. Note that while this field is 5 bits, the msb is not used on the STMP360xx and should be 0.
7	**TXCALIBRATE**	RW	0x0	Set to one to effect calibration of any of the three precision resistances and set back to zero to read the results of calibration in HW_USBPHY_TX_TXCMPOUT. When set to one, it causes the calibration comparator output to continously update the state of HW_USBPHY_TX_TXCMPOUT. Set to zero for normal operation. NOTE: Only one of the following bits can be set to one for any calibration operation: HW_USBPHY_TX_TXENCAL45DN or HW_USBPHY_TX_TXENCAL45DP.
6:0	**RSRVD1**	RO	0x0	Reserved

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.3. USB PHY Receiver Control Register Description

The USB PHY Receiver Control Register handles receive path controls.

HW_USBPHY_RX 0x8007c020
HW_USBPHY_RX_SET 0x8007c024
HW_USBPHY_RX_CLR 0x8007c028
HW_USBPHY_RX_TOG 0x8007c02C

Table 186. HW_USBPHY_RX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD5									RXDBYPASS	RSRVD3						RSRVD4								RSRVD1		DISCONADJ		RSRVD0		ENVADJ	

Table 187. HW_USBPHY_RX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:23	**RSRVD5**	RO	0x0	Reserved
22	**RXDBYPASS**	RW	0x0	Set to one to use the output of the DP single-ended receiver in place of the full-speed differential receiver. This test mode is intended for lab use only. Set to zero for normal operation.
21:16	**RSRVD3**	RO	0x0	Reserved
15:8	**RSRVD4**	RO	0x0	Reserved
7:6	**RSRVD1**	RO	0x0	Reserved
5:4	**DISCONADJ**	RW	0x0	The DISCONADJ field adjusts the trip point for the disconnect detector. Trip Level Voltage 0.57500 V = 0000. Trip Level Voltage 0.56875 V = 0001. Trip Level Voltage 0.58125 V = 0010. Trip Level Voltage 0.58750 V = 0011. Reserved = 01XX. Reserved = 1XXX.
3:2	**RSRVD0**	RO	0x0	Reserved
1:0	**ENVADJ**	RW	0x0	The ENVADJ field adjusts the trip point for the envelope detector. Trip Level Voltage 0.10000 V = 0000. Trip Level Voltage 0.10625 V = 0001. Trip Level Voltage 0.11225 V = 0010. Trip Level Voltage 0.12500 V = 0011. Reserved = 01XX. Reserved = 1XXX.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.4. USB PHY General Control Register Description

The USB PHY General Control Register handles OTG and host controls. This register also includes interrupt enables and connectivity detect enables and results.

HW_USBPHY_CTRL 0x8007c030

HW_USBPHY_CTRL_SET 0x8007c034
HW_USBPHY_CTRL_CLR 0x8007c038
HW_USBPHY_CTRL_TOG 0x8007c03C

Table 188. HW_USBPHY_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	UTMI_SUSPENDM	RSRVD5																		RESUME_IRQ	ENIRQRESUMEDETECT	RSRVD3	ENOTGIDDETECT	RSRVD2			ENDEVPLUGINDETECT	HOSTDISCONDETECT_IRQ	ENIRQHOSTDISCON	ENHOSTDISCONDETECT	ENHSPRECHARGEXMIT

Table 189. HW_USBPHY_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Writing a one to this bit will soft-reset the HW_USBPHY_PWD, HW_USBPHY_TX, HW_USBPHY_RX, and HW_USBPHY_CTRL registers.
30	**CLKGATE**	RW	0x1	Gate UTMI Clocks. Set to 1 to gate clocks. Set to 0 for running clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29	**UTMI_SUSPENDM**	RO	0x0	Used by the PHY to indicate a powered-down state. If all the power-down bits in the HW_USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.
28:11	**RSRVD5**	RO	0x0	Reserved
10	**RESUME_IRQ**	RW	0x0	Indicates that the host is sending a wake-up after suspend. It is reset by software by writing a zero to the bit position or by writing a one to the SCT clear address space.
9	**ENIRQRESUMEDETECT**	RW	0x0	Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.
8	**RSRVD3**	RO	0x0	Reserved
7	**ENOTGIDDETECT**	RW	0x0	Enables circuit to detect resistance of MiniAB ID pin.
6:5	**RSRVD2**	RO	0x0	Reserved
4	**ENDEVPLUGINDETECT**	RW	0x0	For device mode, enables 200-KOhm pullups for detecting connectivity to host.

Table 189. HW_USBPHY_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**HOSTDISCONDETECT_IRQ**	RW	0x0	Indicates that the device has disconnected in high-speed mode. It is reset by software by writing a zero to the bit position or by writing a one to the SCT clear address space.
2	**ENIRQHOSTDISCON**	RW	0x0	Enables interrupt for detection of disconnection to device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.
1	**ENHOSTDISCONDETECT**	RW	0x0	For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection, which is normally done in the UTMI controller. The UTMI controller will enable this circuit whenever the host sends a start-of-frame packet.
0	**ENHSPRECHARGEXMIT**	RW	0x1	Set to one to enable the high-speed transmit precharge circuit .

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.5. USB PHY Status Register Description

The USB PHY Status Register holds results of IRQ and other detects.

HW_USBPHY_STATUS 0x8007c040

Table 190. HW_USBPHY_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD6																					RESUME_STATUS	RSRVD5	OTGID_STATUS	RSRVD4	DEVPLUGIN_STATUS	RSRVD3		HOSTDISCONDETECT_STATUS	RSRVD2		

Table 191. HW_USBPHY_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:11	**RSRVD6**	RO	0x0	Reserved
10	**RESUME_STATUS**	RO	0x0	Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.
9	**RSRVD5**	RO	0x0	Reserved
8	**OTGID_STATUS**	RW	0x0	Indicates the results of ID pin on MiniAB plug. False (0) is when ID resistance is less than Ra_Plug_ID, thus indicating Host (A) side. True (1) is when ID resistance is greater than Rb_Plug_ID, thus indicating Device (B) side.
7	**RSRVD4**	RO	0x0	Reserved
6	**DEVPLUGIN_STATUS**	RO	0x0	Indicates that the device has been connected on the D plus and D minus lines.
5:4	**RSRVD3**	RO	0x0	Reserved
3	**HOSTDISCONDETECT_STATUS**	RO	0x0	Indicates that the device has disconnected while in high-speed host mode.
2:0	**RSRVD2**	RO	0x0	Reserved

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.6. USB PHY Debug Register Description

This register is used to debug the USB PHY.

HW_USBPHY_DEBUG 0x8007c050
HW_USBPHY_DEBUG_SET 0x8007c054
HW_USBPHY_DEBUG_CLR 0x8007c058
HW_USBPHY_DEBUG_TOG 0x8007c05C

Table 192. HW_USBPHY_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD5	CLKGATE	RSRVD4	SQUELCHRESETLENGTH				ENSQUELCHRESET	RSRVD3				SQUELCHRESETCOUNT				RSRVD2			ENTX2RXCOUNT	TX2RXCOUNT				RSRVD1		ENHSTPULLDOWN		HSTPULLDOWN		DEBUG_INTERFACE_HOLD	OTGIDPIOLOCK

Table 193. HW_USBPHY_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD5**	RO	0x0	Reserved
30	**CLKGATE**	RW	0x1	Gate Test Clocks. Set to 1 to gate clocks. Set to 0 for running clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29	**RSRVD4**	RO	0x0	Reserved
28:25	**SQUELCHRESETLENGTH**	RW	0xf	Duration of RESET in terms of the number of 480-MHz cycles.
24	**ENSQUELCHRESET**	RW	0x1	Set bit to allow squelch to reset high-speed receive.
23:21	**RSRVD3**	RO	0x0	Reserved
20:16	**SQUELCHRESETCOUNT**	RW	0x18	Delay in between the detection of squelch to the reset of high-speed receive.
15:13	**RSRVD2**	RO	0x0	Reserved
12	**ENTX2RXCOUNT**	RW	0x0	Set bit to allow a count down to transistion in between TX and RX.
11:8	**TX2RXCOUNT**	RW	0x0	Delay in between end of transmit to the begin of receive. This is a Johnson count value; thus, it will count to 8.
7:6	**RSRVD1**	RO	0x0	Reserved
5:4	**ENHSTPULLDOWN**	RW	0x0	Set bit 5 to 1 to override the ARC control of the D plus 15KOhm pulldown. Set bit 4 to 1 to override the ARC control of the D minus 15KOhm pulldown. Set to 0 to disable.
3:2	**HSTPULLDOWN**	RW	0x0	Set bit 3 to 1 to pull down 15KOhm on D plus line. Set bit 2 to 1 to pull down 15KOhm on D minus line. Set to 0 to disable.
1	**DEBUG_INTERFACE_HOLD**	RW	0x0	Use holding registers to assist in timing for external UTMI interface .
0	**OTGIDPIOLOCK**	RW	0x0	Once OTG ID from HW_USBPHY_STATUS.OTGID_STATUS, use this to hold the value. This is to save power for the comparators that are used to determine the ID status.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.7. *UTMI Debug Status Register 0 Description*

The UTMI Debug Status Register 0 holds results of running counters of error cases.

HW_USBPHY_DEBUG0_STATUS 0x8007c060

Table 194. HW_USBPHY_DEBUG0_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SQUELCH_COUNT						UTMI_RXERROR_FAIL_COUNT										LOOP_BACK_FAIL_COUNT															

Table 195. HW_USBPHY_DEBUG0_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**SQUELCH_COUNT**	RO	0x0	Running count of the squelch reset instead of normal end for high-speed receive.
25:16	**UTMI_RXERROR_FAIL_COUNT**	RO	0x0	Running count of the UTMI_RXERROR.
15:0	**LOOP_BACK_FAIL_COUNT**	RO	0x900d	Running count of the failed pseudo-random generator loopback.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.8. UTMI Debug Status Register 1 Description

The UTMI Debug Status Register 1 holds the observation of UTMI_TX_DATA and UTMI_RX_DATA.

HW_USBPHY_DEBUG1_STATUS 0x8007c070

Table 196. HW_USBPHY_DEBUG1_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UTMI_TX_DATA																UTMI_RX_DATA															

Table 197. HW_USBPHY_DEBUG1_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UTMI_TX_DATA**	RO	0x0	Snapshot of the UTMI transmit data bus.
15:0	**UTMI_RX_DATA**	RO	0x0	Snapshot of the UTMI receive data bus.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.9. UTMI Debug Status Register 2 Description

The UTMI Debug Status Register 2 holds the observation of the UTMI pins.

HW_USBPHY_DEBUG2_STATUS 0x8007c080

Table 198. HW_USBPHY_DEBUG2_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1									UTMI_TXVALIDH	UTMI_TXVALID	UTMI_TERMSELECT	UTMI_XCVRSELECT		UTMI_OPMODE		RSRVD0								UTMI_LINESTATE		UTMI_SUSPENDM	UTMI_RXVALIDH	UTMI_RXVALID	UTMI_RXACTIVE	UTMI_RXERROR	UTMI_TXREADY

Table 199. HW_USBPHY_DEBUG2_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:23	**RSRVD1**	RO	0x0	Reserved
22	**UTMI_TXVALIDH**	RO	0x0	Snapshot of the UTMI TXVALIDH signal.
21	**UTMI_TXVALID**	RO	0x0	Snapshot of the UTMI TXVALID signal.
20	**UTMI_TERMSELECT**	RO	0x0	Snapshot of the UTMI TERM SELECT signal.
19:18	**UTMI_XCVRSELECT**	RO	0x0	Snapshot of the UTMI XCVR SELECT signal.
17:16	**UTMI_OPMODE**	RO	0x0	Snapshot of the UTMI OPMODE bus.
15:8	**RSRVD0**	RO	0x0	Reserved
7:6	**UTMI_LINESTATE**	RO	0x0	Snapshot of the UTMI LINESTATE bus.
5	**UTMI_SUSPENDM**	RO	0x0	Snapshot of the UTMI SUSPENDM signal.
4	**UTMI_RXVALIDH**	RO	0x0	Snapshot of the UTMI RXVALIDH signal.
3	**UTMI_RXVALID**	RO	0x0	Snapshot of the UTMI RXVALID signal.
2	**UTMI_RXACTIVE**	RO	0x0	Snapshot of the UTMI RXACTIVE signal.
1	**UTMI_RXERROR**	RO	0x0	Snapshot of the UTMI RXERROR signal.
0	**UTMI_TXREADY**	RO	0x0	Snapshot of the UTMI TXREADY signal.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.10. UTMI Debug Status Register 3 Description

The UTMI Debug Status Register 3 holds the observation of internal state machines of the High Speed Receive. This includes the unstuff counter, main FSM, bit counter, squelch-detect FSM, and bit-start FSM.

HW_USBPHY_DEBUG3_STATUS 0x8007c090

Table 200. HW_USBPHY_DEBUG3_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2	B_CNT_FSM			RSRVD1		SQ_UNLOCK_FSM			RSRVD0	BIT_CNT										MAIN_HS_RX_FSM				UNSTUFF_BIT_CNT							

Table 201. HW_USBPHY_DEBUG3_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD2**	RO	0x0	Reserved
30:28	**B_CNT_FSM**	RO	0x0	Snapshot of the state machine that determines of the count starts at 0 or 1 after the completion of SYNC.
27:26	**RSRVD1**	RO	0x0	Reserved
25:23	**SQ_UNLOCK_FSM**	RO	0x0	Snapshot of the state machine that detects the removal of squelch and waits for the last transfer to complete.
22	**RSRVD0**	RO	0x0	Reserved
21:12	**BIT_CNT**	RO	0x0	Snapshot of the bit counter that resets at the end of SYNC and allows the other FSM to know the byte alignment.
11:8	**MAIN_HS_RX_FSM**	RO	0x0	Snapshot of the main high-speed FSM.
7:0	**UNSTUFF_BIT_CNT**	RO	0x0	Snapshot of the unstuff bit counter.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.11. UTMI Debug Status Register 4 Description

The UTMI Debug Status Register 4 holds the observation of internal state machines of the high-speed receive. This includes the byte FSM and FIFO-send FSM.

HW_USBPHY_DEBUG4_STATUS 0x8007c0a0

Table 202. HW_USBPHY_DEBUG4_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1				BYTE_FSM												RSRVD0		SND_FSM													

Table 203. HW_USBPHY_DEBUG4_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSRVD1**	RO	0x0	Reserved
28:16	**BYTE_FSM**	RO	0x0	Snapshot of the state machine that determines the number of bytes extracted from the data stream.
15:14	**RSRVD0**	RO	0x0	Reserved
13:0	**SND_FSM**	RO	0x0	Snapshot of the state machine that controls the byte/word transfer to the FIFO.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.12. UTMI Debug Status Register 5 Description

The UTMI Debug Status Register 5 holds the observation of internal state machines of the high-speed transmit. This includes the byte FSM and FIFO-send FSM.

HW_USBPHY_DEBUG5_STATUS 0x8007c0b0

Table 204. HW_USBPHY_DEBUG5_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4				MAIN_FSM				RSRVD3		SYNC_FSM						RSRVD2	PRECHARGE_FSM			RSRVD1	SHIFT_FSM			RSRVD0				SOF_FSM			

Table 205. HW_USBPHY_DEBUG5_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD4**	RO	0x0	Reserved
27:24	**MAIN_FSM**	RO	0x0	Snapshot of the main state machine that determines start of transmission.
23:22	**RSRVD3**	RO	0x0	Reserved
21:16	**SYNC_FSM**	RO	0x0	Snapshot of the state machine that controls the transmit of SYNC.
15	**RSRVD2**	RO	0x0	Reserved
14:12	**PRECHARGE_FSM**	RO	0x0	Snapshot of the state machine that controls the transmit of PRECHARGE.
11	**RSRVD1**	RO	0x0	Reserved
14:12	**SHIFT_FSM**	RO	0x0	Snapshot of the state machine that controls the loading of the shift register.
7:5	**RSRVD0**	RO	0x0	Reserved
4:0	**SOF_FSM**	RO	0x0	Snapshot of the state machine that controls the long transmission of the EOP for SOF.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.13. UTMI Debug Status Register 6 Description

The UTMI Debug Status Register 6 holds the observation of internal state machines of the high-speed transmit. This includes the first EOP FSM and EOP FSM.

HW_USBPHY_DEBUG6_STATUS 0x8007c0c0

Table 206. HW_USBPHY_DEBUG6_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD0																					FIRST_EOP_FSM			EOP_FSM							

Table 207. HW_USBPHY_DEBUG6_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:11	**RSRVD0**	RO	0x0	Reserved

Table 207. HW_USBPHY_DEBUG6_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
10:8	**FIRST_EOP_FSM**	RO	0x0	Snapshot of the state machine that determines transition of data to eop.
7:0	**EOP_FSM**	RO	0x0	Snapshot of the state machine that controls the transmit of EOP.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.14. UTMI Debug Status Register 7 Description

The UTMI Debug Status Register 7 holds the observation of internal state machines of the full-speed receive. This includes the first EOP FSM, main FSM, FIFO control FSM, load FIFO FSM, unstuff count, bit count, first data FSM, and EOP FSM.

HW_USBPHY_DEBUG7_STATUS 0x8007c0d0

Table 208. HW_USBPHY_DEBUG7_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3		FIRST_DATA_FSM		BIT_CNT				RSRVD2	UNSTUFF_CNT			RSRVD1		LD_FSM		RSRVD0		FIFO_FSM						MAIN_FSM				EOP_FSM			

Table 209. HW_USBPHY_DEBUG7_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**RSRVD3**	RO	0x0	Reserved
29:28	**FIRST_DATA_FSM**	RO	0x0	Snapshot of the state machine that determines transition of sync to data.
27:24	**BIT_CNT**	RO	0x0	Snapshot of the bit count.
23	**RSRVD2**	RO	0x0	Reserved
22:20	**UNSTUFF_CNT**	RO	0x0	Snapshot of the unstuff bit count.
19:18	**RSRVD1**	RO	0x0	Reserved
17:16	**LD_FSM**	RO	0x0	Snapshot of the FIFO load FSM.
15:14	**RSRVD0**	RO	0x0	Reserved
13:8	**FIFO_FSM**	RO	0x0	Snapshot of the FIFO FSM.
7:4	**MAIN_FSM**	RO	0x0	Snapshot of the main FSM.
3:0	**EOP_FSM**	RO	0x0	Snapshot of the transmission of EOP FSM.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

9.6.15. UTMI Debug Status Register 8 Description

The UTMI Debug Status Register 8 holds the observation of internal state machines of the full-speed transmit and SIE state machines. This includes the full-speed main transmit FSM, full-speed shift FSM, receive SIE FSM, and transmit SIE FSM.

HW_USBPHY_DEBUG8_STATUS 0x8007c0e0

Table 210. HW_USBPHY_DEBUG8_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RX_SIE_FSM				TX_SIE_FSM				RSRVD2														SHIFT_FSM		RSRVD1	FS_TX_MAIN_FSM						

Table 211. HW_USBPHY_DEBUG8_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RX_SIE_FSM**	RO	0x0	Snapshot of the SIE RX FSM.
27:24	**TX_SIE_FSM**	RO	0x0	Snapshot of the SIE TX FSM.
23:10	**RSRVD2**	RO	0x0	Reserved
9:8	**SHIFT_FSM**	RO	0x0	Snapshot of the transmit shift FSM.
7	**RSRVD1**	RO	0x0	Reserved
6:0	**FS_TX_MAIN_FSM**	RO	0x0	Snapshot of the FIFO load FSM.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

USBPHY XML Revision: 1.54

10. AHB-TO-APBH BRIDGE WITH DMA

This chapter describes the AHB-to-APBH bridge on the STMP36xx, along with its central DMA function and implementation examples. Programmable registers are described in Section 10.5.

10.1. Overview

The AHB-to-APBH bridge provides the STMP36xx with an inexpensive peripheral attachment bus running on the AHB's HCLK. (The "H" in APBH denotes that the APB*H* is synchronous to HCLK, as compared to APB*X*, which runs on the crystal-derived XCLK.)

As shown in Figure 32, the AHB-to-APBH bridge includes the AHB-to-APB PIO bridge for memory-mapped I/O to the APB devices, as well as a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM926 core. Each one of the APB peripherals, including the vectored interrupt controller, are documented in their own chapters elsewhere in this document.

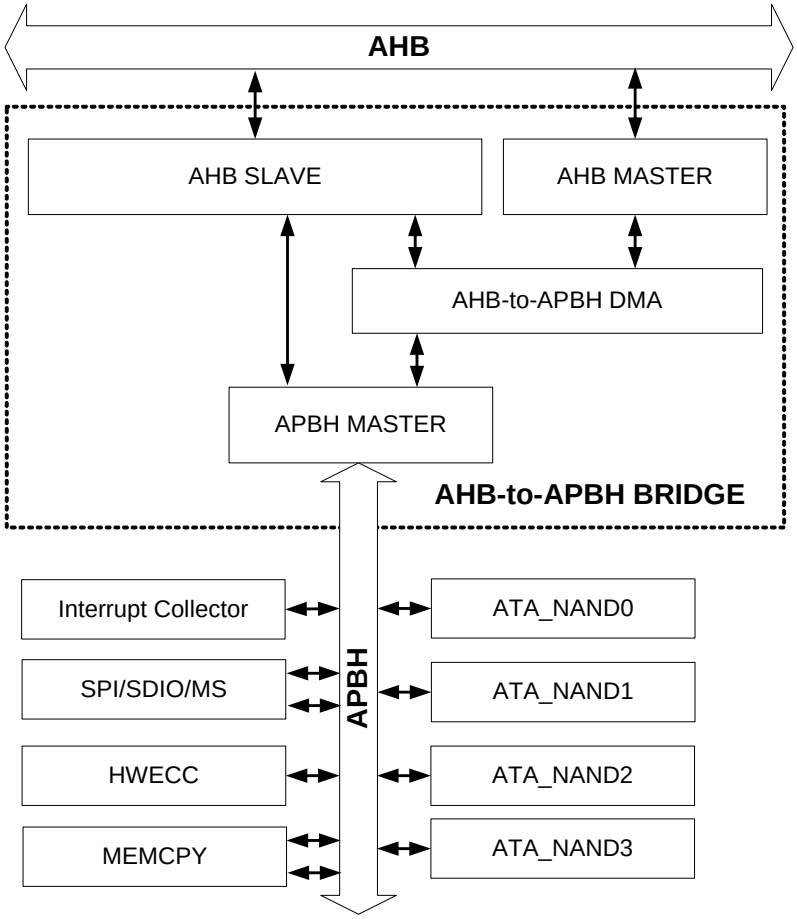


Figure 32. AHB-to-APBH Bridge DMA Block Diagram

The DMA controller uses the APBH bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBH bus and the AHB-to-APB bridge functions' use of the APBH is mediated by internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report "not ready" via its HREADY output until the bridge transfer can complete. The arbiter tracks repeated lockouts and inverts the priority, guaranteeing the CPU every fourth transfer on the APB.

10.2. AHBH DMA

The DMA supports eight channels of DMA services, as shown in Table 212. The shared DMA resource allows each independent channel to follow a simple chained command list. Command chains are built up using the general structure, as shown in Figure 33.

Table 212. APBH DMA Channel Assignments

APBH DMA CHANNEL #	USAGE
0	Hardware ECC Controller
1	SPI/SDIO/MS Controller
2	Memory Copy Source
3	Memory Copy Destination
4	ATA_NAND_DEVICE0
5	ATA_NAND_DEVICE1
6	ATA_NAND_DEVICE2
7	ATA_NAND_DEVICE3

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the CPU can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA, and have no further concern for the device until the DMA completion interrupt occurs. The goal here, as with the entire design of the STMP36xx, is to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1-kHz (arrival intervals longer than one ms).

Thus, a single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and controls that it uses to write DMA data bytes to the device.

For example, this allows a chain of operations to be issued to the ATANAND controller to send NAND command bytes, address bytes, and data transfers where the command and address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA.

Each DMA structure can have from 0 to 15 PIO words appended to it. The #PIOWORDs field, if non-zero, instructs the DMA engine to copy these words to the APB, beginning at PADDR = 0x0000 and incrementing its PADDR for each cycle.

During these operations, the DMA drives PSEL corresponding to the device associated to the DMA channel. The PSEL-to-DMA channel association is defined at synthesis time in the STMP36xx. Subsequent generations might choose to implement selectable associations for limited cases.

The DMA master only generates normal write transfers to the APBH. It does *not* generate SCT set, clear, or toggle transfers.

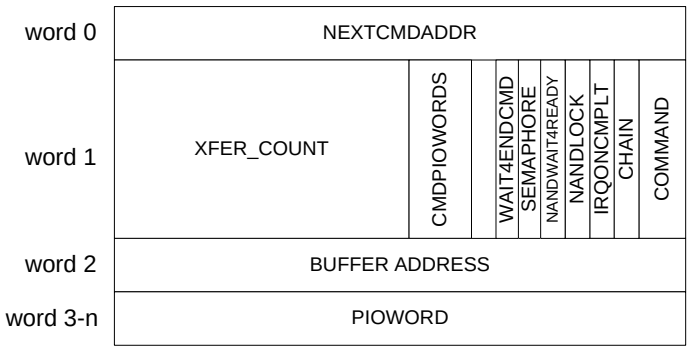


Figure 33. AHB-to-APBH Bridge DMA Channel Command Structure

Once any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. Table 213 shows the four commands implemented by the DMA.

Table 213. APBH DMA Commands

DMA COMMAND	USAGE
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.
11	DMA_SENSE. Perform any requested PIO word transfers, then perform a conditional branch to the next chained device. Follow the NEXTCMD_ADDR pointer if the peripheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true. This command becomes a no-operation for any channel other than a GPMI channel.

DMA_WRITE operations copy data bytes to system memory (on-chip RAM or SDRAM) from the associated peripheral. Each peripheral has a target PADDR value that it expects to receive DMA bytes. This association is synthesized in the DMA. The DMA_WRITE transfer uses the BUFFER_ADDDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers. This command is useful in such applications as activating the ATANAND devices CHECKSTATUS operation. The check status command in the ATANAND peripheral reads a status byte from the NAND device, performs an XOR and MASK against an expected value supplied as part of the PIO transfer. Once the read check completes (see Section 10.3.2), the NO_DMA_XFER command completes. The result in the peripheral is that its PSENSE line is driven by the results of the comparison. The sense flip-flop is only updated by CHECKSTATUS for the device that is executed. At some future point, the chain contains a DMA command structure with the fourth and final command value, i.e., the DMA_SENSE command.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. The DMA_SENSE command uses the DMA buffer pointer word of the command structure in a slightly different way. Namely, it points to an alternate DMA command structure chain or list. The DMA_SENSE command examines the sense line of the associated peripheral. If the sense line is “true,” then the DMA follows the standard list found whose next command is found from the pointer in the NEXTCMD_ADDR word of the command structure. If the sense line is “false,” then the DMA follows the alternate list whose next command is found from the pointer in the DMA Buffer Pointer word of the DMA_SENSE command structure (see Figure 35). The sense command ignores the CHAIN bit, so that both pointers must be valid when the DMA comes to sense command.

If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel waits for the device to signal completion of a command by toggling the APX_ENDCMCD signal before proceeding to load and execute the next command structure. The semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in Table 214, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer-count mechanism is duplicated in the associated peripheral, either as an implied or specified count in the peripheral. For example, the HWECC peripheral uses an implied size of 256 bytes for the SSFDC parity block size, while the ATANAND controller has a corresponding programmable field to specify the number of bytes to transfer at the interface.

Table 214. DMA Channel Command Word in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															
NUMBER DMA BYTES TO TRANSFER																NUMBER PIO WORDS TO WRITE								WAIT4ENDCMD	DECREMENT SEMAPHORE	NANDwAIT4READY	NANDLOCK	IRQ_COMPLETE	CHAIN	COMMAND	
DMA BUFFER or ALTERNATE CCW																															
ZERO OR MORE PIO WORDS TO WRITE TO THE ASSOCIATED PERIPHERAL STARTING AT ITS BASE ADDRESS ON THE APBH BUS																															

Figure 35 also shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to one if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (zero) is loaded into the NEXT_COMMAND_ADDRESS, it will not be detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_FINISH bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by software. It can be used to interrupt the CPU.

The NAND_LOCK bit is monitored by the DMA channel arbiter. Once a NAND channel ([7:4]) succeeds in the arbiter with its NAND_LOCK bit set, then the arbiter will ignore the other three NAND channels until a command is completed in which the NAND_LOCK is not set. Notice that the semantic here is that the NAND_LOCK state is to limit scheduling of a non-locked DMA. A DMA channel can go from unlocked to locked in the arbiter at the beginning of a command when the NAND_LOCK bit is set. When the last DMA command of an atomic sequence is completed, the lock should be removed. To accomplish this, the last command does not have the NAND_LOCK bit. It is still locked in the atomic state within the arbiter when the command starts, so that it is the only NAND command that can be executed. At the end, it drops from the atomic state within the arbiter.

The NAND_WAIT4READY bit also has a special use for DMA channels [7:4], i.e., the ATANAND device channels. The ATANAND device supplies a sample of the ready line for the NAND device. This ready value is used to hold off of a command with this bit set until the ready line is asserted to one. Once the arbiter sees a command with a wait-for-ready set, it holds off that channel until ready is asserted.

NOTE:
In ATA mode, you must set both HW_APBH_CHn_CMD(4).NANDWAIT4READY and HW_APBH_CHn_CMD(4).WAIT4ENDCMD to get the desired behavior. If you set only HW_APBH_CHn_CMD(4).WAIT4ENDCMD in the DMA command set and set HW_GPMI_CTRL0.WAIT_FOR_READY in the GPMI command, but do not set

HW_APBH_CHn_CMD(4).NANDWAIT4READY, the DMA channel will continue without waiting for the interrupt.

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run and process commands and DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to zero as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by software. When the semaphore goes to non-zero and the channel is in its IDLE state, then it uses the value in the HW_APBHn_NXTCMDAR (next command address register) to fetch a pointer to the next command to process. NOTE: This is a double indirect case. This method allows software to append to a running command list under the protection of the counting semaphore.

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the HW_APBHn_NXTCMDAR register, and then writes a one to the counting semaphore in HW_APBHn_SEMA. The DMA channel loads HW_APBHn_CURCMDAR register and then enters the normal state machine processing for the next command. When software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of HW_APBHn_CURCMDAR at any time to determine the location of the command structure currently being processed.

10.3. Implementation Examples

10.3.1. HWECC Example Command Chain

The example in Figure 34 shows how to bring the basic items together to make a simple DMA chain to read and check a HW_ECC Reed-Solomon error-correction block using one DMA channel. This example shows three command structures linked together using their normal command-list pointers.

- The first command writes a single PIO word to the PIOWORD register, selecting the RS decode operation, etc. This first command also performs a 512-byte DMA_READ operation to read the data block bytes into the HWECC, where its error is checked. (Actually two passes are required; see Chapter 14.)
- A second DMA command structure also performs a DMA_READ operation. This time to read the nine parity bytes.
- When the HWECC computes its error-correction information, it writes a fixed-size nine-word error report structure to system memory, using the third DMA command structure to perform a DMA_WRITE operation.

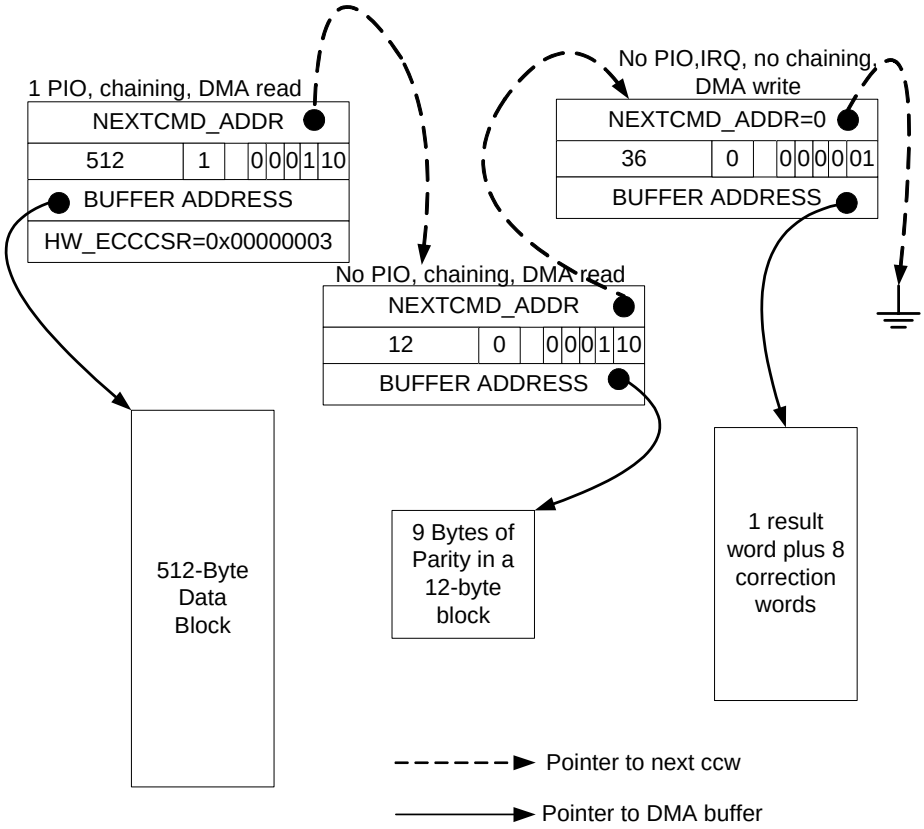


Figure 34. AHB-to-APBH Bridge DMA HWECC Example Command Chain

10.3.2. NAND Read Status Polling Example

Figure 35 shows a more complicated scenario. This subset of a NAND device workload shows that the first two command structures are used during the data-write phase of an ATANAND device write operation (CLE and ALE transfers omitted for clarity).

- After writing the data, one must wait until the NAND device status register indicates that the write charge has been transferred. This is built into the workload using a check status command in the NAND in a loop created from the next two DMA command structures.
- The NO_DMA_TRANSFER command is shown here performing the read check, followed by a DMA_SENSE command to branch the DMA command structure list, based on the status of a bit in the external NAND device.

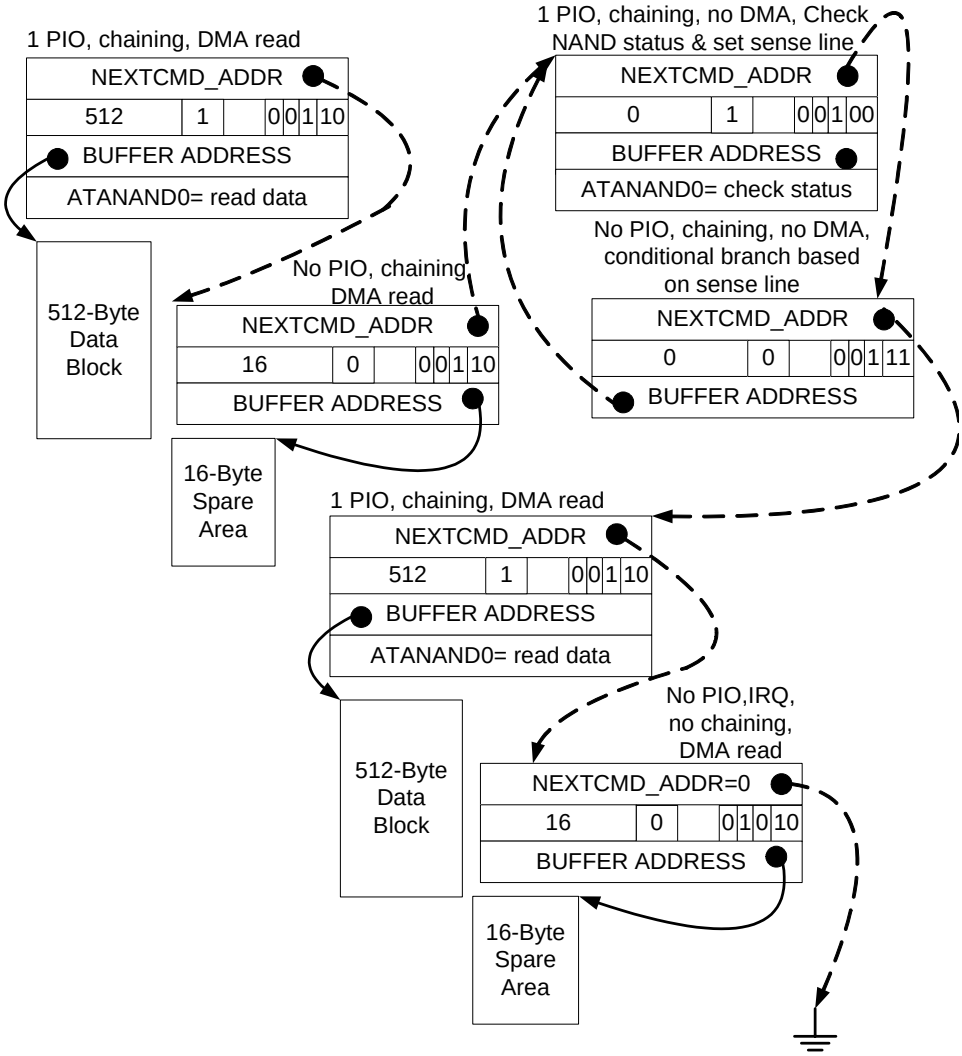


Figure 35. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command

The example in Figure 35 shows the workload continuing immediately to the next NAND page transfer. However, one could perform a second sense operation to see if an error occurred after the write. One could then point the sense command alternate branch at a NO_DMA_XFER command with the interrupt bit set. If the CHAIN bit is not set on this failure branch, then the CPU is interrupted immediately, and the channel process is also immediately terminated in the presence of a workload-detected NAND error bit.

Note that each word of the three-word DMA command structure corresponds to a PIO register of the DMA that is accessible on the APBH bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register.

To start DMA processing for the first command, initialize the PIO registers of the desired channel, as follows:

- First, load the next command address register with a pointer to the first command to be loaded.
- Then, write a one to the counting semaphore register. This causes the DMA to schedule the targeted channel for DMA command structure load, as if it just finished its previous command.

10.3.3. APBH DMA and PIO Bus Implementation Example

Figure 36 shows an AHB-to-APB bridge device interface, and Table 215 defines the interface signals used in the implementation example.

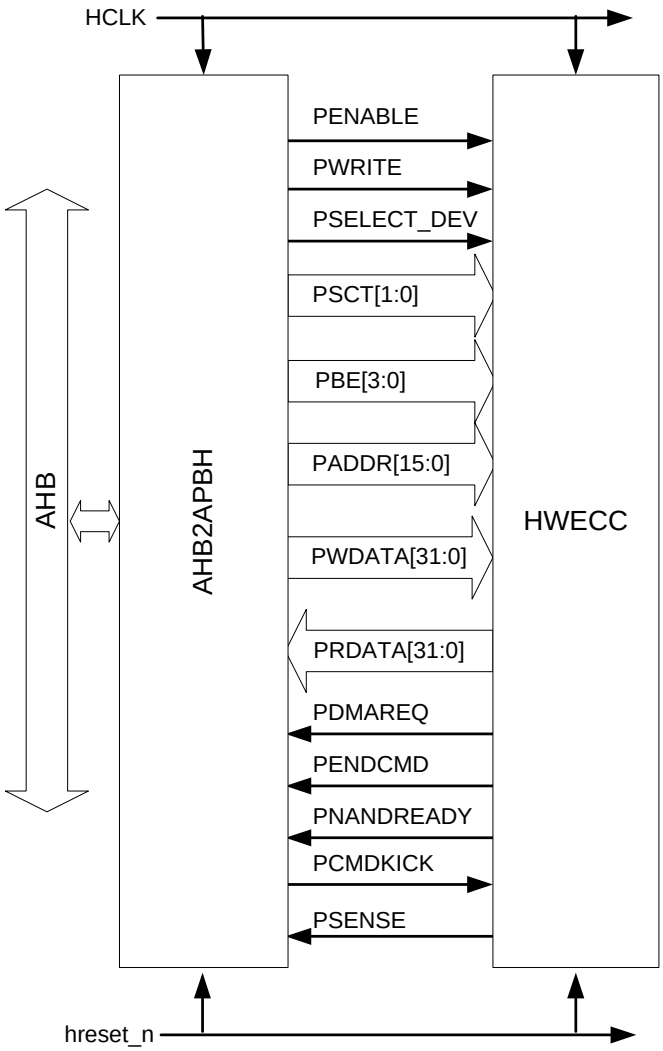


Figure 36. AHB-to-APB Bridge Device Interface

Table 215. APBH Interface Signals

SIGNAL	I/O	USAGE
PENABLE	O	Strobe signal used to time all accesses on the APBH bus. The rising edge of PENABLE indicates the beginning of an APBH transfer.
PWRITE	O	When high, indicates an APBH write access, and when low, a read access.
PSELECT_DEV[N:0]	O	One bit for each PIO device on the APBH bus. The bit corresponding to the selected device is driven high when PENABLE is asserted to select one of the PIO devices.
PSCT[1:0]	O	This two-bit field indicates whether the current bus is: 00—Normal Read, Normal Write Cycle 01—Normal Read, Set Bits Write Cycle 10—Normal Read, Clear Bits Write Cycle 11—Normal Read, Toggle Bits Write Cycle If the targeted PIO register does not support SCT special cycles, then a normal write cycle is performed by the device, even though a set/clear toggle cycle was requested. These values are based on AHB haddr[3:2].
PBE[3:0]	O	The byte-enable signals control which bytes of a 32-bit transfer are to be written for a store half or store byte operation. If a device does not support transfers of less than a word size, then it ignores this field. This is the situation for all IP peripherals.
PADDR[15:0]	O	The APBH address driven by the APBH bridge.
PWDATA[31:0]	O	The write data bus driven by the APBH bridge during write cycles (PWRITE is HIGH).
PRDATA[31:0]	I	The read data bus driven by the separate APBH device during read cycles (PWRITE is low).
PDMAREQ[7:0]	I	This signal is reset by the soft reset signal in each device. It is toggled once for each request. A device must not toggle this signal until it has been serviced by the DMA, a condition recognized by noting the PIO write or read cycles to its various DMA data ports. PDMAREQ[0] is driven by the HWECC. PDMAREQ[1] is driven by the SPI block. PDMAREQ[3:2] are driven by the mem_cpy device PDMAREQ[7:4] is driven by the NAND/ATA
PENDCMD[7:0]	I	The device toggles this line at the end of a sense command to notify the DMA that a WAIT4ENDCMD condition has been satisfied.
PCMDKICK[7:0]	O	The APB master produces a state toggle indicating that a device has been “kicked off” by the DMA command processing.
PDMASENSE[7:0]	I	Sense flag from each DMA peripheral, used in conditional branching within DMA chains. These signals are synchronous with the APBH or APBX clock as appropriate and are synchronized to HCLK within the DMA.
PNANDRDY[7:4]	I	The GPMI samples the state of the NAND device-ready line and presents it on the corresponding signal. The DMA arbiter synchronizes this signal and uses it to process the WAIT4NANDREADY condition.

10.4. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

10.5. Programmable Registers

This section describes the programmable registers of the AHB-to-APBH bridge block.

10.5.1. AHB-to-APBH Bridge Control and Status Register 0 Description

The AHB-to-APBH Bridge Control and Status Register 0 provides overall control of the AHB-to-APBH bridge and DMA.

HW_APBH_CTRL0 0x80004000
HW_APBH_CTRL0_SET 0x80004004
HW_APBH_CTRL0_CLR 0x80004008
HW_APBH_CTRL0_TOG 0x8000400C

Table 216. HW_APBH_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSVD2						RESET_CHANNEL								CLKGATE_CHANNEL								FREEZE_CHANNEL							

Table 217. HW_APBH_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set this bit to zero to enable normal APBH DMA operation. Set this bit to one (default) to disable clocking with the APBH DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBH DMA block to its default state.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29:24	**RSVD2**	RO	0x000000	Reserved, always set to zero.
23:16	**RESET_CHANNEL**	RW	0x0	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. HWECC = 0x01 SSP = 0x02 SRC = 0x04 DEST = 0x08 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x30 NAND3 = 0x40

Table 217. HW_APBH_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	**CLKGATE_CHANNEL**	RW	0x00	These bits must be set to zero for normal operation of each channel. When set to one, they gate off the individual clocks to the channels. HWECC = 0x01 SSP = 0x02 SRC = 0x04 DEST = 0x08 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x30 NAND3 = 0x40
7:0	**FREEZE_CHANNEL**	RW	0x0	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is denied access to the central DMA resources. HWECC = 0x01 SSP = 0x02 SRC = 0x04 DEST = 0x08 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x30 NAND3 = 0x40

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

10.5.2. AHB-to-APBH Bridge Control and Status Register 1 Description

The AHB-to-APBH Bridge Control and Status Register 1 provides overall control of the interrupts generated by the AHB-to-APBH DMA.

HW_APBH_CTRL1 0x80004010
HW_APBH_CTRL1_SET 0x80004014
HW_APBH_CTRL1_CLR 0x80004018
HW_APBH_CTRL1_TOG 0x8000401C

Table 218. HW_APBH_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN	RSVD1								CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ

Table 219. HW_APBH_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x000000	Reserved, always set to zero.
23	**CH7_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 7.
22	**CH6_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 6.
21	**CH5_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 5.
20	**CH4_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 4.
19	**CH3_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 3.
18	**CH2_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 2.
17	**CH1_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 1.
16	**CH0_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 0.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**CH7_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6	**CH6_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5	**CH5_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4	**CH4_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3	**CH3_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2	**CH2_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table 219. HW_APBH_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**CH1_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0	**CH0_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBH DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

DESCRIPTION:

This register contains the per-channel interrupt status bits and the per-channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE:

```
BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0); // use bitfield write macro
BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0; // or, assign to register struct's bitfield
```

10.5.3. AHB-to-APBH DMA Device Assignment Register Description

This register allows reassignment of the APBH device connected to the DMA channels.

HW_APBH_DEVSEL 0x80004020

Table 220. HW_APBH_DEVSEL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CH7				CH6				CH5				CH4				CH3				CH2				CH1				CH0			

Table 221. HW_APBH_DEVSEL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**CH7**	RO	0x0	Reserved.
27:24	**CH6**	RO	0x0	Reserved.
23:20	**CH5**	RO	0x0	Reserved.
19:16	**CH4**	RO	0x0	Reserved.
15:12	**CH3**	RO	0x0	Reserved.
11:8	**CH2**	RO	0x0	Reserved.
7:4	**CH1**	RO	0x0	Reserved.
3:0	**CH0**	RO	0x0	Reserved.

DESCRIPTION:

This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE:

```
Empty Example.
```

10.5.4. APBH DMA Channel 0 Current Command Address Register Description

The APBH DMA Channel 0 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH0_CURCMDAR 0x80004030

Table 222. HW_APBH_CH0_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 223. HW_APBH_CH0_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 0.

DESCRIPTION:

APBH DMA Channel 0 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
  pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR_RD(0); // read the whole register,
since there is only one field
  pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 0, CMD_ADDR); // or, use multi-
register bitfield read macro
  pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR(0).CMD_ADDR; // or, assign from bit-
field of indexed register's struct
```

10.5.5. APBH DMA Channel 0 Next Command Address Register Description

The APBH DMA Channel 0 Next Command Address Register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

HW_APBH_CH0_NXTCMDAR 0x80004040

Table 224. HW_APBH_CH0_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 225. HW_APBH_CH0_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 0.

DESCRIPTION:

APBH DMA Channel 0 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
  HW_APBH_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure); // write the entire register,
since there is only one field
   BF_WRn(APBH_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure); // or, use multi-register
bitfield write macro
  HW_APBH_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bit-
field of indexed register's struct
```

10.5.6. APBH DMA Channel 0 Command Register Description

The APBH DMA Channel 0 Command Register specifies the DMA transaction to perform for the current command chain item.

HW_APBH_CH0_CMD 0x80004050

Table 226. HW_APBH_CH0_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 227. HW_APBH_CH0_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the HWECC device HW_HWECC_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the HWECC, starting with the base PIO address of the HWECC (HW_HWECC_CTRL) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.

Table 227. HW_APBH_CH0_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will wait until the ATA/NAND device reports 'ready' before executing the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH0_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the HWECC (APB PIO Read) to the system memory (AHB master write) 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
hw_apbh_chn_cmd_t dma_cmd;
dma_cmd.XFER_COUNT = 512; // transfer 512 bytes
```

```
  dma_cmd.COMMAND = BV_APBH_CHn_CMD_COMMAND__DMA_WRITE; // transfer to system memory from
peripheral device
  dma_cmd.CHAIN = 1; // chain an additional command structure on to the list
  dma_cmd.IRQONCMPLT = 1; // generate an interrupt on completion of this command structure
```

10.5.7. APBH DMA Channel 0 Buffer Address Register Description

The APBH DMA Channel 0 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH0_BAR 0x80004060

Table 228. HW_APBH_CH0_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 229. HW_APBH_CH0_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
hw_apbh_chn_bar_t dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

10.5.8. APBH DMA Channel 0 Semaphore Register Description

The APBH DMA Channel 0 Semaphore Register is used to synchronize the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH0_SEMA 0x80004070

Table 230. HW_APBH_CH0_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 231. HW_APBH_CH0_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
BF_WR(APBH_CHn_SEMA, 0, INCREMENT_SEMA, 2); // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 0, PHORE); // get instantaneous value
```

10.5.9. AHB-to-APBH DMA Channel 0 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 0 state machine and controls.

HW_APBH_CH0_DEBUG1 0x80004080

Table 232. HW_APBH_CH0_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 233. HW_APBH_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 233. HW_APBH_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 0 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 0.

EXAMPLE:

```
Empty example.
```

10.5.10. AHB-to-APBH DMA Channel 0 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 0.

HW_APBH_CH0_DEBUG2 0x80004090

Table 234. HW_APBH_CH0_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 235. HW_APBH_CH0_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 0.

EXAMPLE:

```
Empty example.
```

10.5.11. APBH DMA Channel 1 Current Command Address Register Description

The APBH DMA Channel 1 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH1_CURCMDAR 0x800040A0

Table 236. HW_APBH_CH1_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 237. HW_APBH_CH1_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 1.

DESCRIPTION:

APBH DMA Channel 1 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
  pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR_RD(1); // read the whole register,
since there is only one field
```

```
  pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 1, CMD_ADDR); // or, use multi-
register bitfield read macro
  pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR(1).CMD_ADDR; // or, assign from bit-
field of indexed register's struct
```

10.5.12. APBH DMA Channel 1 Next Command Address Register Description

The APBH DMA Channel 1 Next Command Address Register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

HW_APBH_CH1_NXTCMDAR 0x800040B0

Table 238. HW_APBH_CH1_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 239. HW_APBH_CH1_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 1.

DESCRIPTION:

APBH DMA Channel 1 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 1 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
  HW_APBH_CHn_NXTCMDAR_WR(1, (reg32_t) pCommandTwoStructure); // write the entire register,
since there is only one field
   BF_WRn(APBH_CHn_NXTCMDAR, 1, (reg32_t) pCommandTwoStructure); // or, use multi-register
bitfield write macro
  HW_APBH_CHn_NXTCMDAR(1).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bit-
field of indexed register's struct
```

10.5.13. APBH DMA Channel 1 Command Register Description

The APBH DMA Channel 1 Command Register specifies the cycle to perform for the current command chain item.

HW_APBH_CH1_CMD 0x800040C0

Table 240. HW_APBH_CH1_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 241. HW_APBH_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SSP device HW_SSP_DATA register. A value of 0 indicates a 64-Kbyte transfer size.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the SSP, starting with the base PIO address of the SSP (HW_SSP_CTRL0) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.

Table 241. HW_APBH_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH1_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the SSP (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty Example.
```

10.5.14. APBH DMA Channel 1 Buffer Address Register Description

The APBH DMA Channel 1 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH1_BAR 0x800040D0

Table 242. HW_APBH_CH1_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 243. HW_APBH_CH1_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
hw_apbh_chn_bar_t dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

10.5.15. APBH DMA Channel 1 Semaphore Register Description

The APBH DMA Channel 1 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH1_SEMA 0x800040E0

Table 244. HW_APBH_CH1_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 245. HW_APBH_CH1_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
BF_WR(APBH_CHn_SEMA, 1, INCREMENT_SEMA, 2); // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 1, PHORE); // get instantaneous value
```

10.5.16. AHB-to-APBH DMA Channel 1 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 1 state machine and controls.

HW_APBH_CH1_DEBUG1 0x800040F0

Table 246. HW_APBH_CH1_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 247. HW_APBH_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA channel's Write FIFO Full signal.

Table 247. HW_APBH_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 1 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 1.

EXAMPLE:

```
Empty example.
```

10.5.17. AHB-to-APBH DMA Channel 1 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

HW_APBH_CH1_DEBUG2 0x80004100

Table 248. HW_APBH_CH1_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 249. HW_APBH_CH1_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 1.

EXAMPLE:

```
Empty example.
```

10.5.18. APBH DMA Channel 2 Current Command Address Register Description

The APBH DMA Channel 2 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH2_CURCMDAR 0x80004110

Table 250. HW_APBH_CH2_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 251. HW_APBH_CH2_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 2.

DESCRIPTION:

APBH DMA Channel 2 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.19. APBH DMA Channel 2 Next Command Address Register Description

The APBH DMA Channel 2 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH2_NXTCMDAR 0x80004120

Table 252. HW_APBH_CH2_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 253. HW_APBH_CH2_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 2.

DESCRIPTION:

APBH DMA Channel 2 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 2 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty Example.
```

10.5.20. APBH DMA Channel 2 Command Register Description

The APBH DMA Channel 2 Command Register specifies the cycle to perform for the current command chain item.

HW_APBH_CH2_CMD 0x80004130

Table 254. HW_APBH_CH2_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 255. HW_APBH_CH2_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the MEMCPY device HW_MEMCPY_DATA register. A value of 0 indicates a 64-Kbyte transfer size.
15:12	**CMDWORDS**	RO	0x00	This field contains the number of command words to send to the MEMCPY, starting with the base PIO address of the MEMCPY (HW_MEMCPY_CTRL) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.

Table 255. HW_APBH_CH2_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH2_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBH device (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.21. APBH DMA Channel 2 Buffer Address Register Description

The APBH DMA Channel 2 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH2_BAR 0x80004140

Table 256. HW_APBH_CH2_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 257. HW_APBH_CH2_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.22. APBH DMA Channel 2 Semaphore Register Description

The APBH DMA Channel 2 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH2_SEMA 0x80004150

Table 258. HW_APBH_CH2_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 259. HW_APBH_CH2_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.23. AHB-to-APBH DMA Channel 2 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 2 state machine and controls.

HW_APBH_CH2_DEBUG1 0x80004160

Table 260. HW_APBH_CH2_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE			

Table 261. HW_APBH_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA channel's Write FIFO Full signal.

Table 261. HW_APBH_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 2 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 2.

EXAMPLE:

```
Empty example.
```

10.5.24. AHB-to-APBH DMA Channel 2 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 2.

HW_APBH_CH2_DEBUG2 0x80004170

Table 262. HW_APBH_CH2_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 263. HW_APBH_CH2_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 2.

EXAMPLE:

```
Empty example.
```

10.5.25. APBH DMA Channel 3 Current Command Address Register Description

The APBH DMA Channel 3 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH3_CURCMDAR 0x80004180

Table 264. HW_APBH_CH3_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 265. HW_APBH_CH3_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 3.

DESCRIPTION:

APBH DMA Channel 3 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.26. APBH DMA Channel 3 Next Command Address Register Description

The APBH DMA Channel 3 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH3_NXTCMDAR 0x80004190

Table 266. HW_APBH_CH3_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 267. HW_APBH_CH3_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 3.

DESCRIPTION:

APBH DMA Channel 3 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 3 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.27. APBH DMA Channel 3 Command Register Description

The APBH DMA Channel 3 Command Register specifies the cycle to perform for the current command chain item.

HW_APBH_CH3_CMD 0x800041A0

Table 268. HW_APBH_CH3_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 269. HW_APBH_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the MEMCPY device HW_MEMCPY_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the MEMCPY, starting with the base PIO address of the MEMCPY (HW_MEMCPY_CTRL) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.

Table 269. HW_APBH_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH3_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBH device (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.28. APBH DMA Channel 3 Buffer Address Register Description

The APBH DMA Channel 3 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH3_BAR 0x800041B0

Table 270. HW_APBH_CH3_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 271. HW_APBH_CH3_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.29. APBH DMA Channel 3 Semaphore Register Description

The APBH DMA Channel 3 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH3_SEMA 0x800041C0

Table 272. HW_APBH_CH3_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 273. HW_APBH_CH3_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.30. AHB-to-APBH DMA Channel 3 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 3 state machine and controls.

HW_APBH_CH3_DEBUG1 0x800041D0

Table 274. HW_APBH_CH3_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE			

Table 275. HW_APBH_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflect the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflect the current state of the DMA channel's Write FIFO Full signal.

Table 275. HW_APBH_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 3 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 3.

EXAMPLE:

```
Empty example.
```

10.5.31. AHB-to-APBH DMA Channel 3 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 3.

HW_APBH_CH3_DEBUG2 0x800041E0

Table 276. HW_APBH_CH3_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 277. HW_APBH_CH3_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 3.

EXAMPLE:

```
Empty example.
```

10.5.32. APBH DMA Channel 4 Current Command Address Register Description

The APBH DMA Channel 4 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH4_CURCMDAR 0x800041F0

Table 278. HW_APBH_CH4_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 279. HW_APBH_CH4_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 4.

DESCRIPTION:

APBH DMA Channel 4 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.33. APBH DMA Channel 4 Next Command Address Register Description

The APBH DMA Channel 4 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH4_NXTCMDAR 0x80004200

Table 280. HW_APBH_CH4_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 281. HW_APBH_CH4_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 4.

DESCRIPTION:

APBH DMA Channel 4 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 4 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.34. APBH DMA Channel 4 Command Register Description

The APBH DMA Channel 4 Command Register specifies the cycle to perform for the current command chain item.

HW_APBH_CH4_CMD 0x80004210

Table 282. HW_APBH_CH4_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 283. HW_APBH_CH4_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_0 device HW_GPMI_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.

Table 283. HW_APBH_CH4_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH4_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.35. APBH DMA Channel 4 Buffer Address Register Description

The APBH DMA Channel 4 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH4_BAR 0x80004220

Table 284. HW_APBH_CH4_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 285. HW_APBH_CH4_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.36. APBH DMA Channel 4 Semaphore Register Description

The APBH DMA Channel 4 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH4_SEMA 0x80004230

Table 286. HW_APBH_CH4_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 287. HW_APBH_CH4_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.37. AHB-to-APBH DMA Channel 4 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 4 state machine and controls.

HW_APBH_CH4_DEBUG1 0x80004240

Table 288. HW_APBH_CH4_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 289. HW_APBH_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense signal sent from the APB GPMI device.
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready signal sent from the APB GPMI device.
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA channel lock for a GPMI channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.

Table 289. HW_APBH_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO display of the DMA Channel 4 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 4.

EXAMPLE:

```
Empty example.
```

10.5.38. AHB-to-APBH DMA Channel 4 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

HW_APBH_CH4_DEBUG2 0x80004250

Table 290. HW_APBH_CH4_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 291. HW_APBH_CH4_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 4.

EXAMPLE:

```
Empty example.
```

10.5.39. APBH DMA Channel 5 Current Command Address Register Description

The APBH DMA Channel 5 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH5_CURCMDAR 0x80004260

Table 292. HW_APBH_CH5_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 293. HW_APBH_CH5_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 5.

DESCRIPTION:

APBH DMA Channel 5 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.40. APBH DMA Channel 5 Next Command Address Register Description

The APBH DMA Channel 5 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH5_NXTCMDAR 0x80004270

Table 294. HW_APBH_CH5_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 295. HW_APBH_CH5_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 5.

DESCRIPTION:

APBH DMA Channel 5 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 5 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.41. APBH DMA Channel 5 Command Register Description

The APBH DMA Channel 5 Command Register specifies the cycle to perform for the current command chain item.

HW_APBH_CH5_CMD 0x80004280

Table 296. HW_APBH_CH5_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 297. HW_APBH_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_1 device HW_GPMI_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.

Table 297. HW_APBH_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH5_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.42. APBH DMA Channel 5 Buffer Address Register Description

The APBH DMA Channel 5 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH5_BAR 0x80004290

Table 298. HW_APBH_CH5_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 299. HW_APBH_CH5_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.43. APBH DMA Channel 5 Semaphore Register Description

The APBH DMA Channel 5 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH5_SEMA 0x800042A0

Table 300. HW_APBH_CH5_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 301. HW_APBH_CH5_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.44. AHB-to-APBH DMA Channel 5 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 5 state machine and controls.

HW_APBH_CH5_DEBUG1 0x800042B0

Table 302. HW_APBH_CH5_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE				

Table 303. HW_APBH_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense signal sent from the APB GPMI device.
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready signal sent from the APB GPMI device.
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA channel lock for a GPMI channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.

Table 303. HW_APBH_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 5 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 5.

EXAMPLE:

```
Empty example.
```

10.5.45. AHB-to-APBH DMA Channel 5 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

HW_APBH_CH5_DEBUG2 0x800042C0

Table 304. HW_APBH_CH5_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 305. HW_APBH_CH5_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 5.

EXAMPLE:

```
Empty example.
```

10.5.46. APBH DMA Channel 6 Current Command Address Register Description

The APBH DMA Channel 6 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH6_CURCMDAR 0x800042D0

Table 306. HW_APBH_CH6_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 307. HW_APBH_CH6_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 6.

DESCRIPTION:

APBH DMA Channel 6 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.47. APBH DMA Channel 6 Next Command Address Register Description

The APBH DMA Channel 6 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH6_NXTCMDAR 0x800042E0

Table 308. HW_APBH_CH6_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 309. HW_APBH_CH6_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 6.

DESCRIPTION:

APBH DMA Channel 6 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 6 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.48. APBH DMA Channel 6 Command Register Description

The APBH DMA Channel 6 Command Register specifies the cycle to perform for the current command chain item.

HW_APBH_CH6_CMD 0x800042F0

Table 310. HW_APBH_CH6_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 311. HW_APBH_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_2 device HW_GPMI_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.

Table 311. HW_APBH_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH6_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.49. APBH DMA Channel 6 Buffer Address Register Description

The APBH DMA Channel 6 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH6_BAR 0x80004300

Table 312. HW_APBH_CH6_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 313. HW_APBH_CH6_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.50. APBH DMA Channel 6 Semaphore Register Description

The APBH DMA Channel 6 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH6_SEMA 0x80004310

Table 314. HW_APBH_CH6_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 315. HW_APBH_CH6_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.51. AHB-to-APBH DMA Channel 6 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 6 state machine and controls.

HW_APBH_CH6_DEBUG1 0x80004320

Table 316. HW_APBH_CH6_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 317. HW_APBH_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense signal sent from the APB GPMI device.
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready signal sent from the APB GPMI device.
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA channel lock for a GPMI channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.

Table 317. HW_APBH_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 6 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 6.

EXAMPLE:

```
Empty example.
```

10.5.52. AHB-to-APBH DMA Channel 6 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

HW_APBH_CH6_DEBUG2 0x80004330

Table 318. HW_APBH_CH6_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 319. HW_APBH_CH6_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 6.

EXAMPLE:

```
Empty example.
```

10.5.53. APBH DMA Channel 7 Current Command Address Register Description

The APBH DMA Channel 7 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBH_CH7_CURCMDAR 0x80004340

Table 320. HW_APBH_CH7_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 321. HW_APBH_CH7_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 7.

DESCRIPTION:

APBH DMA Channel 7 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

10.5.54. APBH DMA Channel 7 Next Command Address Register Description

The APBH DMA Channel 7 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBH_CH7_NXTCMDAR 0x80004350

Table 322. HW_APBH_CH7_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 323. HW_APBH_CH7_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 7.

DESCRIPTION:

APBH DMA Channel 7 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 7 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

10.5.55. APBH DMA Channel 7 Command Register Description

The APBH DMA Channel 7 Command Register specifies the cycle to perform for the current command chain item.

HW_APBH_CH7_CMD 0x80004360

Table 324. HW_APBH_CH7_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	

Table 325. HW_APBH_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI ATANAND_3 device HW_GPMI_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the GPMI, starting with the base PIO address of the GPMI (HW_GPMI_CTRL0) and incrementing from there. Zero means transfer NO command words.
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5	**NANDWAIT4READY**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will will wait until the ATA/NAND device reports 'ready' before execute the command. It is ignored for non-ATA/NAND DMA channels.
4	**NANDLOCK**	RO	0x0	A value of one indicates that the ATA/NAND DMA channel will remain "locked" in the arbiter at the expense of other ATA/NAND DMA channels. It is ignored for non-ATA/NAND DMA channels.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.

Table 325. HW_APBH_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH7_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the GPMI (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- Sense NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. DMA_SENSE = 0x3 Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.

DESCRIPTION:

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

10.5.56. APBH DMA Channel 7 Buffer Address Register Description

The APBH DMA Channel 7 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address, which means transfers can start on any byte boundary.

HW_APBH_CH7_BAR 0x80004370

Table 326. HW_APBH_CH7_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 327. HW_APBH_CH7_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

10.5.57. APBH DMA Channel 7 Semaphore Register Description

The APBH DMA Channel 7 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBH_CH7_SEMA 0x80004380

Table 328. HW_APBH_CH7_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 329. HW_APBH_CH7_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

10.5.58. AHB-to-APBH DMA Channel 7 Debug Register 1 Description

This register gives debug visibility into the APBH DMA Channel 7 state machine and controls.

HW_APBH_CH7_DEBUG1 0x80004390

Table 330. HW_APBH_CH7_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE				

Table 331. HW_APBH_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27	**SENSE**	RO	0x0	This bit reflects the current state of the GPMI Sense signal sent from the APB GPMI device.
26	**READY**	RO	0x0	This bit reflects the current state of the GPMI Ready signal sent from the APB GPMI device.
25	**LOCK**	RO	0x0	This bit reflects the current state of the DMA channel lock for a GPMI channel.
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.

Table 331. HW_APBH_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 7 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 7.

EXAMPLE:

```
Empty example.
```

10.5.59. AHB-to-APBH DMA Channel 7 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 7.

HW_APBH_CH7_DEBUG2 0x800043A0

Table 332. HW_APBH_CH7_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 333. HW_APBH_CH7_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 7.

EXAMPLE:

```
Empty example.
```

APBH XML Revision: 1.58

11. AHB-TO-APBX BRIDGE WITH DMA

This chapter describes the AHB-to-APBX bridge on the STMP36xx, along with its central DMA function and implementation examples. Programmable registers are described in Section 11.5.

11.1. Overview

The AHB-to-APBX bridge provides the STMP36xx with an inexpensive peripheral attachment bus running on the AHB's XCLK. (The "X" in APBX denotes that the APB*X* runs on a crystal-derived clock, as compared to APB*H*, which is synchronous to HCLK.)

As shown in Figure 37, the AHB-to-APBX bridge includes the AHB-to-APB PIO bridge for memory-mapped I/O to the APB devices, as well a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM926 core. Each one of the APB peripherals are documented in their own chapters elsewhere in this document.

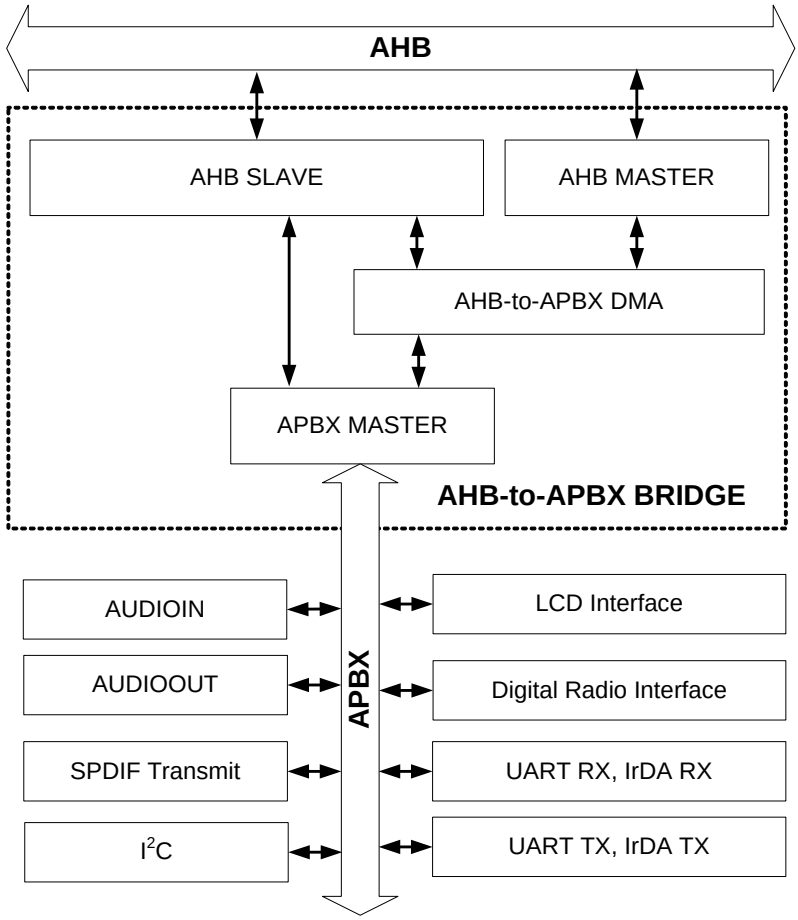


Figure 37. AHB-to-APBX Bridge DMA Block Diagram

The DMA controller uses the APBX bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBX bus and AHB-to-APB bridge functions' use of the APBX is mediated by internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report not ready via its HREADY output until the bridge transfer completes. The arbiter tracks repeated lockouts and inverts the priority, so that the CPU is guaranteed every fourth transfer on the APB.

11.2. APBX DMA

The DMA supports eight channels of DMA services as shown in Table 334. The shared DMA resource allows each independent channel to follow a simple chained command list. Command chains are built up using the DMA command structure, as shown in Figure 38.

Table 334. APBX DMA Channel Assignments

APBX DMA CHANNEL #	USAGE
0	Audio ADCs
1	Audio DACs
2	SPDIF TX
3	I^2C
4	LCD Interface
5	Digital Radio Interface
6	UART RX, IrDA RX
7	UART TX, IrDA TX

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the CPU can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA and have no further concern for the device until the DMA completion interrupt occurs. The STMP36xx is designed to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1 kHz (arrival intervals longer than one ms).

Thus, a single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and controls it uses to write DMA data bytes to the device.

For example, this allows a chain of operations to be issued to the LCD interface to send command bytes, address bytes, and data transfers to the LCD, where the command and address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA.

Each DMA structure can have from 0 to 15 PIO words appended to it. The #PIOWORDs field, if non-zero, instructs the DMA engine to copy these words to the APB beginning at PADDR = 0x0000 and incrementing its PADDR for each cycle. (Note that for APBX DMA Channel 6, which is the UART/IrDA RX channel, the first PIO word in the DMA command is CTRL1. However, for APBX DMA Channel 7, which is the UART/IrDA TX, the first PIO word in a DMA command is CTRL1.)

During these operations, the DMA drives PSEL corresponding to the device associated to the DMA channel. The PSEL-to-DMA channel association is defined at synthesis time in the STMP36xx. Subsequent generations might choose to implement selectable associations for limited cases.

The DMA master only generates normal write transfers to the APBX. It does *not* generate set, clear, or toggle SCT transfers.

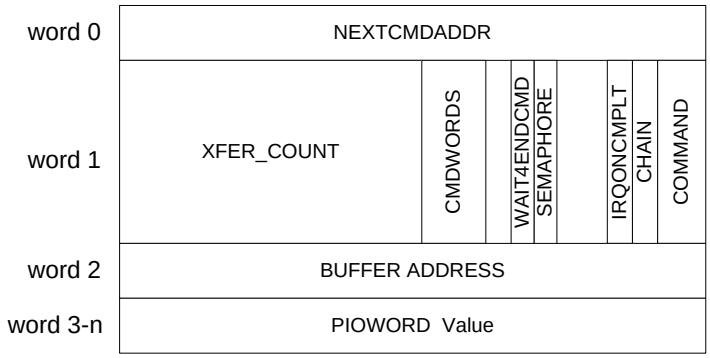


Figure 38. AHB-to-APBX Bridge DMA Channel Command Structure

Once any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. Table 335 shows the four commands implemented by the DMA.

Table 335. APBX DMA Commands

DMA COMMAND	USAGE
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, and then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers, and then perform a DMA transfer to the peripheral for the specified number of bytes.
11	Reserved

DMA_WRITE operations copy data bytes to system memory (on-chip RAM or SDRAM) from the associated peripheral. Each peripheral has a target PADDR value that it expects to receive DMA bytes. This association is synthesized in the DMA. The DMA uses the XML-derived RTL address Include files to make this association, so that the DMA will synthesize to the then current address parametrics extracted from the XML data base. The DMA_WRITE transfer uses the BUFFER_ADDDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from system memory. The DMA engine contains a shared byte aligner that aligns bytes from system

memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel will wait for the device to signal completion of a command by toggling the apx_endcmcd signal before proceeding to load and execute the next command structure. The semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in Table 336, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer count mechanism is duplicated in the associated peripheral, either as an implied or specified count in the peripheral.

Table 336. DMA Channel Command Word in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															
NUMBER DMA BYTES TO TRANSFER																NUMBER PIO WORDS TO WRITE				reserved				WAIT4ENDCMD	DECREMENT SEMAPHORE			IRQ_FINISH	CHAIN	COMMAND	
DMA BUFFER or ALTERNATE CCW																															
ZERO OR MORE PIO WORDS TO WRITE TO THE ASSOCIATED PERIPHERAL STARTING AT ITS BASE ADDRESS ON THE APBX BUS																															

Figure 39 shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to one if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (zero) is loaded into the NEXT_COMMAND_ADDRESS, it will not be detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_FINISH bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by software. It can be used to interrupt the CPU.

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run and process commands and DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the

counting semaphore. If the semaphore goes to zero as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by software. When the semaphore goes to non-zero and the channel is in its IDLE state, then it uses the value in the HW_APBX_CHn_NXTCMDAR (next command address register) to fetch a pointer to the next command to process. NOTE: this is a double indirect case. This method allows software to append to a running command list under the protection of the counting semaphore.

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the HW_APBX_CHn_NXTCMDAR register, and then writes a one to the counting semaphore in HW_APBX_CHn_SEMA. The DMA channel loads HW_APBX_CHn_CURCMDAR register and then enters the normal state machine processing for the next command. When software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of HW_APBXn_CURCMDAR at any time to determine the location of the command structure that is currently being processed.

11.3. DMA Chain Example

The example in Figure 39 shows how to bring the basic items together to make a simple DMA chain to read PCM samples and send them out the Audio Output (DAC) using one DMA channel. This example shows three command structures linked together using their normal command list pointers. The first command writes a single PIO word to the HW_AUDIOOUT_CTRL0 register with a new word count for the DAC. This first command also performs a 512 byte DMA_READ operation to read the data block bytes into the DAC. A second and a third DMA command structure also performs a DMA_READ operation to handle circular buffer style outputs. The completion of each command structure generates an interrupt request. In addition, each command structure decrements the semaphore. If the decompression software has not provided a buffer in a timely fashion, then the DMA will stall. Without the decrement semaphore interlocking, then the DMA will continue to output a stream of samples. In this mode, it is up to software to use the interrupts to synchronize outputs so that underruns do not occur.

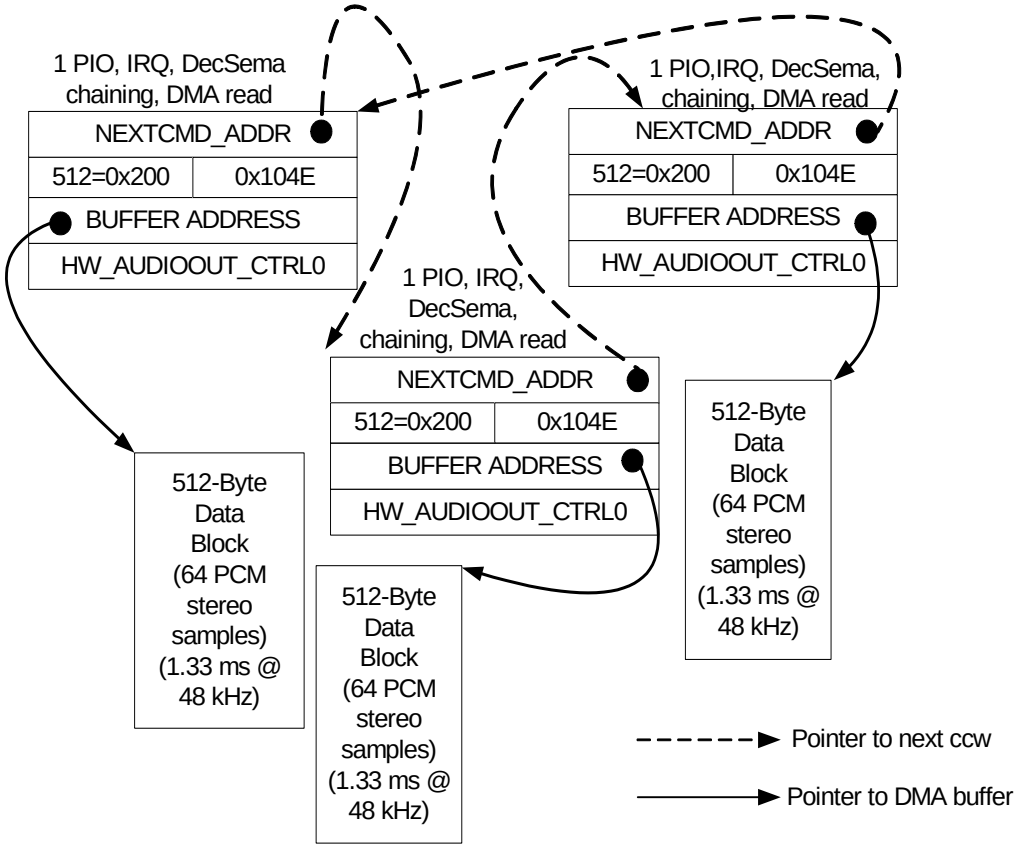


Figure 39. AHB-to-APBX Bridge DMA AUDIOOUT (DAC) Example Command Chain

Note that each word of the three-word DMA Command structure corresponds to a PIO register of the DMA that is accessible on the APBX bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register. In order to start DMA processing, for the first command, one must initialize the PIO registers of the desired channel. FIrst load the next command address register with a pointer to the first command to be loaded. Then write a one to the counting semaphore register. This causes the DMA to schedule the targeted channel for DMA command structure load, as if it just finished its previous command.

11.4. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

11.5. Programmable Registers

This section describes the programmable registers of the AHB-to-APBX bridge block.

11.5.1. AHB-to-APBX Bridge Control and Status Register 0 Description

The AHB-to-APBX Bridge Control and Status Register 0 provides overall control of the AHB-to-APBX bridge and DMA.

HW_APBX_CTRL0 0x80024000
HW_APBX_CTRL0_SET 0x80024004
HW_APBX_CTRL0_CLR 0x80024008
HW_APBX_CTRL0_TOG 0x8002400C

Table 337. HW_APBX_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSVD2						RESET_CHANNEL								RSVD1								FREEZE_CHANNEL							

Table 338. HW_APBX_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set this bit to zero to enable normal APBX DMA operation. Set this bit to one (default) to disable clocking with the APBX DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBX DMA block to its default state.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29:24	**RSVD2**	RO	0x000000	Reserved, always set to zero.
23:16	**RESET_CHANNEL**	RW	0x0	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. Reference the HW_APBX_DEVSEL register to select between the UART and IRDA devices. AUDIOIN = 0x01 AUDIOOUT = 0x02 SPDIF_TX = 0x04 I2C = 0x08 LCDIF = 0x10 DRI = 0x20 UART_RX = 0x30 IRDA_RX = 0x30 UART_TX = 0x40 IRDA_TX = 0x40

Table 338. HW_APBX_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	**RSVD1**	RO	0x000000	Reserved, always set to zero.
7:0	**FREEZE_CHANNEL**	RW	0x0	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is denied access to the central DMA resources. AUDIOIN = 0x01 AUDIOOUT = 0x02 SPDIF_TX = 0x04 I2C = 0x08 LCDIF = 0x10 DRI = 0x20 UART_RX = 0x30 IRDA_RX = 0x30 UART_TX = 0x40 IRDA_TX = 0x40

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

11.5.2. *AHB-to-APBX Bridge Control and Status Register 1 Description*

The AHB-to-APBX Bridge Control and Status Register 1 provides overall control of the interrupts generated by the AHB-to-APBX DMA.

HW_APBX_CTRL1 0x80024010
HW_APBX_CTRL1_SET 0x80024014
HW_APBX_CTRL1_CLR 0x80024018
HW_APBX_CTRL1_TOG 0x8002401C

Table 339. HW_APBX_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN	RSVD1								CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ

Table 340. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x000000	Reserved, always set to zero.
23	**CH7_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 7.

Table 340. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	**CH6_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 6.
21	**CH5_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 5.
20	**CH4_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 4.
19	**CH3_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 3.
18	**CH2_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 2.
17	**CH1_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 1.
16	**CH0_CMDCMPLT_IRQ_EN**	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 0.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**CH7_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6	**CH6_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5	**CH5_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4	**CH4_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3	**CH3_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2	**CH2_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

Table 340. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**CH1_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0	**CH0_CMDCMPLT_IRQ**	RW	0x0	Interrupt request status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

DESCRIPTION:

This register contains the per-channel interrupt status bits and the per-channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE:

```
BF_WR(APBX_CTRL1, CH5_CMDCMPLT_IRQ, 0); // use bitfield write macro
BF_APBX_CTRL1.CH5_CMDCMPLT_IRQ = 0; // or, assign to register struct's bitfield
```

11.5.3. AHB-to-APBX DMA Device Assignment Register Description

This register allows reassignment of the APBX device connected to the DMA channels.

HW_APBX_DEVSEL 0x80024020

Table 341. HW_APBX_DEVSEL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CH7				CH6				CH5				CH4				CH3				CH2				CH1				CH0			

Table 342. HW_APBX_DEVSEL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**CH7**	RW	0x0	These bits allow reassignment of the DMA Channel 7 from the default of the UART transmit device to the IRDA transmit device. USE_UART = 0x0 Use the default assignment of UART transmit DMA channel for DMA Channel 7. USE_IRDA = 0x1 Replace the UART transmit channel with the IRDA transmit channel for DMA Channel 7.
27:24	**CH6**	RW	0x0	These bits allow reassignment of the DMA Channel 6 from the default of the UART receive device to the IRDA receive device. USE_UART = 0x0 Use the default assignment of UART receive DMA channel for DMA Channel 6. USE_IRDA = 0x1 Replace the UART receive channel with the UART receive channel for DMA Channel 6.
23:20	**CH5**	RO	0x0	Reserved.
19:16	**CH4**	RO	0x0	Reserved.
15:12	**CH3**	RO	0x0	Reserved.

Table 342. HW_APBX_DEVSEL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11:8	**CH2**	RO	0x0	Reserved.
7:4	**CH1**	RO	0x0	Reserved.
3:0	**CH0**	RO	0x0	Reserved.

DESCRIPTION:

This register provides a mechanism for assigning DMA channels 6 and 7 to either the UART or IRDA devices.

EXAMPLE:

```
Empty Example.
```

11.5.4. APBX DMA Channel 0 Current Command Address Register Description

The APBX DMA Channel 0 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH0_CURCMDAR 0x80024030

Table 343. HW_APBX_CH0_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 344. HW_APBX_CH0_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 0.

DESCRIPTION:

APBX DMA Channel 0 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
  pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR_RD(0); // read the whole register,
since there is only one field
  pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBX_CHn_CURCMDAR, 0, CMD_ADDR); // or, use multi-
register bitfield read macro
  pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR(0).CMD_ADDR; // or, assign from bit-
field of indexed register's struct
```

11.5.5. APBX DMA Channel 0 Next Command Address Register Description

The APBX DMA Channel 0 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH0_NXTCMDAR 0x80024040

Table 345. HW_APBX_CH0_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 346. HW_APBX_CH0_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for channel 0.

DESCRIPTION:

APBX DMA Channel 0 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
  HW_APBX_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure); // write the entire register,
since there is only one field
   BF_WRn(APBX_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure); // or, use multi-register
bitfield write macro
  HW_APBX_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bit-
field of indexed register's struct
```

11.5.6. APBX DMA Channel 0 Command Register Description

The APBX DMA Channel 0 Command Register specifies the DMA transaction to perform for the current command chain item.

HW_APBX_CH0_CMD 0x80024050

Table 347. HW_APBX_CH0_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 348. HW_APBX_CH0_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the ADC device HW_AUDIOIN_DATA. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the ADC, starting with the base PIO address of the ADC (HW_AUDIOIN_CTRL) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the ABPX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH0_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 0 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
  hw_apbh_chn_cmd_t dma_cmd;
  dma_cmd.XFER_COUNT = 512; // transfer 512 bytes
   dma_cmd.COMMAND = BV_APBX_CHn_CMD_COMMAND__DMA_WRITE; // transfer to system memory from
peripheral device
  dma_cmd.CHAIN = 1; // chain an additional command structure on to the list
  dma_cmd.IRQONCMPLT = 1; // generate an interrupt on completion of this command structure
```

11.5.7. APBX DMA Channel 0 Buffer Address Register Description

The APBX DMA Channel 0 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH0_BAR 0x80024060

Table 349. HW_APBX_CH0_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 350. HW_APBX_CH0_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
  hw_apbh_chn_bar_t dma_data;
  dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

11.5.8. APBX DMA Channel 0 Semaphore Register Description

The APBX DMA Channel 0 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH0_SEMA 0x80024070

Table 351. HW_APBX_CH0_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 352. HW_APBX_CH0_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
BF_WR(APBX_CHn_SEMA, 0, INCREMENT_SEMA, 2); // increment semaphore by two
current_sema = BF_RD(APBX_CHn_SEMA, 0, PHORE); // get instantaneous value
```

11.5.9. AHB-to-APBX DMA Channel 0 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 0 state machine and controls.

HW_APBX_CH0_DEBUG1 0x80024080

Table 353. HW_APBX_CH0_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1																STATEMACHINE			

Table 354. HW_APBX_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 354. HW_APBX_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 0 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 0.

EXAMPLE:

```
Empty example.
```

11.5.10. AHB-to-APBX DMA Channel 0 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 0.

HW_APBX_CH0_DEBUG2 0x80024090

Table 355. HW_APBX_CH0_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 356. HW_APBX_CH0_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 0.

EXAMPLE:

```
Empty example.
```

11.5.11. APBX DMA Channel 1 Current Command Address Register Description

The APBX DMA Channel 1 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH1_CURCMDAR 0x800240A0

Table 357. HW_APBX_CH1_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 358. HW_APBX_CH1_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for channel 1.

DESCRIPTION:

APBX DMA Channel 1 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR_RD(1); // read the whole register,
since there is only one field
```

```
  pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBX_CHn_CURCMDAR, 1, CMD_ADDR); // or, use multi-
register bitfield read macro
  pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBX_CHn_CURCMDAR(1).CMD_ADDR; // or, assign from bit-
field of indexed register's struct
```

11.5.12. APBX DMA Channel 1 Next Command Address Register Description

The APBX DMA Channel 1 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH1_NXTCMDAR 0x800240B0

Table 359. HW_APBX_CH1_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 360. HW_APBX_CH1_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 1.

DESCRIPTION:

APBX DMA Channel 1 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 1 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
  HW_APBX_CHn_NXTCMDAR_WR(1, (reg32_t) pCommandTwoStructure); // write the entire register,
since there is only one field
   BF_WRn(APBX_CHn_NXTCMDAR, 1, (reg32_t) pCommandTwoStructure); // or, use multi-register
bitfield write macro
  HW_APBX_CHn_NXTCMDAR(1).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bit-
field of indexed register's struct
```

11.5.13. APBX DMA Channel 1 Command Register Description

The APBX DMA Channel 1 Command Register specifies the cycle to perform for the current command chain item.

HW_APBX_CH1_CMD 0x800240C0

Table 361. HW_APBX_CH1_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 362. HW_APBX_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the DAC device HW_AUDIOOUT_DATA. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the DAC, starting with the base PIO address of the DAC (HW_AUDIOOUT_CTRL) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH1_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 1 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty Example.
```

11.5.14. APBX DMA Channel 1 Buffer Address Register Description

The APBX DMA Channel 1 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH1_BAR 0x800240D0

Table 363. HW_APBX_CH1_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 364. HW_APBX_CH1_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
hw_apbh_chn_bar_t dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

11.5.15. APBX DMA Channel 1 Semaphore Register Description

The APBX DMA Channel 1 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH1_SEMA 0x800240E0

Table 365. HW_APBX_CH1_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 366. HW_APBX_CH1_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
BF_WR(APBX_CHn_SEMA, 1, INCREMENT_SEMA, 2); // increment semaphore by two
current_sema = BF_RD(APBX_CHn_SEMA, 1, PHORE); // get instantaneous value
```

11.5.16. AHB-to-APBX DMA Channel 1 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 1 state machine and controls.

HW_APBX_CH1_DEBUG1 0x800240F0

Table 367. HW_APBX_CH1_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE			

Table 368. HW_APBX_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 368. HW_APBX_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 1 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 1.

EXAMPLE:

```
Empty example.
```

11.5.17. AHB-to-APBX DMA Channel 1 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

HW_APBX_CH1_DEBUG2 0x80024100

Table 369. HW_APBX_CH1_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 370. HW_APBX_CH1_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 1.

EXAMPLE:

```
Empty example.
```

11.5.18. APBX DMA Channel 2 Current Command Address Register Description

The APBX DMA Channel 2 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH2_CURCMDAR 0x80024110

Table 371. HW_APBX_CH2_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 372. HW_APBX_CH2_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 2.

DESCRIPTION:

APBX DMA Channel 2 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.19. APBX DMA Channel 2 Next Command Address Register Description

The APBX DMA Channel 2 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH2_NXTCMDAR 0x80024120

Table 373. HW_APBX_CH2_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 374. HW_APBX_CH2_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 2.

DESCRIPTION:

APBX DMA Channel 2 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 2 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty Example.
```

11.5.20. APBX DMA Channel 2 Command Register Description

The APBX DMA Channel 2 Command Register specifies the cycle to perform for the current command chain item.

HW_APBX_CH2_CMD 0x80024130

Table 375. HW_APBX_CH2_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 376. HW_APBX_CH2_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SPDIF device HW_SPDIF_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the SPDIF, starting with the base PIO address of the SPDIF (HW_SPDIF_CTRL) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH2_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 2 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.21. APBX DMA Channel 2 Buffer Address Register Description

The APBX DMA Channel 2 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH2_BAR 0x80024140

Table 377. HW_APBX_CH2_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 378. HW_APBX_CH2_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.22. APBX DMA Channel 2 Semaphore Register Description

The APBX DMA Channel 2 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH2_SEMA 0x80024150

Table 379. HW_APBX_CH2_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 380. HW_APBX_CH2_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.23. AHB-to-APBX DMA Channel 2 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 2 state machine and controls.

HW_APBX_CH2_DEBUG1 0x80024160

Table 381. HW_APBX_CH2_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE			

Table 382. HW_APBX_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 382. HW_APBX_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 2 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 2.

EXAMPLE:

```
Empty example.
```

11.5.24. AHB-to-APBX DMA Channel 2 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 2.

HW_APBX_CH2_DEBUG2 0x80024170

Table 383. HW_APBX_CH2_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 384. HW_APBX_CH2_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 2.

EXAMPLE:

```
Empty example.
```

11.5.25. APBX DMA Channel 3 Current Command Address Register Description

The APBX DMA Channel 3 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH3_CURCMDAR 0x80024180

Table 385. HW_APBX_CH3_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 386. HW_APBX_CH3_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 3.

DESCRIPTION:

APBX DMA Channel 3 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.26. APBX DMA Channel 3 Next Command Address Register Description

The APBX DMA Channel 3 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH3_NXTCMDAR 0x80024190

Table 387. HW_APBX_CH3_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 388. HW_APBX_CH3_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 3.

DESCRIPTION:

APBX DMA Channel 3 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 3 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.27. APBX DMA Channel 3 Command Register Description

The APBX DMA Channel 3 Command Register specifies the cycle to perform for the current command chain item.

HW_APBX_CH3_CMD 0x800241A0

Table 389. HW_APBX_CH3_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 390. HW_APBX_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the I2C device HW_I2C_DATA. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the I2C, starting with the base PIO address of the I2C (HW_I2C_CTRL0) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH3_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 3 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.28. APBX DMA Channel 3 Buffer Address Register Description

The APBX DMA Channel 3 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH3_BAR 0x800241B0

Table 391. HW_APBX_CH3_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 392. HW_APBX_CH3_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.29. APBX DMA Channel 3 Semaphore Register Description

The APBX DMA Channel 3 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH3_SEMA 0x800241C0

Table 393. HW_APBX_CH3_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 394. HW_APBX_CH3_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.30. AHB-to-APBX DMA Channel 3 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 3 state machine and controls.

HW_APBX_CH3_DEBUG1 0x800241D0

Table 395. HW_APBX_CH3_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	**BURST**	**KICK**	**END**	**RSVD2**			**NEXTCMDADDRVALID**	**RD_FIFO_EMPTY**	**RD_FIFO_FULL**	**WR_FIFO_EMPTY**	**WR_FIFO_FULL**	**RSVD1**																**STATEMACHINE**			

Table 396. HW_APBX_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 396. HW_APBX_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 3 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 3.

EXAMPLE:

```
Empty example.
```

11.5.31. AHB-to-APBX DMA Channel 3 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 3.

HW_APBX_CH3_DEBUG2 0x800241E0

Table 397. HW_APBX_CH3_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 398. HW_APBX_CH3_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 3.

EXAMPLE:

```
Empty example.
```

11.5.32. APBX DMA Channel 4 Current Command Address Register Description

The APBX DMA Channel 4 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH4_CURCMDAR 0x800241F0

Table 399. HW_APBX_CH4_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 400. HW_APBX_CH4_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 4.

DESCRIPTION:

APBX DMA Channel 4 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.33. APBX DMA Channel 4 Next Command Address Register Description

The APBX DMA Channel 4 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH4_NXTCMDAR 0x80024200

Table 401. HW_APBX_CH4_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 402. HW_APBX_CH4_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 4.

DESCRIPTION:

APBX DMA Channel 4 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 4 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.34. APBX DMA Channel 4 Command Register Description

The APBX DMA Channel 4 Command Register specifies the cycle to perform for the current command chain item.

HW_APBX_CH4_CMD 0x80024210

Table 403. HW_APBX_CH4_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 404. HW_APBX_CH4_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the LCDIF device HW_LCDIF_DATA. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the LCDIF, starting with the base PIO address of the LCDIF (HW_LCDIF_CTRL) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH4_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 4 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.35. APBX DMA Channel 4 Buffer Address Register Description

The APBX DMA Channel 4 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH4_BAR 0x80024220

Table 405. HW_APBX_CH4_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 406. HW_APBX_CH4_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device associate with this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.36. APBX DMA Channel 4 Semaphore Register Description

The APBX DMA Channel 4 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH4_SEMA 0x80024230

Table 407. HW_APBX_CH4_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 408. HW_APBX_CH4_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.37. AHB-to-APBX DMA Channel 4 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 4 state machine and controls.

HW_APBX_CH4_DEBUG1 0x80024240

Table 409. HW_APBX_CH4_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE			

Table 410. HW_APBX_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 410. HW_APBX_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 4 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 4.

EXAMPLE:

```
Empty example.
```

11.5.38. AHB-to-APBX DMA Channel 4 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

HW_APBX_CH4_DEBUG2 0x80024250

Table 411. HW_APBX_CH4_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 412. HW_APBX_CH4_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 4.

EXAMPLE:

```
Empty example.
```

11.5.39. APBX DMA Channel 5 Current Command Address Register Description

The APBX DMA Channel 5 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH5_CURCMDAR 0x80024260

Table 413. HW_APBX_CH5_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 414. HW_APBX_CH5_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 5.

DESCRIPTION:

APBX DMA Channel 5 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.40. APBX DMA Channel 5 Next Command Address Register Description

The APBX DMA Channel 5 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH5_NXTCMDAR 0x80024270

Table 415. HW_APBX_CH5_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 416. HW_APBX_CH5_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 5.

DESCRIPTION:

APBX DMA Channel 5 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 5 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.41. APBX DMA Channel 5 Command Register Description

The APBX DMA Channel 5 Command Register specifies the cycle to perform for the current command chain item.

HW_APBX_CH5_CMD 0x80024280

Table 417. HW_APBX_CH5_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 418. HW_APBX_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the DRI device HW_DRI_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the DRI, starting with the base PIO address of the DRI (HW_DRI_CTRL) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH5_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 5 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.42. APBX DMA Channel 5 Buffer Address Register Description

The APBX DMA Channel 5 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH5_BAR 0x80024290

Table 419. HW_APBX_CH5_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 420. HW_APBX_CH5_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.43. APBX DMA Channel 5 Semaphore Register Description

The APBX DMA Channel 5 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH5_SEMA 0x800242A0

Table 421. HW_APBX_CH5_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 422. HW_APBX_CH5_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.44. AHB-to-APBX DMA Channel 5 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 5 state machine and controls.

HW_APBX_CH5_DEBUG1 0x800242B0

Table 423. HW_APBX_CH5_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE			

Table 424. HW_APBX_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 424. HW_APBX_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 5 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 5.

EXAMPLE:

```
Empty example.
```

11.5.45. AHB-to-APBX DMA Channel 5 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

HW_APBX_CH5_DEBUG2 0x800242C0

Table 425. HW_APBX_CH5_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 426. HW_APBX_CH5_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 5.

EXAMPLE:

```
Empty example.
```

11.5.46. APBX DMA Channel 6 Current Command Address Register Description

The APBX DMA Channel 6 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH6_CURCMDAR 0x800242D0

Table 427. HW_APBX_CH6_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 428. HW_APBX_CH6_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 6.

DESCRIPTION:

APBX DMA Channel 6 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.47. APBX DMA Channel 6 Next Command Address Register Description

The APBX DMA Channel 6 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH6_NXTCMDAR 0x800242E0

Table 429. HW_APBX_CH6_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 430. HW_APBX_CH6_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 6.

DESCRIPTION:

APBX DMA Channel 6 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 6 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.48. APBX DMA Channel 6 Command Register Description

The APBX DMA Channel 6 Command Register specifies the cycle to perform for the current command chain item.

HW_APBX_CH6_CMD 0x800242F0

Table 431. HW_APBX_CH6_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 432. HW_APBX_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH6_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 6 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.49. APBX DMA Channel 6 Buffer Address Register Description

The APBX DMA Channel 6 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH6_BAR 0x80024300

Table 433. HW_APBX_CH6_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 434. HW_APBX_CH6_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.50. APBX DMA Channel 6 Semaphore Register Description

The APBX DMA Channel 6 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH6_SEMA 0x80024310

Table 435. HW_APBX_CH6_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 436. HW_APBX_CH6_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.51. AHB-to-APBX DMA Channel 6 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 6 state machine and controls.

HW_APBX_CH6_DEBUG1 0x80024320

Table 437. HW_APBX_CH6_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1															STATEMACHINE			

Table 438. HW_APBX_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 438. HW_APBX_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 6 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 6.

EXAMPLE:

```
Empty example.
```

11.5.52. AHB-to-APBX DMA Channel 6 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

HW_APBX_CH6_DEBUG2 0x80024330

Table 439. HW_APBX_CH6_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 440. HW_APBX_CH6_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 6.

EXAMPLE:

```
Empty example.
```

11.5.53. APBX DMA Channel 7 Current Command Address Register Description

The APBX DMA Channel 7 Current Command Address Register points to the multi-word command that is currently being executed. Commands are threaded on the command address.

HW_APBX_CH7_CURCMDAR 0x80024340

Table 441. HW_APBX_CH7_CURCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 442. HW_APBX_CH7_CURCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RO	0x00000000	Pointer to command structure currently being processed for Channel 7.

DESCRIPTION:

APBX DMA Channel 7 is controlled by a variable-sized command structure. This register points to the command structure currently being executed.

EXAMPLE:

```
Empty example.
```

11.5.54. APBX DMA Channel 7 Next Command Address Register Description

The APBX DMA Channel 7 Next Command Address Register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

HW_APBX_CH7_NXTCMDAR 0x80024350

Table 443. HW_APBX_CH7_NXTCMDAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ADDR																															

Table 444. HW_APBX_CH7_NXTCMDAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ADDR**	RW	0x00000000	Pointer to next command structure for Channel 7.

DESCRIPTION:

APBX DMA Channel 7 is controlled by a variable-sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 7 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE:

```
Empty example.
```

11.5.55. APBX DMA Channel 7 Command Register Description

The APBX DMA Channel 7 Command Register specifies the cycle to perform for the current command chain item.

HW_APBX_CH7_CMD 0x80024360

Table 445. HW_APBX_CH7_CMD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
XFER_COUNT																CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	

Table 446. HW_APBX_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**XFER_COUNT**	RO	0x0	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART device HW_UARTAPP_DATA register. A value of 0 indicates a 64-Kbyte transfer.
15:12	**CMDWORDS**	RO	0x00	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART (HW_UARTAPP_CTRL0) and incrementing from there. Zero means transfer NO command words
11:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7	**WAIT4ENDCMD**	RO	0x0	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6	**SEMAPHORE**	RO	0x0	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5:4	**RSVD0**	RO	0x0	Reserved, always set to zero.
3	**IRQONCMPLT**	RO	0x0	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e., after the DMA transfer is complete.
2	**CHAIN**	RO	0x0	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH7_CMDAR to find the next command.
1:0	**COMMAND**	RO	0x00	This bitfield indicates the type of current command: 00- No DMA transfer 01- Write transfers, i.e., data sent from the APBX device (APB PIO read) to the system memory (AHB master write). 10- Read transfer 11- Reserved NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer. DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DESCRIPTION:

The APBX DMA Channel 7 Command Register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an

additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE:

```
Empty example.
```

11.5.56. APBX DMA Channel 7 Buffer Address Register Description

The APBX DMA Channel 7 Buffer Address Register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

HW_APBX_CH7_BAR 0x80024370

Table 447. HW_APBX_CH7_BAR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADDRESS																															

Table 448. HW_APBX_CH7_BAR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ADDRESS**	RO	0x00000000	Address of system memory buffer to be read or written over the AHB bus.

DESCRIPTION:

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE:

```
Empty example.
```

11.5.57. APBX DMA Channel 7 Semaphore Register Description

The APBX DMA Channel 7 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

HW_APBX_CH7_SEMA 0x80024380

Table 449. HW_APBX_CH7_SEMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2								PHORE								RSVD1								INCREMENT_SEMA							

Table 450. HW_APBX_CH7_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD2**	RO	0x0	Reserved, always set to zero.
23:16	**PHORE**	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	**RSVD1**	RO	0x0	Reserved, always set to zero.
7:0	**INCREMENT_SEMA**	RW	0x00	The value written to this field is added to the semaphore count in an atomic way, such that simultaneous software adds and DMA hardware subtracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, in which case the count is incremented by a net one.

DESCRIPTION:

Each DMA channel has an 8-bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE:

```
Empty example.
```

11.5.58. AHB-to-APBX DMA Channel 7 Debug Register 1 Description

This register gives debug visibility into the APBX DMA Channel 7 state machine and controls.

HW_APBX_CH7_DEBUG1 0x80024390

Table 451. HW_APBX_CH7_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REQ	**BURST**	**KICK**	**END**	**RSVD2**			**NEXTCMDADDRVALID**	**RD_FIFO_EMPTY**	**RD_FIFO_FULL**	**WR_FIFO_EMPTY**	**WR_FIFO_FULL**	**RSVD1**																**STATEMACHINE**			

Table 452. HW_APBX_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**REQ**	RO	0x0	This bit reflects the current state of the DMA Request signal from the APB device.
30	**BURST**	RO	0x0	This bit reflects the current state of the DMA Burst signal from the APB device.
29	**KICK**	RO	0x0	This bit reflects the current state of the DMA Kick signal sent to the APB device.
28	**END**	RO	0x0	This bit reflects the current state of the DMA End Command signal sent from the APB device.
27:25	**RSVD2**	RO	0x0	Reserved
24	**NEXTCMDADDRVALID**	RO	0x0	This bit reflects the internal bit that indicates whether the channel's next command address is valid.
23	**RD_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Read FIFO Empty signal.
22	**RD_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Read FIFO Full signal.
21	**WR_FIFO_EMPTY**	RO	0x1	This bit reflects the current state of the DMA channel's Write FIFO Empty signal.
20	**WR_FIFO_FULL**	RO	0x0	This bit reflects the current state of the DMA channel's Write FIFO Full signal.

Table 452. HW_APBX_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	**RSVD1**	RO	0x0	Reserved
4:0	**STATEMACHINE**	RO	0x0	PIO Display of the DMA Channel 7 state machine state. IDLE = 0x00 This is the idle state of the DMA state machine. REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command. REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command. REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command. XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly. REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete. REQ_CMD4 = 0x06 State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers. READ_FLUSH = 0x08 During read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. WRITE = 0x0C During DMA write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. READ_REQ = 0x0D During DMA read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. WAIT_END = 0x15 When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. WRITE_WAIT = 0x1C During DMA write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 7.

EXAMPLE:

```
Empty example.
```

11.5.59. AHB-to-APBX DMA Channel 7 Debug Register 2 Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 7.

HW_APBX_CH7_DEBUG2 0x800243A0

Table 453. HW_APBX_CH7_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
APB_BYTES																AHB_BYTES															

Table 454. HW_APBX_CH7_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**APB_BYTES**	RO	0x0	This value reflects the current number of APB bytes remaining to be transferred in the current transfer.
15:0	**AHB_BYTES**	RO	0x0	This value reflects the current number of AHB bytes remaining to be transferred in the current transfer.

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 7.

EXAMPLE:

```
Empty example.
```

APBX XML Revision: 1.31

12. EXTERNAL MEMORY INTERFACE (EMI)

This chapter describes the external memory interface (EMI) on the STMP36xx, including sections on the dynamic memory controller, static memory controller, and an operation example. Programmable registers are described in Section 12.6.

12.1. Overview

The EMI is a configurable interface to external SDRAM and static memories such as NOR flash. The EMI provides memory-mapped access to as many as four external devices utilizing chip selects. This allows a contiguous address space of 1 Gbyte. The EMI has native support for 16-Mbit, 64-Mbit, 128-Mbit, 256-Mbit, and 512-Mbit SDRAMs. NOR flash up to 128 MB are supported. The EMI has a 16-bit external data bus and up to a 26-bit address bus. The upper 11 bits of the external address bus are shared with the General-Purpose Media Interface (GPMI). This arrangement allows simultaneous access to a GPMI peripheral (such as an ATA drive) and an SDRAM. Figure 40 shows a block diagram of the external memory interface.

External memory is connected to the system via an AHB slave. The configuration registers are accessed via an APB slave or the APBH bus. The EMI uses three clocks: HCLK, EMICLK, and EXRAM_XCLK16K. Access time to asynchronous memories is programmable, with setup and hold timed defined by integer numbers of HCLK cycles.

EMI peripherals are memory-mapped into the system's address space and can be directly accessed by the CPU or another AHB bus master. The DMA's memcopy peripheral can automatically copy blocks of data between external memory and on-chip SRAM.

The EMI's static memory controller allows individual slaves to be write-protected (generates Bus Error on write to write-protected slave).

The EMI's DRAM controller has a number of power-saving and performance-increasing features, including the following:

- Programmable refresh timer allows for temperature-compensated refresh.
- Auto-standby mode gates the clock during periods of inactivity.
- Support for self-refresh while the STMP36xx is off.
- AHB spits allow other bus masters to access on-chip memory while the EMI accesses slower off-chip memory. This allows multiple outstanding requests to be pipelined within the controller.
- SDRAM clock speeds up to 66 MHz are supported.
- AHB transactions of 8, 16, or 32 bits are supported.
- The EMI includes 64x32 read/write buffers to improve performance.
- Programmable address modes tune memory utilization to a specific application.

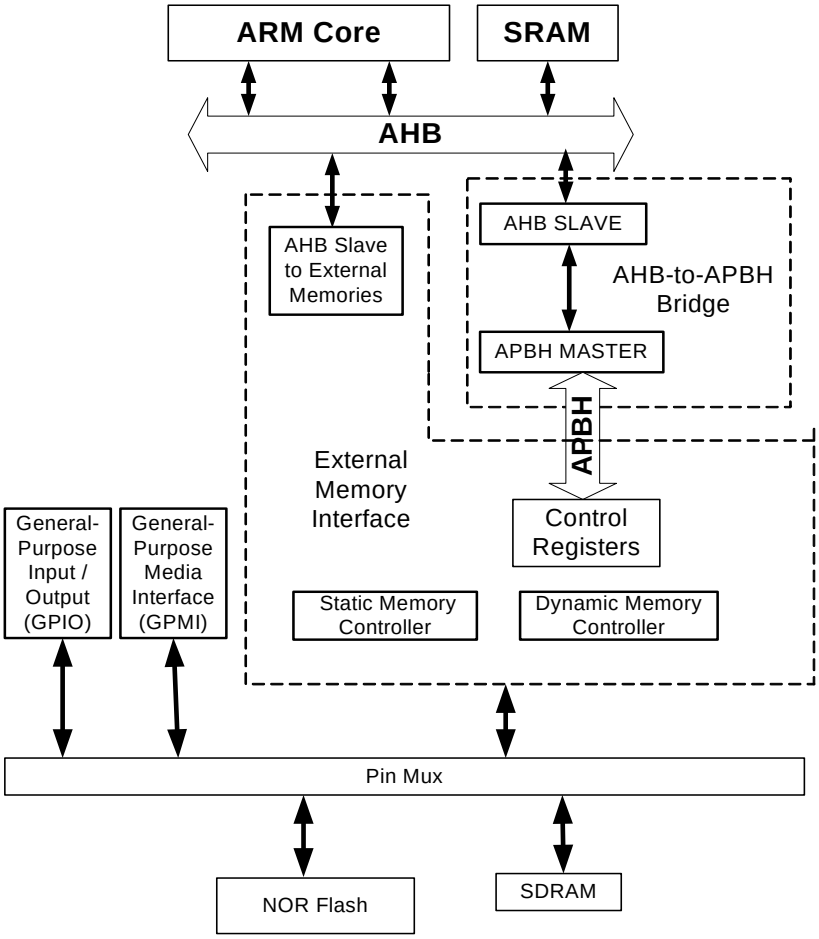


Figure 40. External Memory Interface Block Diagram

12.2. Dynamic Memory Controller

DRAM timing parameters are based on EMICLK cycles. To avoid having to change them when EMICLK changes, it is necessary to set them according to the worst-case EMICLK speed. Parameters for minimum timings are set according to the maximum EMICLK speed. The refresh timer is a special case and uses a stable clock reference rather than EMICLK.

To ensure correct operation, the DRAM is put into self-refresh (or power-down) when EMICLK is lower than the minimum necessary to perform auto-refresh. Extremely low-power modes, such as USB suspend, may not be able to tolerate the DRAM's self-refresh current, which can be as high as 2.5 mA per DRAM chip. In that case, software must put the DRAM into the power-down state in which it is not refreshed and loses its contents.

The DRAM controller can be configured to always auto precharge after each burst or wait until either an out-of-row access has occurred or the maximum row-open time (typically 1 ms) has expired. The row-open timer is based on an internal counter that counts EMICLK cycles.

The DRAM controller always uses an 8-cycle burst (16 bytes). AHB WRAP8 accesses require the DRAM controller to perform two DRAM burst accesses. To support the performance enhancement of the ARM926 Critical Word First cache load, the DRAM controller provides the first requested byte as soon as it can be read.

Commands given to the SDRAM chips are encoded on three interface signals: sdram_ras, sdram_cas, and sdram_we. On any rising edge of the SDRAM chip's clock, it captures these three lines and decodes them. When a read command is encoded, the data is available a fixed number of cycles later. For the STMP36xx, the mode register is always loaded with a CAS latency of two, which tells the SDRAM to return data on the second rising edge after a read command. Many of the timing parameters specified in the SDRAM timing registers set the minimum time between various commands. SDRAM No-Op commands are used to fill the gaps and keep all the timing within specification.

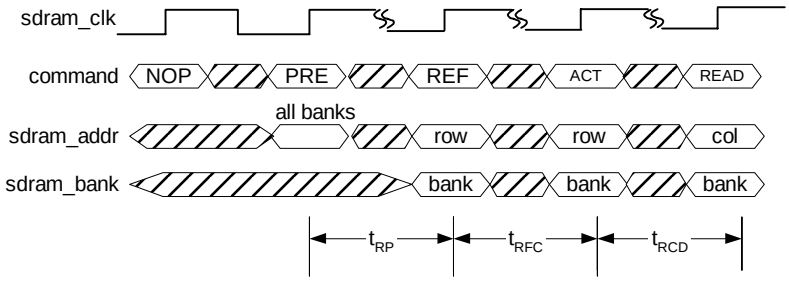


Figure 41. SDRAM Programmable Timing Parameters

12.2.1. DRAM Timing

Most DRAM timing parameters are programmable. However, the following DRAM timing parameter is fixed:

WRITE Recovery Time (**tWR**)—Always 2 EMICLK Cycles. Minimum time between last WRITE data in and Precharge command. Additional cycles may be inserted to meet minimum tRAS requirement.

12.3. Static Memory Controller (SMC)

The EMI static memory controller supports external ROM, NOR flash, and most other asynchronous, parallel data and address devices. Its primary function in the STMP36xx is to support NOR flash.

The static memory controller supports 16-bit external devices. Most AHB accesses in the STMP36xx are bursts of four or eight 32-bit words, so the static memory controller will often perform sixteen sequential accesses on the external device. To increase efficiency, the SMC supports page-mode access.

The SMC has a single pin timing controller, so if more than one type of device is connected to it, then the worst-case timing of all devices should be programmed. Pin timing values are all based on EMICLK cycles. If the EMICLK speed is to change in the application, the timing values must either be set according to the highest speed or adjusted when EMICLK changes.

12.4. EMI Operation Example

Initialization steps:

1. Set the PIN MUXSEL registers to enable EMI control of the following pads. (See Chapter 35, "Pin Descriptions" on page 809.)

- SDRAM pads:
  - emi_a[14:0]
  - emi_data[15:0]
  - emi_casn
  - emi_rasn
  - emi_wen
  - emi_ce0n, emi_ce1n, emi_ce2n, emi_ce3n (as appropriate)
  - emi_dqm0, emi_dqm1
  - emi_clk
  - emi_cke

- NOR flash pad:
  - emi_a[25:0]
  - emi_data[7:0] or emi_data[15:0] based upon whether flash is 8 bit or 16 bit
  - emi_wen
  - emi_oen
  - emi_ce0n, emi_ce1n, emi_ce2n, emi_ce3n (as appropriate)

2. Clear the CLKGATE bit and set the appropriate divisor value in the Clock Control's EMICLKCTRL register. (See Chapter 4, "Clock Generation and Control" on page 47 for more information.)
3. Bring the EMI out of soft reset and clock gate.
4. Set the chip enable fields in CE3_MODE, CE2_MODE, CE1_MODE, and CE0_MODE to the appropriate value. If no device is connected to one of the chip enables, leave the default of zero.

If SDRAM is present, program the following:

1. HW_EMIDRAMCTRL register must be programmed appropriately. Note that EMICLK_ENABLE and EMICLKEN_ENABLE must be high for the SDRAM to function.
2. Program the HW_EMIDRAMADDR with the appropriate row, column, and mode for the SDRAM.
3. Program the timings for the SDRAM from the SDRAM's specification sheet into the HW_EMIDRAMTIME and HW_EMIDRAMTIME2 registers. Calculate the frequency time based on the settings programmed in the CLKCTRL registers. The timing values represent X+1 cycle delay. For example, if the SDRAM requires a minimum 45 ns TRC delay and the EMICLK is 25 MHz (40-ns clock frequency), then round the clock cycle TRC delay to two cycles (80 ns) in order to cover the 45 ns TRC. The value programmed into the TRC field would be 1.
4. Finally, program the HW_EMIDRAMMODE register. This register must be programmed last, because it triggers a LOAD MODE operation to the SDRAM. At this point, the SDRAM is ready for operation.

If NOR flash is present, program the following:

1. Program the HW_EMISTATICCTRL register with the appropriate MEM_WIDTH and WRITE_PROTECT values.
2. Program the timings for the NOR flash from the NOR flash specification sheet into the HW_EMISTATICTIME register. Note that the timing are X+1 clock cycles based off the EMICLK.
3. You may reset the NOR flash by appropriately setting the RESET_OUT field, being careful to observe the proper reset requirements of the part. At this point, the NOR flash should be ready for operation.

How to perform a self-refresh to the SDRAM:

1. Wait until the BUSY field in register HW_EMIDRAMSTAT indicates the SDRAM is not busy.
2. Set SELF_REFRESH to 1. This instructs the EMI to place the SDRAM into self-refresh mode.
3. To come out of self-refresh, set SELF_REFRESH to 0. The user must wait the requisite amount of time according to the part's specification sheet before bringing the SDRAM out of self-refresh.

How to safely change the EMI clock divider:

1. Ensure that no device in the system is accessing the EMI controller (APBH DMA, APBX DMA, or USB).
2. Wait until the HW_EMISTAT_BUSY bit is clear, indicating that there are no outstanding transactions for the EMI to process.
3. Write the new value to the HW_CLKCTRL_EMICLKCTRL_DIV bit field.
4. Resume accesses to the EMI controller.

Note that the code running on the ARM processor to perform this function should be executing out of the on-chip RAM and not from the SDRAM or NOR flash.

12.5. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

12.6. Programmable Registers

This section describes the programmable registers of the external memory interface (EMI).

12.6.1. EMI Control Register Description

HW_EMICTRL	0x80020000
HW_EMICTRL_SET	0x80020004
HW_EMICTRL_CLR	0x80020008
HW_EMICTRL_TOG	0x8002000C

Table 455. HW_EMICTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSVD0																										CE3_MODE	CE2_MODE	CE1_MODE	CE0_MODE

Table 456. HW_EMICTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Reset EMI. 0: EMI is not reset. 1: EMI is reset.
30	**CLKGATE**	RW	0x1	Gates HCLK going into EMI. 0: Clocks are not gated (EMI on). 1: Clocks are gated (EMI off)
29:4	**RSVD0**	RO	0x0	Reserved
3	**CE3_MODE**	RW	0x0	Selects which controller is connected to CE3. This should only be changed during initialization. STATIC = 0x0 CE3 is connected to an external static memory device. DRAM = 0x1 CE3 is connected to an external DRAM device.
2	**CE2_MODE**	RW	0x0	Selects which controller is connected to CE2. This should only be changed during initialization. STATIC = 0x0 CE2 is connected to an external static memory device. DRAM = 0x1 CE2 is connected to an external DRAM device.
1	**CE1_MODE**	RW	0x0	Selects which controller is connected to CE1. This should only be changed during initialization. STATIC = 0x0 CE1 is connected to an external static memory device. DRAM = 0x1 CE1 is connected to an external DRAM device.
0	**CE0_MODE**	RW	0x0	Selects which controller is connected to CE0. This should only be changed during initialization. STATIC = 0x0 CE0 is connected to an external static memory device. DRAM = 0x1 CE0 is connected to an external DRAM device.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.2. EMI Status Register Description

HW_EMISTAT 0x80020010

Table 457. HW_EMISTAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DRAM_PRESENT	STATIC_PRESENT	LARGE_DRAM_ENABLED	RSVD0																											WRITE_BUFFER_DATA	BUSY

Table 458. HW_EMISTAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**DRAM_PRESENT**	RO	0x1	Indicates that the dynamic memory controller (SDRAM/DDR) is present in this product.
30	**STATIC_PRESENT**	RO	0x1	Indicates that the static memory controller is present in this product.
29	**LARGE_DRAM_ENABLED**	RO	0x1	Indicates that the DRAM controller supports large DRAM memories in this product. A large DRAM is defined as being greater than 2 MB in size.
28:2	**RSVD0**	RO	0x0	Reserved
1	**WRITE_BUFFER_DATA**	RO	0x0	Indicates if the EMI write buffer has data waiting to be written. EMPTY = 0x0 The write data buffer is empty. NOT_EMPTY = 0x1 The write buffer is not empty.
0	**BUSY**	RO	0x0	Indicates if the EMI is busy. Activity on either the DRAM or static memory controllers will cause this bit to be set. NOT_BUSY = 0x0 The EMI is not busy. BUSY = 0x1 The EMI is busy.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.3. EMI Debug Register Description

HW_EMIDEBUG 0x80020020

Table 459. HW_EMIDEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1													STATIC_STATE			RSVD0												DRAM_STATE			

Table 460. HW_EMIDEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSVD1**	RO	0x0	Reserved
18:16	**STATIC_STATE**	RO	0x0	Represents the current state of the static memory controller.
15:5	**RSVD0**	RO	0x0	Reserved
4:0	**DRAM_STATE**	RO	0x0	Represents the current state of the DRAM memory controller.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.4. EMI DRAM Status Register Description

HW_EMIDRAMSTAT 0x80020080

Table 461. HW_EMIDRAMSTAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																													SELF_REFRESH_ACK	BUSY	READY

Table 462. HW_EMIDRAMSTAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:3	**RSVD0**	RO	0x0	Reserved
2	**SELF_REFRESH_ACK**	RO	0x0	Indicates that the DRAM has entered self-refresh mode.
1	**BUSY**	RO	0x0	Indicates that the DRAM controller is busy (memory access, command, or auto refresh). This bit must be cleared before configuration changes are made or before the DRAM is placed into self-refresh mode.
0	**READY**	RO	0x0	Indicates that the DRAM controller is ready for a new command (not clock-gated or in self-refresh mode)

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.5. EMI DRAM Control Register Description

HW_EMIDRAMCTRL 0x80020090
HW_EMIDRAMCTRL_SET 0x80020094
HW_EMIDRAMCTRL_CLR 0x80020098
HW_EMIDRAMCTRL_TOG 0x8002009C

Table 463. HW_EMIDRAMCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD3					EMICLK_DIVIDE			AUTO_EMICLK_GATE	RSVD2	EMICLK_ENABLE	EMICLKEN_ENABLE	DRAM_TYPE				RSVD1													PRECHARGE	SELF_REFRESH	RSVD0

Table 464. HW_EMIDRAMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD3**	RO	0x0	Reserved
26:24	**EMICLK_DIVIDE**	RO	0x1	HCLK-to-EMICLK Divide Ratio. See HW_CLKCTRL_EMICLKCTRL_DIV in the Clock Control chapter for specifics on the valid values. Note that HW_CLKCTRL_EMICLKCTRL_DIV cannot be changed during a transfer, or else the data in the DRAM may be corrupted.

Table 464. HW_EMIDRAMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23	**AUTO_EMICLK_GATE**	RW	0x0	When set and the EMI is not busy, EMI_CLK does not run and EMI_CKE is low. When low, the EMI_CLK always runs (if EMICLK_ENABLE is set) and EMI_CKE is always high (if EMICLKEN_ENABLE is set).
22	**RSVD2**	RO	0x0	Reserved.
21	**EMICLK_ENABLE**	RW	0x0	Enables the EMI_CLK output. When AUTO_EMICLK_GATE is not set, this bit forces the clock to be active. When AUTO_EMICLK_GATE is set, the clock will propogate to the SDRAM chip when active cycles are taking place.
20	**EMICLKEN_ENABLE**	RW	0x0	Enabled the EMI_CKE output. When set, EMI_CKE output is high or automatically driven high/low based on the setting of AUTO_EMICLK_GATE. When low, EMI_CKE is always driven low, effectively disabling the external DRAM devices.
19:16	**DRAM_TYPE**	RO	0x0	Controls the type of memory that is connected to the EMI dynamic controller. 0x0: SDRAM, 0x1: LP-SDRAM, 0x2:Mobile-DDR, 0x3: Reserved. At this time, only the SDRAM type is supported.
15:3	**RSVD1**	RO	0x0	Reserved
2	**PRECHARGE**	RW	0x0	Forces the DRAM state machine to close any open row at the end of a burst. Otherwise, a terminate will be performed, leaving the row open.
1	**SELF_REFRESH**	RW	0x0	Places the DRAM into self-refresh mode. This mode must be used if the DRAM contents are to remain valid when EMICLK is too slow, when the EMI is disabled or if the STMP36xx is powered down. DRAM cannot be accessed while in self-refresh mode or a bus error will occur. This bit should only be set after the DRAM controller is not busy, as indicated in the EMI DRAM Status Register.
0	**RSVD0**	RO	0x0	Reserved

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.6. EMI DRAM Address Configuration Register Description

HW_EMIDRAMADDR 0x800200A0
HW_EMIDRAMADDR_SET 0x800200A4
HW_EMIDRAMADDR_CLR 0x800200A8
HW_EMIDRAMADDR_TOG 0x800200AC

Table 465. HW_EMIDRAMADDR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																							MODE	ROW_BITS				COLUMN_BITS			

Table 466. HW_EMIDRAMADDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:9	**RSVD0**	RO	0x0	Reserved
8	**MODE**	RW	0x1	This bit determines the addressing mode. 0: RBC. 1: BRC. RBC = 0x0 Extract the DRAM address from the AHB address using Row/Bank/Column order. BRC = 0x1 Extract the DRAM address from the AHB address using Bank/Row/Column order.
7:4	**ROW_BITS**	RW	0xb	Determines the number of AHB address bits used for the external DRAM Row Address. Valid values range between 8 and 13. The exact position of these bits in the AHB address is determined by the addressing mode (BRC vs. RBC) and the number of column bits.
3:0	**COLUMN_BITS**	RW	0x8	Determines the number of AHB address bits used for the external DRAM Column Address (picked from the lower bits of the AHB). Valid values range between 8 and 12.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.7. EMI DRAM Mode Configuration Register Description

A write to this register forces the DRAM state machine to perform a Load Mode operation. This is required to occur after all other DRAM timing and address registers are programmed. For the STMP36xx, all SDRAM chips are set up with a CAS latency of 2 and a burst length of 2.

HW_EMIDRAMMODE 0x800200B0

Table 467. HW_EMIDRAMMODE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																									CAS_LATENCY			RSVD0			

Table 468. HW_EMIDRAMMODE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:7	**RSVD1**	RO	0x0	Reserved
6:4	**CAS_LATENCY**	RW	0x0	This value sets the internal latency at which the SDRAM read data is captured. The SDRAM chip is always set up with a CAS latency of 2. This register allows the internal latency to be changed in case of timing issues. RESERVED0 = 0x0 Reserved. RESERVED1 = 0x1 Reserved. CAS2 = 0x2 Set the CAS Latency to 2. CAS3 = 0x3 Set the CAS Latency to 3. RESERVED4 = 0x4 Reserved. RESERVED5 = 0x5 Reserved. RESERVED6 = 0x6 Reserved. RESERVED7 = 0x7 Reserved.
3:0	**RSVD0**	RO	0x0	Reserved

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.8. EMI DRAM Timing Control Register 1 Description

The EMI DRAM Timing Control Register 1 sets up timing parameters for DRAM memories. All timing should be set according to the fastest EMICLK and slowest DRAM used in a system.

HW_EMIDRAMTIME 0x800200C0
HW_EMIDRAMTIME_SET 0x800200C4
HW_EMIDRAMTIME_CLR 0x800200C8
HW_EMIDRAMTIME_TOG 0x800200CC

Table 469. HW_EMIDRAMTIME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1				TRFC				TRC				TRAS				TRCD				RSVD0		TRP		TXSR				REFRESH_COUNTER			

Table 470. HW_EMIDRAMTIME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD1**	RO	0x0	Reserved
27:24	**TRFC**	RW	0xF	Auto-Refresh Period. Minimum number of EMICLKs between each auto-refresh cycle.
23:20	**TRC**	RW	0xF	Active-to-Active Command Period. Minimum number of EMICLK cycles between activations of the same DRAM bank.
19:16	**TRAS**	RW	0xF	Active-to-Precharge Command. Minimum number of EMICLK cycles to wait after activating a bank before precharging that bank.
15:12	**TRCD**	RW	0xF	tRCD. Time between active and read/write command in EMICLK cycles.
11:10	**RSVD0**	RO	0x0	Reserved
9:8	**TRP**	RW	0x3	Precharge Command Period (tRP) in EMICLK Cycles. Time to wait for precharge command to complete before issuing another command to the same bank. Should be set according to the highest EMICLK.
7:4	**TXSR**	RW	0xF	Number of EMICLK Cycles to wait before issuing command after exiting self-refresh (0=1 cycle; 0xF=16 cycles)
3:0	**REFRESH_COUNTER**	RW	0xF	Number of refresh cycles per 16-kHz (62.5-us) reference clock tick. (0 = 1 refresh per tick; 0xF = 16 cycles per tick)

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.9. EMI DRAM Timing Control Register 2 Description

The EMI DRAM Timing Control Register 2 sets up timing parameters for DRAM memories. All timing should be set according to the fastest EMICLK and slowest DRAM used in a system.

HW_EMIDRAMTIME2 0x800200D0
HW_EMIDRAMTIME2_SET 0x800200D4
HW_EMIDRAMTIME2_CLR 0x800200D8
HW_EMIDRAMTIME2_TOG 0x800200DC

Table 471. HW_EMIDRAMTIME2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																PRECHARGE_COUNT															

Table 472. HW_EMIDRAMTIME2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSVD1**	RO	0x0	Reserved
15:0	**PRECHARGE_COUNT**	RW	0xffff	Number of EMI Clock cycles after an activate command before we must force a precharge to close a bank. If no other command causes a precharge after the activate, this counter forces the precharge.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.10. EMI Static Memory Control Register Description

HW_EMISTATICCTRL 0x80020100
HW_EMISTATICCTRL_SET 0x80020104
HW_EMISTATICCTRL_CLR 0x80020108
HW_EMISTATICCTRL_TOG 0x8002010C

Table 473. HW_EMISTATICCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																													MEM_WIDTH	WRITE_PROTECT	RESET_OUT

Table 474. HW_EMISTATICCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:3	**RSVD0**	RO	0x0	Reserved
2	**MEM_WIDTH**	RW	0x1	0: 8-bit memory (for NOR only). 1: 16-bit memory (for SDRAM only).
1	**WRITE_PROTECT**	RW	0x0	0: Writes to addresses mapped to static memory are allowed. 1: Write to the addresses mapped to the static memory are not allowed. This prevents bus contention if a ROM is connected to the static memory controller.
0	**RESET_OUT**	RW	0x0	0: Reset ouput is low. 1: Reset output is high

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

12.6.11. EMI Static Memory Timing Control Register Description

HW_EMISTATICTIME 0x80020110
HW_EMISTATICTIME_SET 0x80020114
HW_EMISTATICTIME_CLR 0x80020118
HW_EMISTATICTIME_TOG 0x8002011C

Table 475. HW_EMISTATICTIME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD4				THZ				RSVD2				TDH				RSVD1				TDS				RSVD0				TAS			

Table 476. HW_EMISTATICTIME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSVD4**	RO	0x0	Reserved
27:24	**THZ**	RW	0xF	Number of EMICLK cycles to wait for the EMI to return the data bus to high-Z after write-enable strobe is deasserted. Set this to prevent bus contention when switching from read to write.
23:20	**RSVD2**	RO	0x0	Reserved
19:16	**TDH**	RW	0xF	Data Hold. EMICLK+1 cycles that the data is held after the write strobe is deasserted. Also the time that the read/write strobe is deasserted in a cycle.
15:12	**RSVD1**	RO	0x0	Reserved
11:8	**TDS**	RW	0xF	Data Setup. EMICLK+1 cycles that the data is valid before write strobe is deasserted. Also the time that the read/write strobe is asserted in a cycle. The total cycle time is TDS+TDH.
7:4	**RSVD0**	RO	0x0	Reserved
3:0	**TAS**	RW	0xF	Address Setup. EMICLK+1 cycles between chip select and address assertion to read/write strobe assertion.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

EMI XML Revision: 1.40

13. GENERAL-PURPOSE MEDIA INTERFACE (GPMI)

This chapter describes the general-purpose media interface (GPMI) on the STMP36xx, including sections on both ATA and NAND modes. Programmable registers are described in Section 13.5.

13.1. Overview

The general-purpose media interface (GPMI) controller is a flexible interface to an ATA device or up to four NAND flash. ATA UDMA and PIO modes are supported. The NAND mode has configurable address and command behavior, providing support for future devices not yet specified. The GPMI resides on the APBH.

Registers are clocked on the HCLK domain. The I/O and pin timing are clocked on a dedicated GPMICLK domain. GPMICLK can be set to maximize I/O performance:

(HCLK / 4) <= GPMICLK <= (HCLK * 4)

Figure 42 shows a block diagram of the GPMI controller.

13.2. GPMI ATA Mode

The GPMI supports ATA devices using ATA-PIO mode 4 and UDMA-4. It can also support CompactFlash devices configured for True IDE mode. The GPMI is designed to support a single ATA device, although additional devices can be added using GPIO pins.

13.2.1. Basic ATA Operation

The GPMI supports all basic ATA operations, including:

- **Register/Data Read/Write**—The GPMI can repeatedly access one register address, as is done with the ATA data register. Or, it can increment the ATA address to read/write a configuration to several ATA registers. The GPMI uses a transfer counter to know how many bytes to read/write to the ATA device. This allows the GPMI to properly use its receive FIFO full during reads.
- **Wait for ATA IRQ**—Many ATA commands require a significant amount of time to complete. When the device is complete, it can signal the GPMI using the IRQ function. The GPMI has a timeout counter that asserts an error IRQ to the CPU if it expires before the ATA device has asserted the IRQ signal.
- **Check Status**—The Read and Compare mode can be used to compare an ATA status word against a reference. If unexpected status is returned, then the GPMI indicates an error to the DMA controller. The DMA can then branch to an alternate command descriptor, which either fixes the problem or interrupts the CPU.
- **Data Transfer**—Data can be transferred to/from an ATA or ATAPI device using PIO or UDMA mode. In most cases, UDMA mode is preferred.

13.2.2. GPMI ATA Clocking and Timing

The GPMI has programmable timing for the important parameters. All timing is based on GPMICLK, which is a dedicated clock. GPMICLK is derived from the PLL and has its own integer divider. Therefore, GPMICLK is dependant on PLL frequency changes, but not on other dividers. Complete information about GPMICLK generation can be found in Chapter 4, "Clock Generation and Control" on page 47.

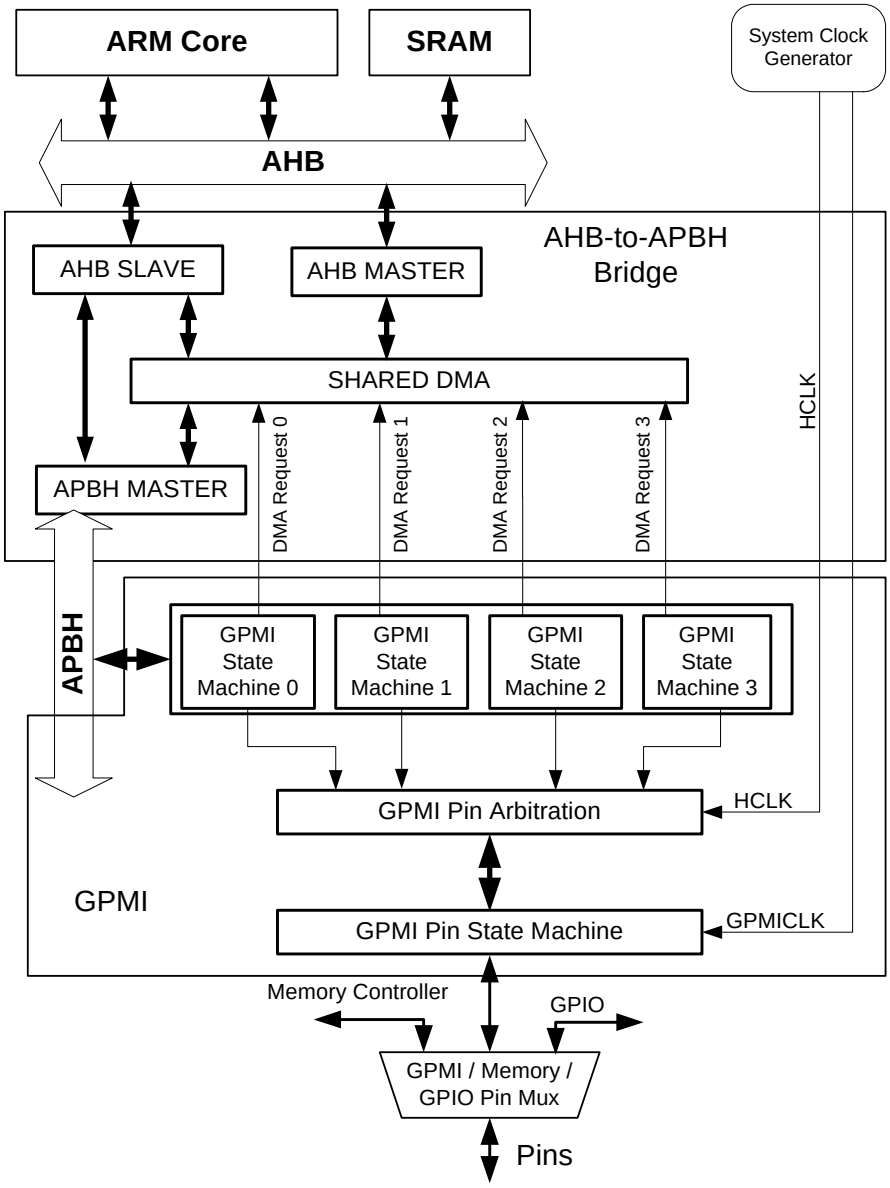


Figure 42. General-Purpose Media Interface Controller Block Diagram

13.2.3. GPMI ATA Pin Sharing

The STMP36xx pinout allows for simultaneous ATA and SDRAM activity. An ATA device can also be used in the same system as a NAND or NOR flash, but the devices cannot operate simultaneously, so software arbitration of the pins is necessary. The GPMI has the basic set of pins needed to support an ATA device:

- **CS0, CS1 Chip Selects**—Shared with the GPMI NAND controller and the EMI. If ATA is used, then these chip selects must not be used for another device.
- **ADDR[2:0] Address Lines**—Shared with the GPMI NAND controller's ALE and CLE lines.

- **DATA[15:0] Data Bus**—Shared with the GPMI NAND controller. The upper data bits (15:8) are also shared with the EMI's NOR flash interface.
- **IRQ Interrupt**—Shared with the NAND1 Ready/Busy.
- **IORDY-:DDMARDY-:DSTROBE I/O Ready, Device DMA Ready, and Device Strobe**—Shared with NAND0 Ready/Busy.
- **DMACK DMA Acknowledge—**Shared with NAND2 Ready/Busy. This pin can be configured as an input for NAND mode and output for ATA UDMA.
- **DIOR-:HDMARDY-:HSTROBE. PIO Read Strobe, Host DMA Ready and Host DMA Strobe**—Shared with the NAND Read Strobe.
- **DIOW-:STOP PIO Write Strobe and Host DMA Stop**—Shared with the NAND Write Strobe.
- **DMARQ Device DMA Request**—Shared with NAND Read/Busy 3. This pin is an input but the NAND and ATA can share the line since they both have three-state (ATA) or open-drain (NAND) drivers.

It is possible for an STMP36xx-based system to support an ATA and one or two NAND flash. However, the ATA and NAND cannot perform simultaneous transactions.

13.2.4. ATA PIO Mode Timing

Figure 43 illustrates the basic GPMI timing parameters in ATA mode.

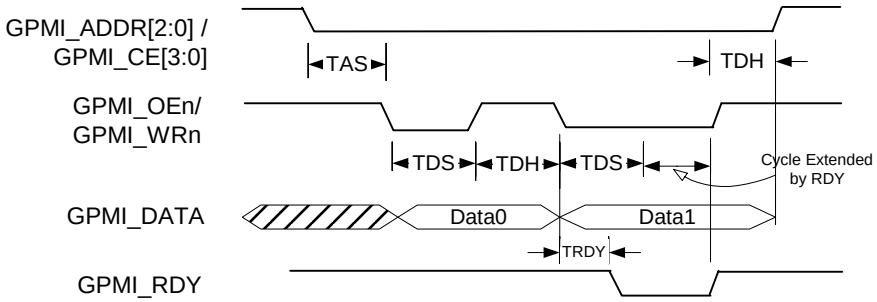


Figure 43. ATA PIO Timing Mode

13.2.5. ATA UDMA Mode

The GPMI supports ATA UDMA up to mode 4, which allows bus transfers of up to 66 MB/s. UDMA also allows a much simpler DMA descriptor chain than PIO operation. ATA-PIO requires that the host receive an interrupt and check status for each block (typically 512 bytes). UDMA allows multiple blocks, up to 64 Kbytes total, to be transferred with a simple set of descriptors.

UDMA uses a significantly different pin architecture than PIO. PIO mode timing uses asynchronous read/write strobes that are controlled by the host. The device can slow down the data transfer within a block using the IORDY signal. The device uses the INTRQ signal to indicate when it is ready to send/receive data between blocks.

UDMA uses data strobes that are asserted by the side actively writing the data (HSTROBE for the host, DSTROBE for the device). The data is transferred on both the rising and falling edges of the strobes.

The device has a DMARQ (DMA request) signal to indicate that it is ready to transfer data to/from the host.

The host has a DMACK (DMA acknowledge) signal to respond to the device DMARQ that it is also ready to transfer data.

The host and device each have DMA ready (HDMARDY and DDMARDY) signals used for cycle-to-cycle flow control during transfers. For example, if the host is reading a block and its FIFO becomes full, it negates HDMARDY, and the device stops toggling DSTROBE and sending new data until the host reasserts HDMARDY.

The host can abort a data transfer by asserting the STOP signal.

13.2.6. UDMA Timings

Figure 44 illustrates the UDMA data write timing. Valid data is presented at TDS before the rising or falling edge of each strobe and held for TDH. The cycle time per word is therefore TDS+TDH, and the time for a complete strobe cycle (rising and falling edges) is 2(TDS+TDH).

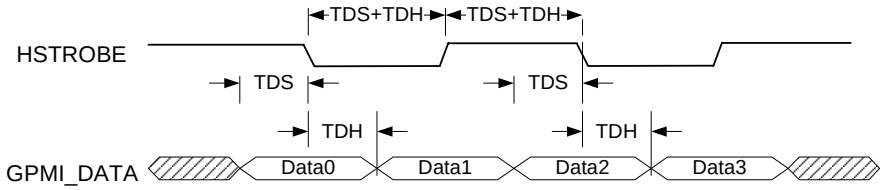


Figure 44. UDMA Timing

13.2.7. ATA Command/IRQ/Check Status Example

Figure 45 illustrates a complex ATA operation including an ATA Read command, Wait for IRQ and Check Status.

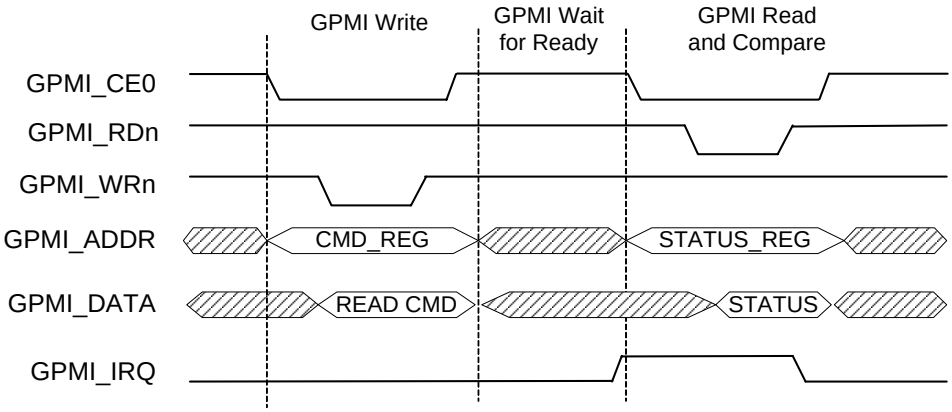


Figure 45. ATA Command/IRQ/Check Status Example

13.3. GPMI NAND Mode

The general-purpose media interface has several features to efficiently support NAND:

- Individual chip select and ready/busy pins for four NANDs (two in 100-pin package).
- Individual state machine and DMA channel for each chip select.
- Special command modes work with DMA controller to perform all normal NAND functions without CPU intervention.
- Configurable timing based on a dedicated clock allows optimal balance of high NAND performance and low system power.

Since current NAND flash does not support multiple page read/write commands, the GPMI and DMA have been designed to handle complex multi-page operations without CPU intervention. The DMA uses a linked descriptor function with branching capability to automatically handle all of the operations needed to read/write multiple pages:

- **Data/Register Read/Write**—The GPMI can be programmed to read or write multiple cycles to the NAND address, command or data registers.
- **Wait for NAND Ready**—The GPMI's Wait-for-Ready mode can monitor the ready/busy signal of a single NAND flash and signal the DMA when the device has become ready. It also has a timeout counter and can indicate to the DMA that a timeout error has occurred. The DMAs can conditionally branch to a different descriptor in the case of an error.
- **Check Status**—The Read-and-Compare mode allows the GPMI to check NAND status against a reference. If an error is found, the GPMI can instruct the DMA to branch to an alternate descriptor, which attempts to fix the problem or asserts a CPU IRQ.

13.3.1. Multiple NAND Support

The GPMI supports up to four NAND chip selects, each with independent ready/busy signals. Since they share a data bus and control lines, the GPMI can only actively communicate with a single NAND at a time. However, all NANDs can concurrently perform internal read, write, or erase operations. With fast NAND flash and software support for concurrent NAND operations, this architecture allows the total throughput to approach the data bus speed, which can be as high as 66 MB/s (16-bit bus running at 33 MHz).

13.3.2. GPMI NAND Timing and Clocking

The dedicated clock, GPMICLK, is used as a timing reference for NAND flash I/O. Since various NANDs have different timing requirements, GPMICLK may need to be adjusted for each application. While the actual pin timings are limited by the NAND chips used, the GPMI can support data bus speeds of up to 33 MHz x 16 bits. The actual read/write strobe timing parameters are adjusted as indicated in the register descriptions in Section 13.5. Refer to Chapter 4 for more information about setting GPMICLK.

13.3.3. Basic NAND Timing

Figure 46 illustrates the operation of the timing parameters in NAND mode.

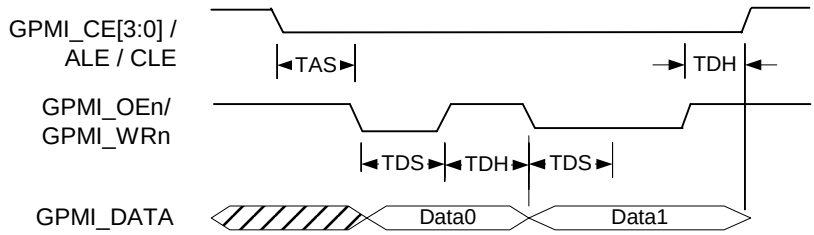


Figure 46. BASIC NAND Timing

13.3.4. NAND Command and Address Timing Example

Figure 47 illustrates a command and address being sent to a NAND flash.

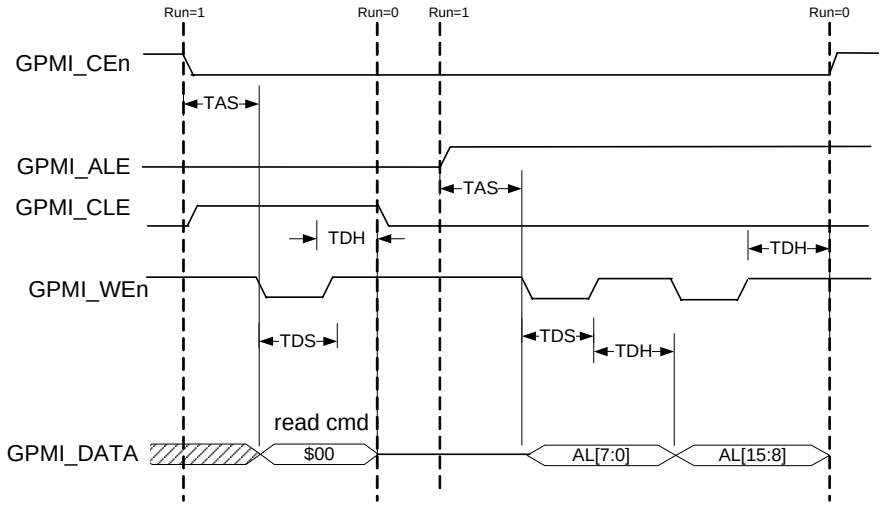


Figure 47. NAND Command and Address Timing Example

13.3.5. NAND Read Timing

The DSAMPLE_TIME is a programmable field in HW_GPMI_CTRL1 register that controls when read data from a NAND device is sampled in the GPMI module. This section describes how to understand the timing issues involved in order to correctly program it.

In the NAND read path timing shown in Figure 48, tSAMPLE represents the time from sample point DS0 (the rising edge of RDN @ Host) to the middle of the window of valid data. By knowing tSAMPLE and the GPMICLK period, the correct sample point can be selected.

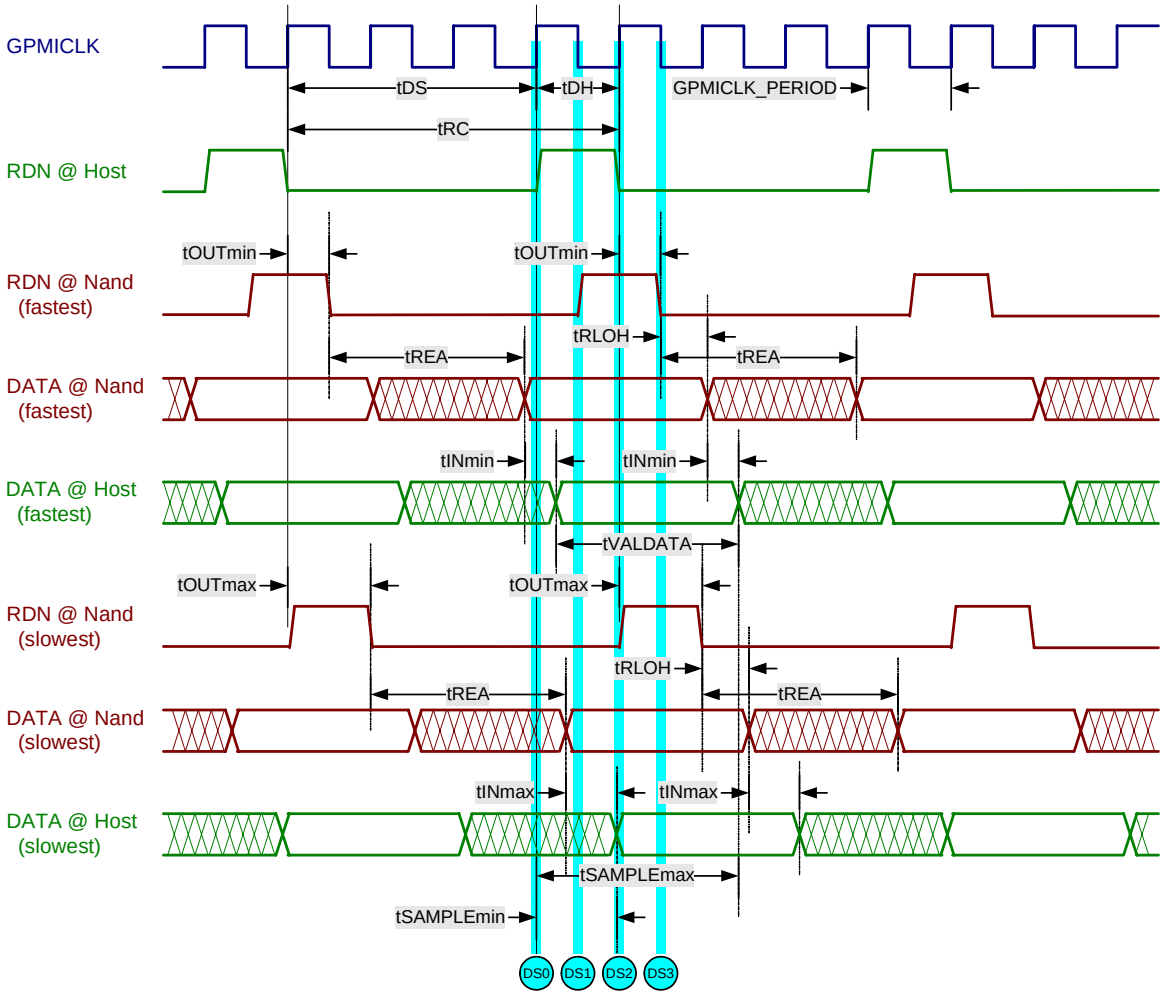


Figure 48. GPMI NAND Read Path Timing

In Figure 48, tVALDATA is the length of the valid read data window. This and the other values shown can be calculated from the following:

```
tVALDATA = tDS + tRHOH - tREA  (Apply when tRC < 30 ns)
tVALDATA = tRC + tRLOH – tREA (Apply when tRC > 30 ns)

Read cycle time is tRC = tDS + tDH.

tSAMPLEmax = tOUTmin + (tREA – tDS) + tINmin + tVALDATA
tSAMPLEmin = tOUTmax + (tREA – tDS) + tINmax
tSAMPLE = (tSAMPLEmin + tSAMPLEmax) / 2

DSAMPLE_TIME = tSAMPLE / (GPMICLK_PERIOD / 2). Round to the nearest integer.
```

Note that (tREA – tDS) could be negative, and therefore tSAMPLE could also be negative. If so, then DSAMPLE_TIME will be zero. Also, parameters must be chosen such that tSAMPLEmax is greater than tSAMPLEmin. Further, ensure that the following condition is met after rounding:

```
tSAMPLEmin < (DSAMPLE_TIME * GPMICLK_PERIOD/2) < tSAMPLEmax
```

Note that Figure 48 has been drawn approximately to scale to represent the minimum and maximum timings for a system using a 120-MHz GPMICLK (i.e., GPMICLK_PERIOD = 8.33 ns) with a total load between the board and the NAND of 30 pF.

The tREA for the NAND in Figure 48 was 20 ns, and the tRLOH was 5 ns. With this kind of load, the STMP36xx timing was a worst case of tOUTmax of 8.5 ns and tINmax of 5 ns and a best case of tOUTmin of 4 ns and iINmin of 3 ns.

With such fast timings as this (33.3-ns read cycles), the only acceptable DSAMPLE_TIME setting is 3. Appropriate timing values for other systems can be calculated using the timing data in Table 477.

Table 477. tOUT: PAD_GPMI_RDN Output Delay (ns)

Load (pf)	4 ma (Min)	4 ma (Max)	8 ma (Min)	8 ma (Max)
10	4.22	8.53	3.61	7.31
20	5.14	10.16	4.00	8.04
30	6.06	11.85	4.38	8.76
40	6.98	13.53	4.75	9.46
50	7.90	15.21	5.13	10.17
60	8.80	16.86	5.50	10.87
70	9.67	18.48	5.87	11.58
80	10.49	20.03	6.24	12.28
90	11.21	21.45	6.62	12.99
100	11.93	22.88	6.99	13.69

13.4. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

13.5. Programmable Registers

The following registers provide control for programmable elements of the GPMI module.

13.5.1. GPMI Control Register 0 Description

The GPMI Control Register 0 specifies the GPMI transaction to perform for the current command chain item.

```
HW_GPMI_CTRL0      0x8000C000
HW_GPMI_CTRL0_SET 0x8000C004
HW_GPMI_CTRL0_CLR 0x8000C008
HW_GPMI_CTRL0_TOG 0x8000C00C
```

Table 478. HW_GPMI_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RUN	DEV_IRQ_EN	TIMEOUT_IRQ_EN	UDMA	COMMAND_MODE		WORD_LENGTH	LOCK_CS	CS		ADDRESS			ADDRESS_INCREMENT	XFER_COUNT															

Table 479. HW_GPMI_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. RUN = 0x0 Allow GPMI to operate normally. RESET = 0x1 Hold GPMI in reset.
30	**CLKGATE**	RW	0x1	Set this bit zero for normal operation. Setting this bit to one (default) gates all of the block level clocks off to miniminize AC energy consumption. RUN = 0x0 Allow GPMI to operate normally. NO_CLKS = 0x1 Do not clock GPMI gates in order to minimize power consumption.
29	**RUN**	RW	0x0	The GPMI is busy running a command whenever this bit is set to '1'. The GPMI is idle whenever this bit set to zero. This can be set to one by a CPU write. In addition, the DMA sets this bit each time a DMA command has finished its PIO transfer phase. IDLE = 0x0 The GPMI is idle. BUSY = 0x1 The GPMI is busy running a command.
28	**DEV_IRQ_EN**	RW	0x0	When set to 1 and ATA_IRQ pin is asserted, the GPMI_IRQ output will assert.
27	**TIMEOUT_IRQ_EN**	RW	0x0	Setting this bit to 1 will enable timeout IRQ for transfers in ATA mode only, and for WAIT_FOR_READY commands in both ATA and NAND mode. The Device_Busy_Timeout value is used for this timeout.
26	**UDMA**	RW	0x0	0= Use ATA-PIO mode on the external bus. 1= Use ATA-Ultra DMA mode on the external bus. DISABLED = 0x0 Use ATA-PIO mode on the external bus. ENABLED = 0x1 Use ATA-Ultra DMA mode on the external bus.

Table 479. HW_GPMI_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:24	**COMMAND_MODE**	RW	0x0	00= Write mode. 01= Read Mode. 10= Read and Compare Mode (setting sense flop). 11= Wait for Ready. WRITE = 0x0 Write mode. READ = 0x1 Read mode. READ_AND_COMPARE = 0x2 Read and compare mode (setting sense flop). WAIT_FOR_READY = 0x3 Wait for ready mode. For ATA WAIT_FOR_READY command set CS=01.
23	**WORD_LENGTH**	RW	0x0	0= 16-bit data bus mode. 1= 8-bit data bus mode. This bit should only be changed when RUN==0. 16_BIT = 0x0 16-bit data bus mode. 8_BIT = 0x1 8-bit data bus mode.
22	**LOCK_CS**	RW	0x0	For ATA/NAND mode: 0= Deassert chip select (CS) after RUN is complete. 1= Continue to assert chip select (CS) after RUN is complete. For Camera Mode: 0= Do not wait for VSYNC rising edge before capturing data. 1= Wait for VSYNC rising edge before capturing data (camera mode only). DISABLED = 0x0 Deassert chip select (CS) after RUN is complete. ENABLED = 0x1 Continue to assert chip select (CS) after RUN is complete.
21:20	**CS**	RW	0x0	Selects which chip select is active for this command. For ATA WAIT_FOR_READY command, this must be set to b01.
19:17	**ADDRESS**	RW	0x0	Specifies the three address lines for ATA mode. In NAND mode, use A0 for CLE and A1 for ALE. NAND_DATA = 0x0 In NAND mode, this address is used to read and write data bytes. NAND_CLE = 0x1 In NAND mode, this address is used to write command bytes. NAND_ALE = 0x2 In NAND mode, this address is used to write address bytes.
16	**ADDRESS_INCREMENT**	RW	0x0	0= Address does not increment. 1= Increment address. In ATA mode, the address will increment with each cycle. In NAND mode, the address will increment once, after the first cycle (going from CLE to ALE). DISABLED = 0x0 Address does not increment. ENABLED = 0x1 Increment address.
15:0	**XFER_COUNT**	RW	0x0	Number of words (8- or 16-bit wide) to transfer for this command.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.2. GPMI Compare Register Description

The GPMI Compare Register specifies the expected data and the XOR mask for comparing to the status values read from the device. This register is used by the Read and Compare command.

HW_GPMI_COMPARE 0x8000C010

Table 480. HW_GPMI_COMPARE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MASK																REFERENCE															

Table 481. HW_GPMI_COMPARE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**MASK**	RW	0x0000	16-bit mask that is applied after the read data is XORed with the REFERENCE bit field.
15:0	**REFERENCE**	RW	0x0000	16-bit value that is XORed with data read from the NAND device.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.3. GPMI Control Register 1 Description

The GPMI Control Register 1 specifies additional control fields that are not used on a per-transaction basis.

HW_GPMI_CTRL1 0x8000C020
HW_GPMI_CTRL1_SET 0x8000C024
HW_GPMI_CTRL1_CLR 0x8000C028
HW_GPMI_CTRL1_TOG 0x8000C02C

Table 482. HW_GPMI_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2																		DSAMPLE_TIME		RSVD1	DEV_IRQ	TIMEOUT_IRQ	BURST_EN	ABORT_WAIT_FOR_READY3	ABORT_WAIT_FOR_READY2	ABORT_WAIT_FOR_READY1	ABORT_WAIT_FOR_READY0	DEV_RESET	ATA_IRQRDY_POLARITY	CAMERA_MODE	GPMI_MODE

Table 483. HW_GPMI_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:14	**RSVD2**	RO	0x0	Always write zeroes to this bit field.
13:12	**DSAMPLE_TIME**	RW	0x0	This variable allows you to select the time when device read data is sampled. 0= At the rising edge of GPMI_CLK when the read strobe deasserts. 1= One-half GPMI_CLK cycle later than when DSAMPLE_TIME == 0. 2= One full GPMI_CLK cycle later than when DSAMPLE_TIME == 0. 3= One and one-half GPMI_CLK cycles later than when DSAMPLE_TIME == 0.
11	**RSVD1**	RO	0x0	Always write zeroes to this bit field.
10	**DEV_IRQ**	RW	0x0	This bit is set when an interrupt is received from the ATA device. Write 0 to clear.
9	**TIMEOUT_IRQ**	RW	0x0	This bit is set when a timeout occurs using the Device_Busy_Timeout value. Write 0 to clear.
8	**BURST_EN**	RW	0x0	When set to 1, each DMA request will generate a 4-transfer burst on the APB bus.
7	**ABORT_WAIT_FOR_READY3**	RW	0x0	Abort a wait-for-ready command on channel 3.
6	**ABORT_WAIT_FOR_READY2**	RW	0x0	Abort a wait-for-ready command on channel 2.
5	**ABORT_WAIT_FOR_READY1**	RW	0x0	Abort a wait-for-ready command on NAND channel 1 or ATA channel 0.
4	**ABORT_WAIT_FOR_READY0**	RW	0x0	Abort a wait-for-ready command on channel 0.
3	**DEV_RESET**	RW	0x0	0= Device Reset pin is held low (asserted). 1= Device Reset pin is held high (deasserted). ENABLED = 0x0 Device Reset pin is held low (asserted). DISABLED = 0x1 Device Reset pin is held high (deasserted).

Table 483. HW_GPMI_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**ATA_IRQRDY_POLARITY**	RW	0x1	For ATA MODE: 0= External ATA IORDY and IRQ are active low. 1= External ATA IORDY and IRQ are active high. For NAND MODE: 0= External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when low and busy when high. 1= External RDY_BUSY[1] and RDY_BUSY[0] pins are ready when high and busy when low. Note NAND_RDY_BUSY[3:2] are not affected by this bit. ACTIVELOW = 0x0 ATA IORDY and IRQ are active low, or NAND_RDY_BUSY[1:0] are active low ready. ACTIVEHIGH = 0x1 ATA IORDY and IRQ are active high, or NAND_RDY_BUSY[1:0] are active high ready.
1	**CAMERA_MODE**	RW	0x0	When set to 1 and ATA UDMA is enabled the UDMA interface becomes a camera interface.
0	**GPMI_MODE**	RW	0x0	0= NAND mode. 1= ATA mode. ATA mode is only supported on channel zero. If ATA mode is selected, then only channel three is available for NAND use. NAND = 0x0 NAND mode. ATA = 0x1 ATA mode.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.4. GPMI Timing Register 0 Description

The GPMI Timing Register 0 specifies the timing parameters that are used by the cycle state machine to guarantee the various setup, hold, and cycle times for the external media type.

HW_GPMI_TIMING0 0x8000C030

Table 484. HW_GPMI_TIMING0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1								ADDRESS_SETUP								DATA_HOLD								DATA_SETUP							

Table 485. HW_GPMI_TIMING0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD1**	RO	0x0	Always write zeroes to this bit field.
23:16	**ADDRESS_SETUP**	RW	0x01	Number of GPMICLK cycles that the CE/ADDR signals are active before a strobe is asserted. A value of zero is interpreted as 0. For ATA PIO modes, this is known in the ATA7 specification as "Address valid to DIOR-/DIOW- setup"
15:8	**DATA_HOLD**	RW	0x02	Data bus hold time in GPMICLK cycles. Also the time that the data strobe is deasserted in a cycle. A value of 0 is interpreted as 64K cycles. For ATA PIO modes this is known in the ATA7 specification as "DIOR-/DIOW- recovery time"
7:0	**DATA_SETUP**	RW	0x03	Data bus setup time in GPMICLK cycles. Also the time that the data strobe is asserted in a cycle. This value must be greater than 2 for ATA devices that use IORDY to extend transfer cycles. A value of 0 is interpreted as 64K cycles. For ATA PIO modes, this is known in the ATA7 specification as "DIOR-/DIOW-"

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.5. GPMI Timing Register 1 Description

The GPMI Timing Register 1 specifies the timeouts used when monitoring the NAND READY pin or the ATA IRQ and IOWAIT signals.

HW_GPMI_TIMING1 0x8000C040

Table 486. HW_GPMI_TIMING1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DEVICE_BUSY_TIMEOUT																ATA_READY_TIMEOUT															

Table 487. HW_GPMI_TIMING1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**DEVICE_BUSY_TIMEOUT**	RW	0x0000	Timeout waiting for NAND Ready/Busy or ATA IRQ. Used in WAIT_FOR_READY mode. This value is the number of GPMI_CLK cycles multiplied by 4096.
15:0	**ATA_READY_TIMEOUT**	RW	0x0000	(DEPRECATED - use DEVICE_BUSY_TIMEOUT instead) Timeout waiting for IORDY to be asserted high. ATAMode only. This value is the number of GPMI_CLK cycles.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.6. GPMI Timing Register 2 Description

The GPMI Timing Register 2 specifies the UDMA timing parameters that are used by the cycle state machine to guarantee the various setup, hold, and cycle times for the external media type.

HW_GPMI_TIMING2 0x8000C050

Table 488. HW_GPMI_TIMING2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UDMA_TRP								UDMA_ENV								UDMA_HOLD								UDMA_SETUP							

Table 489. HW_GPMI_TIMING2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**UDMA_TRP**	RW	0x09	UDMA Ready-to-pause timing. This bit field is also used to specify Tss. (Refer to the UDMA timing specification.) Set UDMA_TRP to the larger of (Trp, Tss). A value of zero is interpreted as 64K cycles.
23:16	**UDMA_ENV**	RW	0x02	UDMA Envelope time. This bit field is also used to specify Tmli and Tzah. (Refer to the UDMA timing specification.) Set UDMA_ENV = max (Tenv, Tmli, Tzah). A value of zero is interpreted as 64K cycles.
15:8	**UDMA_HOLD**	RW	0x01	UDMA Data bus hold time in GPMICLK cycles. A value of zero is interpreted as 64K cycles.
7:0	**UDMA_SETUP**	RW	0x01	UDMA Data bus setup time in GPMICLK cycles. A value of zero is interpreted as 64K cycles.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.7. GPMI DMA Data Transfer Register Description

The GPMI DMA Data Transfer Register is used by the DMA to read or write data to or from the ATA/NAND control state machine.

HW_GPMI_DATA 0x8000C060

Table 490. HW_GPMI_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 491. HW_GPMI_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x00000	In 16-bit mode, this register can be accessed in two 16-bit operations, one bus cycle per operation. In 8-bit mode, one, two, three, or four bytes can can be accessed to send the same number of bus cycles. Byte writes are supported only for the least significant byte. Half-word (16-bit) writes are supported only for the lower half-word.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.8. GPMI Status Register Description

The GPMI Status Register provides a read-back path for various operational states of the GPMI controller.

HW_GPMI_STAT 0x8000C070

Table 492. HW_GPMI_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PRESENT	RSVD1																			RDY_TIMEOUT				ATA_IRQ	RSVD2	FIFO_EMPTY	FIFO_FULL	DEV3_ERROR	DEV2_ERROR	DEV1_ERROR	DEV0_ERROR

Table 493. HW_GPMI_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**PRESENT**	RO	0x1	0= GPMI is not present in this product. 1= GPMI is present is in this product. UNAVAILABLE = 0x0 GPMI is not present in this product. AVAILABLE = 0x1 GPMI is present in this product.
30:12	**RSVD1**	RO	0x0	Always write zeroes to this bit field.
11:8	**RDY_TIMEOUT**	RO	0x0	Status of the RDY/BUSY timeout flags.
7	**ATA_IRQ**	RO	0x0	Status of the ATA_IRQ input pin.
6	**RSVD2**	RO	0x0	Always write zeroes to this bit field.
5	**FIFO_EMPTY**	RO	0x1	0= FIFO is not empty. 1= FIFO is empty. NOT_EMPTY = 0x0 FIFO is not empty. EMPTY = 0x1 FIFO is empty.
4	**FIFO_FULL**	RO	0x0	0= FIFO is not full. 1= FIFO is full. NOT_FULL = 0x0 FIFO is not full. FULL = 0x1 FIFO is full.
3	**DEV3_ERROR**	RO	0x0	0= No error condition present on ATA/NAND Device 3. 1= An error has occurred on ATA/NAND Device 3 (Timeout or compare failure, depending on COMMAND_MODE).
2	**DEV2_ERROR**	RO	0x0	0= No error condition present on ATA/NAND Device 2. 1= An error has occurred on ATA/NAND Device 2 (Timeout or compare failure, depending on COMMAND_MODE).
1	**DEV1_ERROR**	RO	0x0	0= No error condition present on ATA/NAND Device 1. 1= An error has occurred on ATA/NAND Device 1 (Timeout or compare failure, depending on COMMAND_MODE).
0	**DEV0_ERROR**	RO	0x0	0= No error condition present on ATA/NAND Device 0. 1= An error has occurred on ATA/NAND Device 0 (Timeout or compare failure, depending on COMMAND_MODE).

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

13.5.9. GPMI Debug Information Register Description

The GPMI Debug Information Register provides a read-back path for diagnostics to determine the current operating state of the GPMI controller.

HW_GPMI_DEBUG 0x8000C080

Table 494. HW_GPMI_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
READY3	READY2	READY1	READY0	WAIT_FOR_READY_END3	WAIT_FOR_READY_END2	WAIT_FOR_READY_END1	WAIT_FOR_READY_END0	SENSE3	SENSE2	SENSE1	SENSE0	DMAREQ3	DMAREQ2	DMAREQ1	DMAREQ0	CMD_END				UDMA_STATE				BUSY	PIN_STATE			MAIN_STATE			

Table 495. HW_GPMI_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**READY3**	RO	0x0	Read-only view of Ready Line 3.
30	**READY2**	RO	0x0	Read-only view of Ready Line 2.
29	**READY1**	RO	0x0	Read-only view of Ready Line 1.
28	**READY0**	RO	0x0	Read-only view of Ready Line 0.
27	**WAIT_FOR_READY_END3**	RO	0x0	Read-only view of WAIT_FOR_READY command end of channel 3. This view sees the toggle state.
26	**WAIT_FOR_READY_END2**	RO	0x0	Read-only view of WAIT_FOR_READY command end of channel 2. This view sees the toggle state.
25	**WAIT_FOR_READY_END1**	RO	0x0	Read-only view of WAIT_FOR_READY command end of channel 1. This view sees the toggle state.
24	**WAIT_FOR_READY_END0**	RO	0x0	Read-only view of WAIT_FOR_READY command end of channel 0. This view sees the toggle state.
23	**SENSE3**	RO	0x0	Read-only view of sense state of channel 3. A value of "1" indicates that a read and compare command failed or a timeout occured.
22	**SENSE2**	RO	0x0	Read-only view of sense state of channel 2. A value of "1" indicates that a read and compare command failed or a timeout occured.
21	**SENSE1**	RO	0x0	Read-only view of sense state of channel 1. A value of "1" indicates that a read and compare command failed or a timeout occured.
20	**SENSE0**	RO	0x0	Read-only view of sense state of channel 0. A value of "1" indicates that a read and compare command failed or a timeout occured.
19	**DMAREQ3**	RO	0x0	Read-only view of DMA request line for channel 3. This view sees the toggle state.
18	**DMAREQ2**	RO	0x0	Read-only view of DMA request line for channel 2. This view sees the toggle state.
17	**DMAREQ1**	RO	0x0	Read-only view of DMA request line for channel 1. This view sees the toggle state.

Table 495. HW_GPMI_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	**DMAREQ0**	RO	0x0	Read-only view of DMA request line for channel 0. This view sees the toggle state.
15:12	**CMD_END**	RO	0x0	Read-only view of the Command End toggle to DMA. One per channel
11:8	**UDMA_STATE**	RO	0x0	USM_IDLE = 4'h0, idle USM_DMARQ = 4'h1, DMA req USM_ACK = 4'h2, DMA ACK USM_FIFO_E = 4'h3, FIFO empty USM_WPAUSE = 4'h4, WR DMA Paused by device USM_TSTRB = 4'h5, Toggle HSTROBE USM_CAPTUR = 4'h6, Capture Stage, (data sampled with DSTROBE is valid) USM_DATOUT = 4'h7, Change Burst DATAOUT USM_CRC = 4'h8, Source CRC to Device USM_WAIT_R = 4'h9, Waiting for DDMARDY- USM_END = 4'ha; Negate DMAACK (end of DMA)
7	**BUSY**	RO	0x0	When asserted, the GPMI is busy. Undefined results may occur if any registers are written when BUSY is asserted. DISABLED = 0x0 The GPMI is not busy. ENABLED = 0x1 The GPMI is busy.
6:4	**PIN_STATE**	RO	0x0	Parameter PSM_IDLE = 3'h0, PSM_BYTCNT = 3'h1, PSM_ADDR = 3'h2, PSM_STALL = 3'h3, PSM_STROBE = 3'h4, PSM_ATARDY = 3'h5, PSM_DHOLD = 3'h6, PSM_DONE = 3'h7. PSM_IDLE = 0x0 PSM_BYTCNT = 0x1 PSM_ADDR = 0x2 PSM_STALL = 0x3 PSM_STROBE = 0x4 PSM_ATARDY = 0x5 PSM_DHOLD = 0x6 PSM_DONE = 0x7
3:0	**MAIN_STATE**	RO	0x0	Parameter MSM_IDLE = 4'h0, MSM_BYTCNT = 4'h1, MSM_WAITFE = 4'h2, MSM_WAITFR = 4'h3, MSM_DMAREQ = 4'h4, MSM_DMAACK = 4'h5, MSM_WAITFF = 4'h6, MSM_LDFIFO = 4'h7, MSM_LDDMAR = 4'h8, MSM_RDCMP = 4'h9, MSM_DONE = 4'hA. MSM_IDLE = 0x0 MSM_BYTCNT = 0x1 MSM_WAITFE = 0x2 MSM_WAITFR = 0x3 MSM_DMAREQ = 0x4 MSM_DMAACK = 0x5 MSM_WAITFF = 0x6 MSM_LDFIFO = 0x7 MSM_LDDMAR = 0x8 MSM_RDCMP = 0x9 MSM_DONE = 0xA

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

GPMI XML Revision: 1.43

14. HARDWARE ECC ACCELERATOR (HWECC)

This chapter describes the DMA-based hardware ECC accelerator (HWECC) available on the STMP36xx. It provides detailed descriptions of how to use the Reed-Solomon ECC accelerator. Programmable registers are described in Section 14.4.

14.1. Overview

The hardware ECC accelerator provides a forward error-correction function for improving the reliability of various storage media that can be attached to the STMP36xx. Modern high-density NAND flash devices, for example, presume the existence of forward error-correction algorithms, because permitting some soft or hard bit errors within the flash device allows a much higher yield and therefore lower-cost storage devices.

The hardware ECC block is comprised of a robust algorithm for multi-bit error correction using Reed-Solomon block codes. Having a DMA-based hardware accelerator for this function allows the CPU to focus on signal processing for enhanced functionality and to operate at lower clock frequencies and voltages for improved battery life. The CPU is *not* directly involved in generated parity symbols or checking for the errors. The hardware ECC accelerator is illustrated in Figure 49.

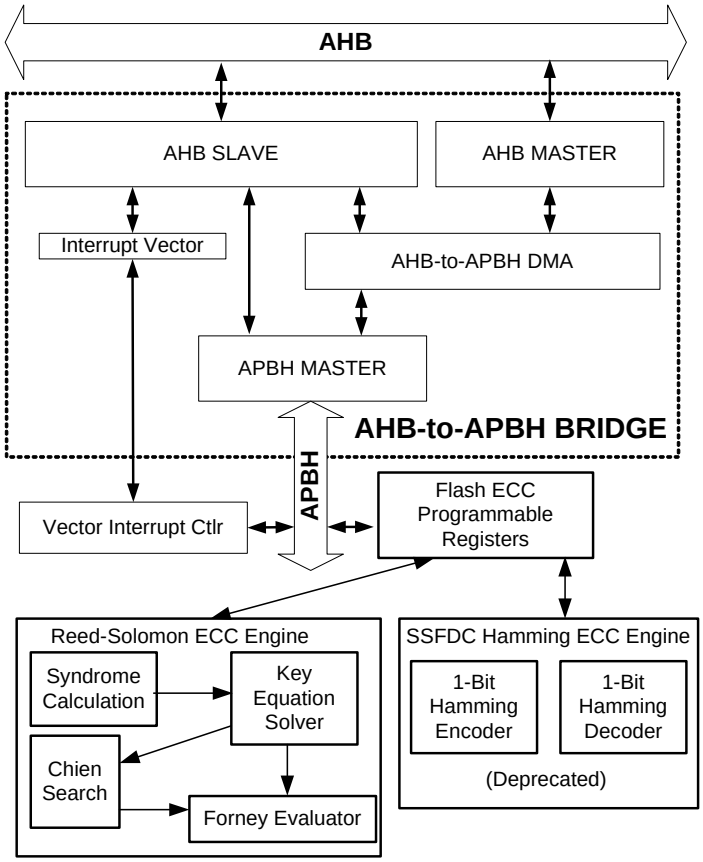


Figure 49. Hardware ECC Accelerator Block Diagram

14.2. Reed-Solomon ECC Accelerator

The Reed-Solomon algorithm is capable of correcting up to 4 9-bit symbols in a 512-byte block. Thus, up to 36 bits in error can be corrected in a 512-byte block, provided they are clustered within no more than 4 9-bit symbols. This algorithm generates 9 bytes (8 symbols) of error code or parity (sometimes called syndromes) per 512-byte block. The parity bytes are stored in the spare area at the end of each NAND flash page.

To understand how the Reed-Solomon algorithm is implemented on the STMP36xx, consider the case where there is a 512-byte data block located in the on-chip RAM that needs to be written to a NAND flash device. Further, assume that a 9-byte Reed-Solomon parity field is to be written into the 16-byte spare area of the 528-byte NAND flash page. Assume that the GPMI media interface is used to write the resultant 521 bytes of data and parity from on-chip memory to the NAND flash device.

- Channel commands in APBH DMA Channel 0 are used to point to the data block in either on-chip or off-chip RAM (as shown in Figure 50).
- The Reed-Solomon (RS) algorithm uses 9-bit symbols. Thus, a 512-byte data block encompasses 455 1/9 symbols.
- As the data is read from on-chip RAM, the hardware appends 47 8/9 zero-pad symbols to form the basic 511-symbol RS block.
- This block is treated as a large polynomial and is divided by the hardware using the mathematics of Galois Fields[1].
- The hardware retains the 8-symbol (72 bits or 9 bytes) remainder from this division, which it then stores as the parity for the block. Channel command words in the same APBH DMA channel (0) are used to store the parity into on-chip RAM.
- The GPMI DMA can then be started to copy the 521 bytes to the NAND flash device. Of course, both units can be fully overlapped.

It is likely that ECC is to be calculated on more than 512 bytes, such as the case of MLC or AG-AND. For those devices, it may be important to protect all metadata, which can be up to 7 additional bytes (page size is usually limited (physically or virtually) to 528 bytes. If 9 bytes are used for parity syndrome data, then there are 512 data plus 7 bytes of metadata available.

It is not necessary for the data to be ordered as: data, metadata, parity. In fact, the decode error report requires that any corrections be applied on 16-bit boundaries, so it makes sense to align the data and parity bytes on that boundary. While not required, it is suggested to place 512 bytes of data, then 9 bytes of parity data, and then the 7 bytes of metadata in a 528-byte page. The DMA engine can use chained descriptors to read/write the data that way, making it seem like one continuous transfer.

Channel command word processing in the APBH DMA allows the buffer to start on an arbitrary byte boundary within system memory.

1.Oliver Pretzel, “Error-Correction Codes and Finite Fields,” Oxford Univ. Press, 1992 ISBN 0-19-269067-1.

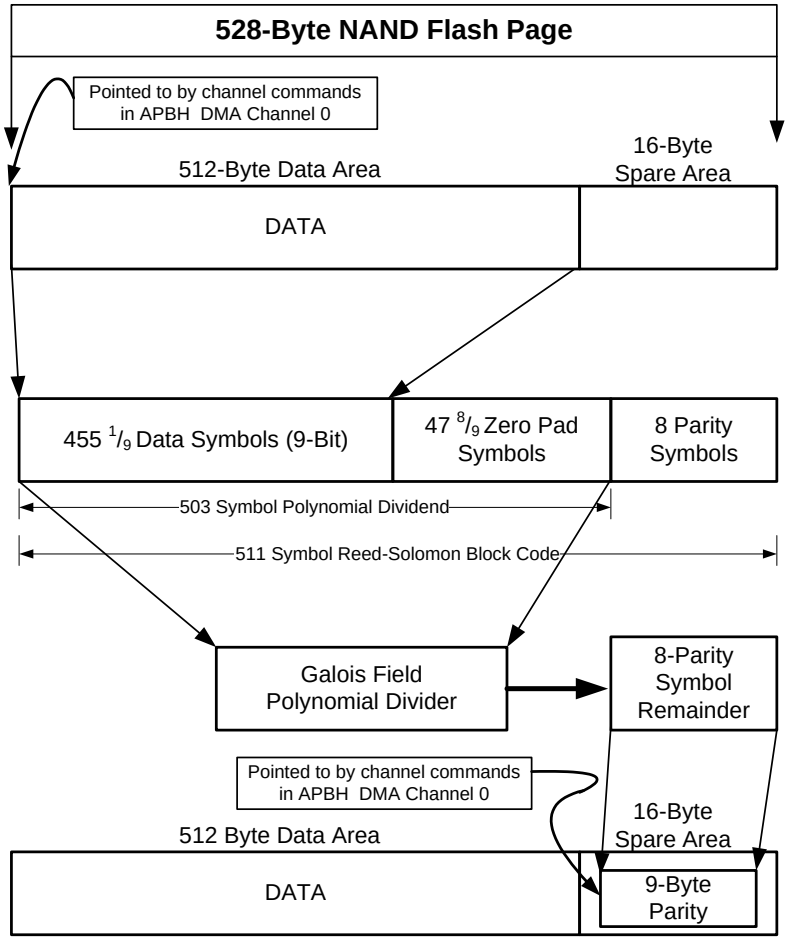


Figure 50. Hardware ECC Reed-Solomon Block Coding—Encoder

Utilizing the DMA engine capabilities, the DMA command chain would point to the 512 bytes of data, then the additional 7 bytes of metadata at the end of the spare area, and then finally to the first 9 bytes of the spare area for storing the returned parity bytes. This is advantageous because the RS-ECC block does not correct the data in place (as with the STMP35xx), and the corrections returned are half-word (16-bit) aligned. Placing the parity data at the end of the 528-byte data+spare area means copying, shifting, and masking is involved in correcting the data, whereas placing the parity bytes at the end of the data allows simpler error correction (if needed).

14.2.1. Reed-Solomon Encoding

The RS encoder flowchart in Figure 51 shows the detailed steps involved in programming and using the hardware ECC's Reed-Solomon encoder. This flowchart shows how to use the HWECC block with the APHB DMA, which is the normal operating mode.

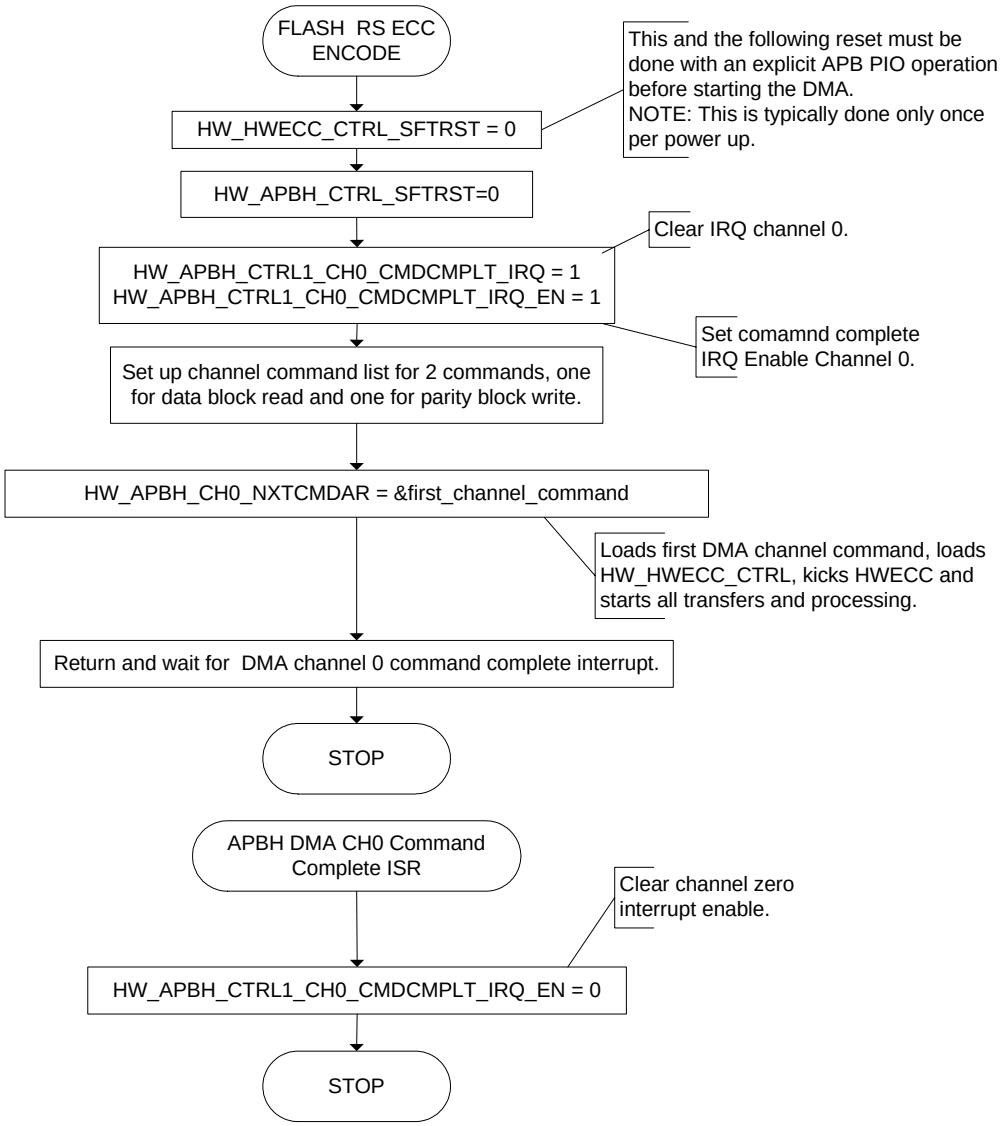


Figure 51. Hardware ECC Reed-Solomon Encode Flowchart

To use the encoder with the DMA:

- Create a DMA command chain with two command structures on it (as shown in Figure 52). The first command structure points to the 512-byte data block that is to be RS-encoded. The second points to a 12-byte (3-word) area to receive the computed parity. Only 9 bytes will be written into this 12-byte buffer.

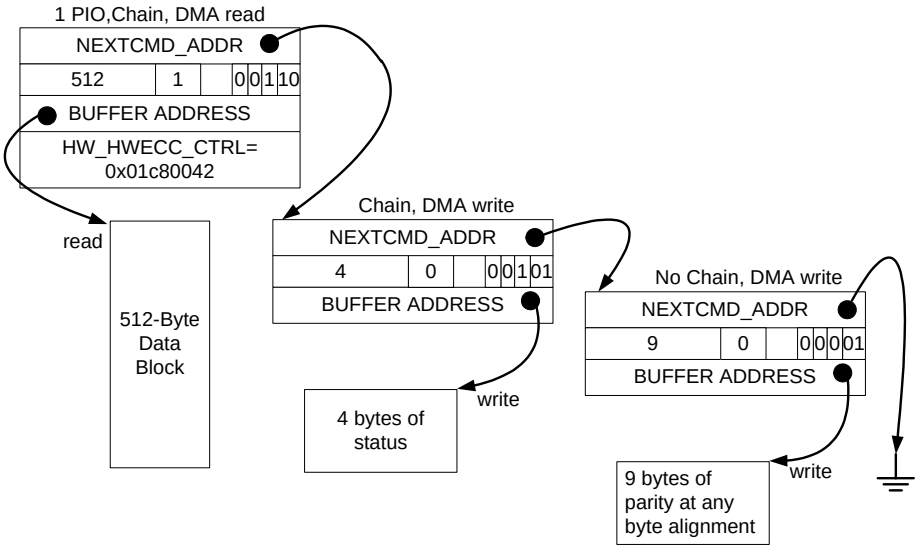


Figure 52. Hardware ECC Reed-Solomon Encode DMA Chain

- To use the encoder, first turn off the module-wide soft reset bit before starting any DMA activity. Note that turning off the soft reset must take place by itself, prior to programming the rest of the control register.
- Program the remainder of the desired HW_HWECC_CTRL register contents by modifying the first DMA channel command structure. DMA channel commands have the option to include a variable number of PIO data transfers that occur before data DMA transfers begin. In this case, the first channel command is initialized to transfer one word to the HWECC PIO space before starting the data DMA. DMA PIO transfers always begin at the device's base address. Thus, this command copies a 32-bit value from the end of the channel command structure to the HW_HWECC_CTRL register. Set bits [1:0] of this channel command word to a value of 2, which will be loaded into HW_HWECC_CTRL_ECC_SEL, causing the device to operate in Reed-Solomon Encode mode when the copy occurs.
- Since the hardware ECC is a memory-to-memory DMA device, its DMA utilization is nominally limited only by the encoder's demand for data. This natural limit may use too much DMA bus bandwidth over its command time. As a result, the HW_HWECC_CTRL_DMAWAIT_COUNT bit field can specify additional wait cycles to insert between the DMA cycle requests to reduce the hardware ECC's short-term utilization.
- Do NOT program the KICK bit to one in the channel command word. The last thing the DMA controller does after performing the PIO copies and before waiting on DMA requests from the HWECC is to set the KICK bit via a specific hardware signal connecting the two blocks. (Note: The automatic KICK from the DMA engine to the HWECC block only occurs if there is PIO done by the DMA to the HWECC.)
- Any previous command-complete interrupt status would have been cleared by writing a one to the interrupt bits' clear address prior to starting the DMA channel command chain processing. Software can then poll the DMA command complete bit for channel 0, waiting for it to be set to one. However, this typically takes hundreds of clock cycles. To get full overlap of the CPU, the GPMI, and the

hardware ECC module, use the APHB DMA Channel 0 command-complete interrupt, source bit HW_APBH_CTRL1:0. When this interrupt is received, the GPMI block can be scheduled to write the entire page to the NAND hardware device

The “footprint” of the RS parity bits in system memory is shown in Table 496. This footprint is for the case where the DMA buffer address is 32-bit word-aligned. Note that the other three starting byte alignments are supported by the DMA channel, as well. The byte aligner built into the shared DMA aligns them so that the data copied from the HW_HWECC_DATA register by the DMA is ***ALWAYS*** 32-bit word-aligned, even though the system memory footprint may not be 32-bit word-aligned (see Table 497).

Table 496. HW_ECC Reed-Solomon Parity Bytes in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ALL_ONES	ALL_ZEROES	Set to zero by the encoder																													
RS_PARITY3								RS_PARITY2								RS_PARITY1								RS_PARITY0							
RS_PARITY7								RS_PARITY6								RS_PARITY5								RS_PARITY4							
Unused and Unwritten																								RS_PARITY8							

Table 497. HW_ECC Reed-Solomon Parity Bytes, Unaligned in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Unused and Unwritten								ALL_ONES	ALL_ZEROES	Set to zero by the encoder																					
RS_PARITY2								RS_PARITY1								RS_PARITY0								set to zero by the encoder							
RS_PARITY6								RS_PARITY5								RS_PARITY4								RS_PARITY3							
Unused and Unwritten																RS_PARITY8								RS_PARITY7							

14.2.2. Reed-Solomon Decoding

When a page is read from NAND flash, its RS parity must be checked and if correctable errors are found, they must be corrected. This decoding process can also be fully overlapped with CPU execution. The RS decoder flowchart in Figure 53 shows the steps involved in programming the hardware ECC’s Reed-Solomon decoder, and Figure 54 summarizes the process.

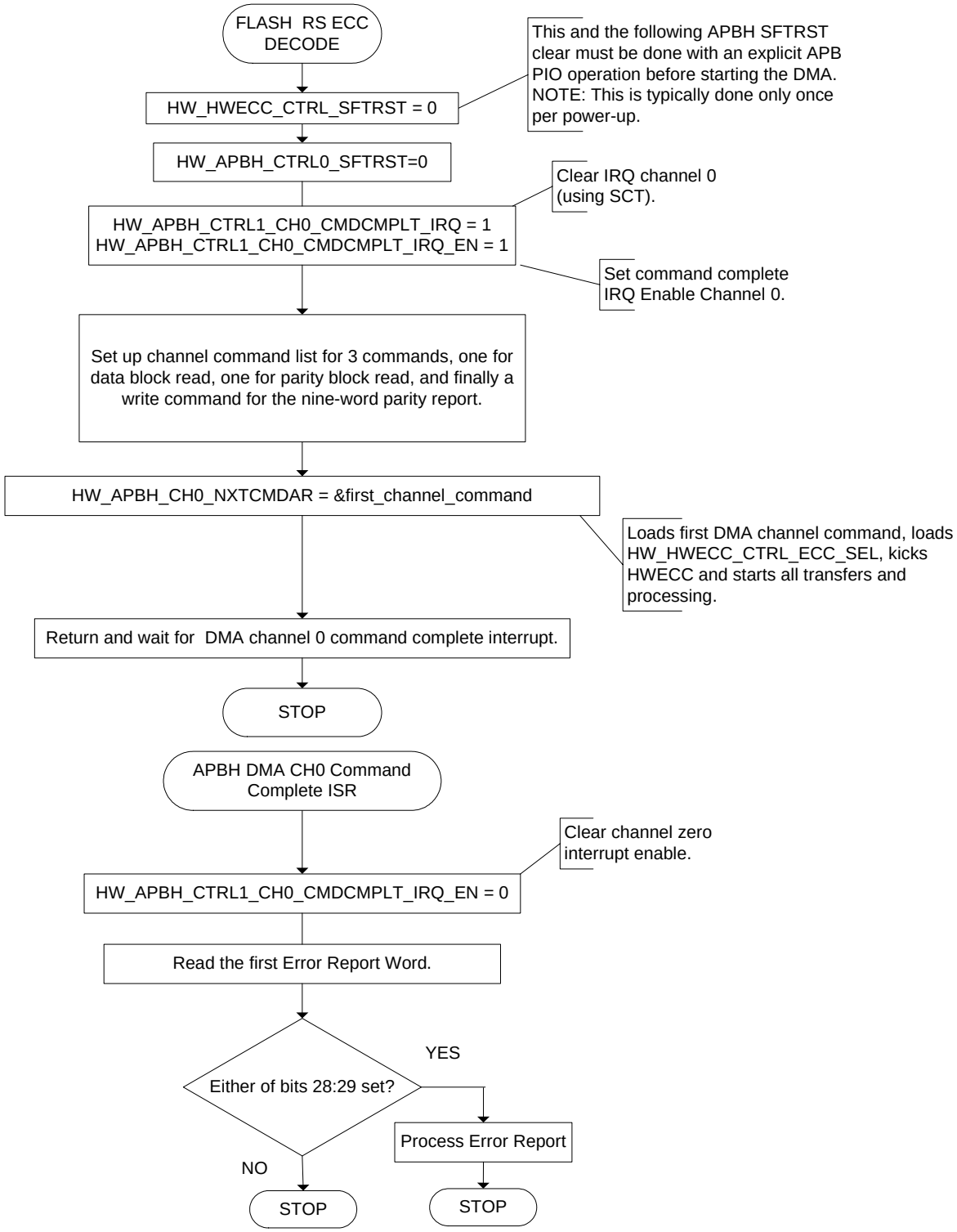


Figure 53. Hardware ECC Reed-Solomon Decode Flowchart

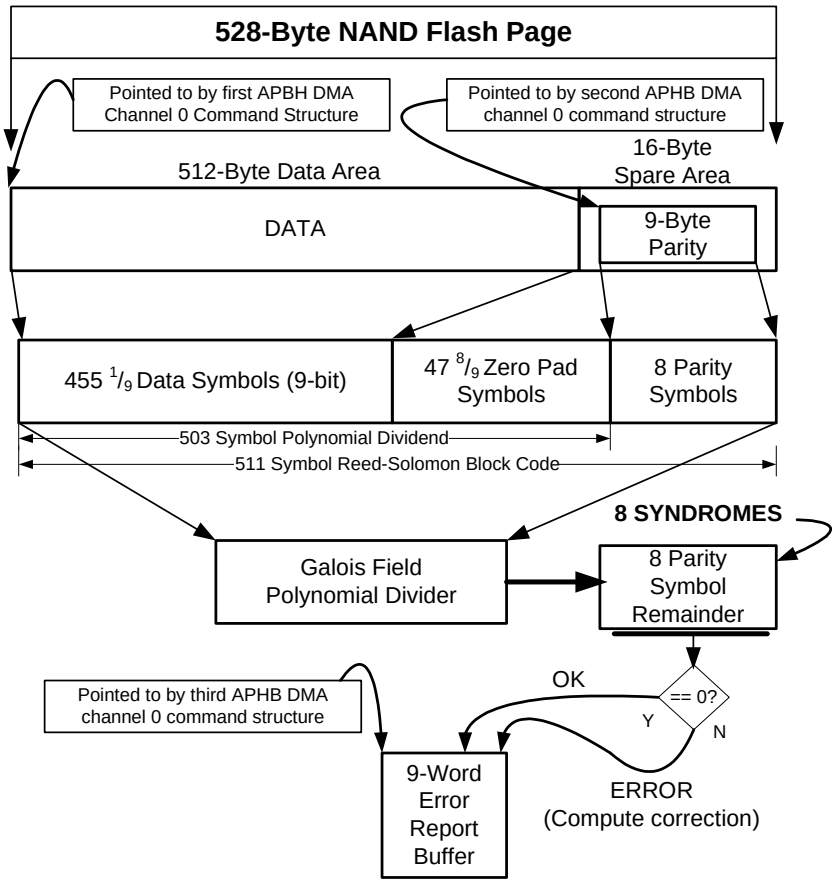


Figure 54. Hardware ECC Reed-Solomon Block Coding—Decoder Phase 1

Conceptually, an APHB DMA Channel 0 command chain with three (or more) command structures linked together is used to perform the RS decode operation (as shown in Figure 55).

- Notice that in this case, the first two DMA command structures point into the either on-chip or off-chip RAM buffer where bytes were read from the NAND flash device, i.e., the data block and the parity block.
- The decoder is initialized to read the data block, append the zero pad, and perform the polynomial division, this time with the supplied parity bytes.
- If the resulting division yields a zero remainder, then no errors are present and the hardware ECC block can immediately report back to firmware.
- If the remainder was non-zero, then it further examines the syndrome bytes to determine which bits must be corrected within the data block or parity block, if possible (not all errors are correctable).
- The third APBH DMA command structure is used to write an error report structure into system memory. This error report structure includes error summary information, as well information on exactly how firmware can correct the error.

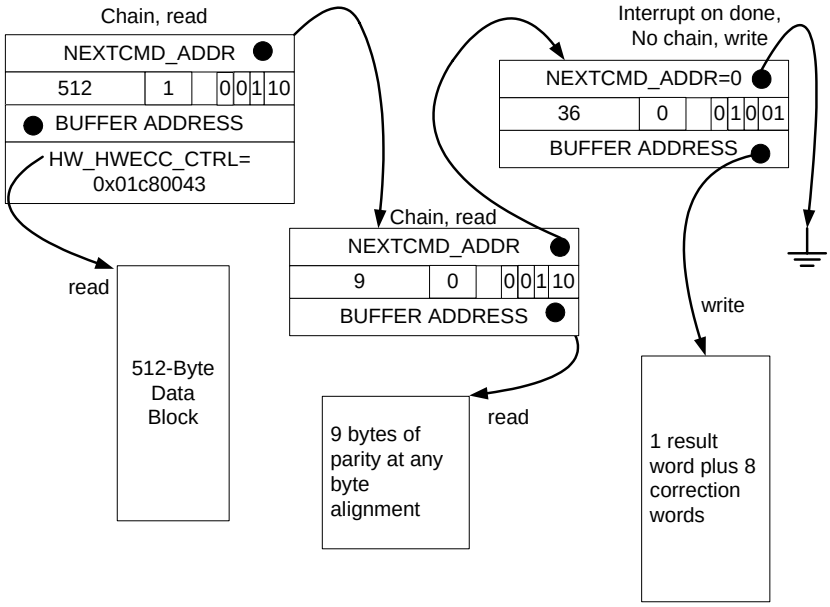


Figure 55. Hardware ECC Reed-Solomon Block Decode DMA Chain

Unlike its predecessor, the STMP36xx does not automatically correct bit errors found by the Reed-Solomon. Instead it supplies up to eight pairs of index and mask values that can be used to correct the data and/or parity blocks (see Table 498).

- The indices should be treated as applying to one of two spaces: the data space (where the MSB of the index is set to zero) or the parity space (where the MSB of the index is set to one).
- For optimum performance, these arrays must be half-word (16-bit aligned).
- Positive indices indicate that the error is in the data block, and negative indices indicate an error in the parity block.
- To correct an error, the index sign bit is checked. If positive, an unsigned short reference into the data block is formed, and the half-word at that location is XORed with the MASK value corresponding to the INDEX used.
- When an INDEX/MASK pair has no error to repair, then both half-words are written as zeroes by the hardware.

NOTE: The HWECC *always* writes all nine 32-bit words of the error report in RS mode whether an error is detected or not. Thus, one can chain the DMA command structures for a large number of data blocks together and let the HWECC work on a whole 2048 or 4096 (or more) byte page at once. When the DMA command complete interrupt arrives, firmware can then check the error correction state for each block. This allows large units of work to be scheduled without the need for frequent CPU interrupts. Also note that if there are corrections to be made, the index values are 1-based, so software must subtract one from them before applying the mask value.

Table 498. HW_ECC Reed-Solomon Decoder Error Report Buffer in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ALL_ONES	ALL_ZEROES	ERROR	UNCORR	Reserved, reads zeroes								ECC EXCEPTION				Reserved, reads zero							Number of bits in error					NUMBER of symbols in ERRORS			
INDEX[0]																MASK[0]															
INDEX[1]																MASK[1]															
INDEX[2]																MASK[2]															
INDEX[3]																MASK[3]															
INDEX[4]																MASK[4]															
INDEX[5]																MASK[5]															
INDEX[6]																MASK[6]															
INDEX[7]																MASK[7]															

One recommended organization for the error report DMA command structure is as follows:

```
struct {
  DMACmdStruc * pNextCommand;
  unsigned long  XferSizeWord;
  unsigned short * pBuffer;  // point to next word
  unsigned long ErrorReportStatus
  struct {
     short Index;  // use plus or minus to select data or parity block
     unsigned short Mask; //if Mask is zero then no error for this index
  } Report[8];
} ErrorReportDMACommand
```

As firmware walks the DMA chain to check ECC results for multiple blocks in a command transaction, it can quickly examine the ERROR bit in the MSB of the first word to determine if further corrective action is required. For the anticipated NAND flash error rates, corrective action will not be required for most blocks.

If corrective action is indicated, then the UNCORR bit should next be checked; if it is set, no corrective action can be reliably performed for the block. If there are errors that are correctable, then firmware should examine each INDEX/MASK pair and perform the indicated XOR operations. There are never more than eight INDEX/MASK pairs generated by the HWECC in Reed-Solomon mode.

As the RS decoder reads the data block and the 9-byte parity block, it records either of two special conditions, i.e., that all of the bits are one or that all of the bits are zero. The all-ones case for both parity and data indicates an erased page in the NAND device.

To summarize, the APBH DMA command chain for a Reed-Solomon decode operation is shown in Figure 55. Three DMA command structures must be present for each block decoded by the HWECC. The three DMA command structures for multiple DATA/PARITY blocks can be chained together to make larger units of work for the HWECC, and each will produce an appropriate error report structure.

The RS decoder processes the 511-symbol block code in three phases. All phases may not be necessary, for example when no errors are found or when uncorrectable errors are found. The three phases are:

1. **Syndrome Calculation Phase (SC)**—This is the process of reading in all of the symbols of the block and continuously dividing by the generator polynomial for the field. The eight syndromes are calculated as the remainder of this division and must be examined, as described above. This phase takes approximately 700 cycles for a 512-byte data block, with no planned DMA wait states added.
2. **Key Equation Solver Phase (KES)**—Once the eight syndromes have been calculated, a set of eight linear equations in eight unknowns is formed. The process of solving these equations and selecting from the numerous possible solutions constitutes the KES phase. The partial solution is obtained by dividing a polynomial based on the syndromes by a Euclidian polynomial. This division, again using the mathematics of Galois Fields, yields two polynomials, the Error Evaluator (EE) polynomial and the Error Locator (EL) polynomial. The EE polynomial is the remainder of this division and is zero if an uncorrectable (non-solvable) case exists. The hardware terminates with an uncorrectable error in this case. This phase takes up to 560 HCLKs, with no planned DMA wait states added.
3. **Chien Search and Forney Evaluator Phase (EVAL)**—This phase takes the EE and EL polynomials from the KES phase and uses Chien's algorithm for finding the locations of the errors based on the EE polynomial. The method basically involves substituting all 512 9-bit symbols into the EE polynomial. All non-zero results of these substitutions represent the locations of the various errors. Another GF division is performed at this point to determine the error value or the correction to apply at the symbol in error location. This phase consumes approximately 550 HCLKs, with no planned DMA wait states added. The EVAL phase terminates either with an uncorrectable error interrupt or simply a "done" interrupt. Done is reported in either case.

If uncorrectable errors occurred, it is up to software to determine how to deal with a bad block. One strategy might be to reread the data from NAND flash in the hope that enough soft errors will have been removed to make correction successful on a second pass.

14.2.3. Reed-Solomon Decoding Using PIO Debug Mode

The block is connected only as a PIO device to the APBH bus. Even though it is designed to work with the DMA controller integrated in the APBH bridge, all transfers to and from the block are programmed I/O (PIO) read or write cycles. When the DMA is ready to write to the HW_HWECC_DATA register, it does so with standard APB write cycles. When it is ready to read from the HW_HWECC_DATA register, it does so with standard APB read cycles. There are four DMA-related signals that connect the HWECC to the DMA, *but* all data transfers are standard PIO cycles on the APB. The state of these four signals can be seen in the HW_HWECC_DEBUG0 register.

Thus, is it is possible to completely exercise the HWECC block for diagnostic purposes, using only load and store instructions from the CPU, without ever starting the DMA controller. This section describes how to interact with the block using PIO operations and also defines the block's detailed behavior.

Whenever the HW_HWECC_CTRL register is written either by the CPU or the DMA, it establishes the basic operation mode for the block, e.g., RS encode or RS decode. Refer to the HW_HWECC_CTRL_ECC_SEL bit field. If the HW_HWECC_CTRL register is written with a one in the KICK bit, then the operation begins and the HWECC attempts to read the data block by toggling its PDMAREQ signal to the DMA. Notice that the PDMAREQ and PENCMD signals are defined as toggle signals. They change state to signify either a request for another DMA word or a notification that the current command transfer is ended by the HWECC. Diagnostic software should poll these signals to determine when the HWECC is ready for another DMA write and can then supply the data by storing a 32-bit word to the HW_HWECC_DATA register, just as the DMA would do in normal operation.

To perform a Reed-Solomon decode using PIO debug mode (for 512-byte data block size), diagnostic software would perform the following:

1. Turn off the soft reset bit, HW_HWECC_CTRL_SFTRST.
2. Write a value of 0x01C80043 to HW_HWECC_CTRL, selecting RS decode mode.
3. Set HW_HWECC_CTRL_RUN to one to simulate a DMA kick
4. Wait for HW_HWECC_DEBUG0_DMA_REQUEST status bit to toggle.
5. Write four bytes of the DATA block data to the HW_HWECC_DATA register.
6. Repeat steps 3, 4, and 5 until 512 bytes have been written to HW_HWECC_DATA, four at a time.
7. Wait for HW_HWECC_DEBUG0_DMA_REQUEST status bit to toggle.
8. Write four bytes of the parity block data to the HW_HWECC_DATA register.
9. Repeat steps 6, 7, and 8 until all 9 bytes of parity have been written.
10. Wait for HW_HWECC_DEBUG0_DMA_REQUEST to toggle.
11. Read the first Error Report word from HW_HWECC_DATA.
12. Repeat steps 10, 11, and 12 until all nine words of the Error Report have been read.
13. Write a zero to the HW_HWECC_CTRL_RUN bit and the operation is now complete.

For debug purposes, the HWECC makes certain intermediate results available such as the Syndrome calculation results or the key equation solver results. These intermediate variables are available for debug purposes in HW_HWECC_DEBUGx.

14.3. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

14.4. Programmable Registers

The following registers are available for programmer access and control of the hardware ECC accelerator.

14.4.1. Hardware ECC Accelerator Control Register Description

The Hardware ECC Accelerator Control Register provides overall control of the hardware ECC accelerator.

HW_HWECC_CTRL 0x80008000
HW_HWECC_CTRL_SET 0x80008004
HW_HWECC_CTRL_CLR 0x80008008
HW_HWECC_CTRL_TOG 0x8000800C

Table 499. HW_HWECC_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSRVD4					NUM_SYMBOLS									RSRVD3			DMAWAIT_COUNT					RSRVD2	BYTE_ENABLE	ECC_SEL	ENC_SEL	RSRVD1	UNCORR_IRQ	UNCORR_IRQ_EN	RUN

Table 500. HW_HWECC_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set this bit to zero to enable normal HWECC operation. Set this bit to one (default) to disable clocking with the HWECC and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the HWECC block to its default state.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29:25	**RSRVD4**	RO	0x0	Reserved, always set these bits to zero.
24:16	**NUM_SYMBOLS**	RW	0x1C8	Number of RS symbols (9-bits/symbol) to encode/decode. The maximum data-block size for RS is 503 symbols. 8 parity symbols are appended to generate an RS-Codeword of 511 symbols. 0x1C8 represents a 512-byte data block. 0x1CE represents a 519-byte data block.
15:13	**RSRVD3**	RO	0x0	Reserved, always set this bits to zero.
12:8	**DMAWAIT_COUNT**	RW	0x0	This bit field specifies the number of HCLKs to insert before requesting a DMA transfer. A value of 2 causes the HWECC state machine to delay two clocks after it is ready to request to a DMA cycle until it toggles the PDMAREQ line. This field acts as a throttle on the bandwidth consumed by the HWECC block. This field can be loaded by the DMA.
7	**RSRVD2**	RO	0x0	Reserved, always set this bit to zero.

Table 500. HW_HWECC_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	**BYTE_ENABLE**	RW	0x0	When ECC_SELECT = 0 (RS-Mode), this bit must be set if NUM_SYMBOLS does not exactly cover the desired number of bytes (i.e., num_syms * 9 / 8 != 0). This will force the HWECC to mask the byte adjacent to the last byte intended to be included in the data-block. (Note: Should be set to 1 for 512- or 519-byte blocks)
5	**ECC_SEL**	RW	0x0	The ECC select field determines the operation to be carried out by the HWECC block. 0= HW_ECC_RS 1= HW_ECC_SSFDC (deprecated) This field can be loaded by the DMA.
4	**ENC_SEL**	RW	0x0	Determines whether to perform an encode or decode for the correction algorithm specified in ECC_SEL. 0=Encode. 1=Decode
3	**RSRVD1**	RO	0x0	Reserved, always set this bit to zero.
2	**UNCORR_IRQ**	RW	0x0	Uncorrectable Error Interrupt Request. An uncorrectable error has been detected. Consult DEBUG0 for further information as to the cause.
1	**UNCORR_IRQ_EN**	RW	0x0	Set this bit to enable UNCORR_IRQ.
0	**RUN**	RW	0x0	For debug purposes, the HWECC can be kicked off by setting this bit to one. In this mode, software diagnostics can simulate the operation of the DMA by reading and writing the appropriate number of words from the DMA Read and Write registers. In normal operation, the HWECC is kicked off after the last setup PIO cycle of a DMA command.

DESCRIPTION:

The HWECC Accelerator Control Register is used to select the specific type of operation to be performed by the HWECC, e.g., Reed-Solomon decode. Once kicked off by the DMA, the selected command processes data supplied by the DMA in a specific order, i.e., data block read, parity block read, followed by writing the error correction result block. This register contains bit fields that throttle the DMA request rate and other bit fields that control various debug modes. It also contains the overall soft reset bit. This bit is set to one at reset and must be turned off before any other bit fields are set and before any DMA operations commence. Note: This is typically done once per power-up.

EXAMPLE:

```
HW_HWECC_CTRL.U = 0x00000000; // turn off the soft reset bit before starting DMA transfers
// all other bit fields are set by the first DMA command
```

14.4.2. Hardware ECC Accelerator Status Register Description

The Hardware ECC Accelerator Status Register provides overall status of the hardware ECC accelerator.

HW_HWECC_STAT 0x80008010
HW_HWECC_STAT_SET 0x80008014

HW_HWECC_STAT_CLR 0x80008018
HW_HWECC_STAT_TOG 0x8000801C

Table 501. HW_HWECC_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSDEC_PRESENT	RSENC_PRESENT	SSDEC_PRESENT	SSENC_PRESENT	RSRVD3																											

Table 502. HW_HWECC_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSDEC_PRESENT**	RO	0x0	0= Reed-Solomon decode error correction encoding and decoding is not supported in this product.
30	**RSENC_PRESENT**	RO	0x0	0= Reed-Solomon encode error correction encoding and decoding is not supported in this product.
29	**SSDEC_PRESENT**	RO	0x1	0= Deprecated SSFDC-Decode error correction encoding and decoding is supported in this product.
28	**SSENC_PRESENT**	RO	0x1	0= Deprecated SSFDC-Encode error correction encoding and decoding is not supported in this product.
27:0	**RSRVD3**	RO	0x0	Reserved, always set these bits to zero.

DESCRIPTION:

The HWECC STAT register provides visibility into the run-time status of the HWECC. The register also reflects the HWECC configurations supported in this version of the STMP36xx.

EXAMPLE:

```
HW_HWECC_STAT.U = 0x00000000;
```

14.4.3. Hardware ECC Accelerator Debug Register 0 Description

The hardware ECC accelerator internal state machines and signals can be seen in this ECC debug register.

HW_HWECC_DEBUG0 0x80008020
HW_HWECC_DEBUG0_SET 0x80008024
HW_HWECC_DEBUG0_CLR 0x80008028
HW_HWECC_DEBUG0_TOG 0x8000802C

Table 503. HW_HWECC_DEBUG0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4		DMA_PENDCMD	DMA_PREQ	SYMBOL_STATE				RSRVD3		CTRL_STATE						ECC_EXCEPTION				RSRVD2		NUM_BIT_ERRORS						RSRVD1	NUM_SYMBOL_ERRORS		

Table 504. HW_HWECC_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**RSRVD4**	RO	0x0	Reserved, always set these bits to zero.
29	**DMA_PENDCMD**	RO	0x0	This read-only field indicates the state of the DMA End Command signal as it is sent from the HWECC to the DMA.
28	**DMA_PREQ**	RO	0x0	This read-only field indicates the state of the DMA Request command signal as it is sent from the HWECC to the DMA.
27:24	**SYMBOL_STATE**	RO	0x0	A copy of the HWECC symbol state-machine bits are visible in this register for diagnostic and validation purposes.
23:22	**RSRVD3**	RO	0x0	Reserved, always set these bits to zero.
21:16	**CTRL_STATE**	RO	0x0	A copy of the HWECC control state-machine bits are visible in this register for diagnostic and validation purposes.
15:12	**ECC_EXCEPTION**	RO	0x0	This read-only field indicates the reason for termination of the most recent operation. 0000= No exception 0001= RS, degree of lambda exceeds 4 0010= RS, lambda is all zeroes 0100= RS, degree of lambda not equal number of roots or lambda, i.e duplicate roots 1000= SSFDC, more than one error (deprecated) This information is stored as part of the error correction result block.
11:10	**RSRVD2**	RO	0x0	Reserved, always set these bits to zero.
9:4	**NUM_BIT_ERRORS**	RO	0x0	This read-only field indicates the number of bit errors detected and/or corrected. Refer to the ECC_EXCEPTION field for detailed information about uncorrectable errors. This information is stored as part of the error correction result block.

Table 504. HW_HWECC_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**RSRVD1**	RO	0x0	Reserved, always set these bits to zero.
2:0	**NUM_SYMBOL_ERRORS**	RO	0x0	This read-only field indicates the number of symbol errors detected and/or corrected. Refer to the ECC_EXCEPTION field for detailed information about uncorrectable errors. This information is stored as part of the error correction result block.

DESCRIPTION:

The HW_HWECC_DEBUG0 register provides access to various internal state information which might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_HWECC_DEBUG0.U; // diagnostic programs can read and act upon various bit fields.
```

14.4.4. Hardware ECC Accelerator Debug Register 1 Description

The hardware ECC accelerator internal state machines and signals can be seen in this ECC debug register.

HW_HWECC_DEBUG1 0x80008030
HW_HWECC_DEBUG1_SET 0x80008034
HW_HWECC_DEBUG1_CLR 0x80008038
HW_HWECC_DEBUG1_TOG 0x8000803C

Table 505. HW_HWECC_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1					SYNDROME2									SYNDROME1									SYNDROME0								

Table 506. HW_HWECC_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSRVD1**	RO	0x00	Reserved, always set these bits to zero.
26:18	**SYNDROME2**	RO	0x000	Syndromes visible for debug
17:9	**SYNDROME1**	RO	0x0000	Syndromes visible for debug
8:0	**SYNDROME0**	RO	0x0000	Syndromes visible for debug

DESCRIPTION:

The HW_HWECC_DEBUG1 register provides access to various internal state information that might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_HWECC_DEBUG1.U; // diagnostic programs can read and act upon various bit fields.
```

14.4.5. Hardware ECC Accelerator Debug Register 2 Description

The hardware ECC accelerator internal state machines and signals can be seen in this ECC debug register.

HW_HWECC_DEBUG2 0x80008040
HW_HWECC_DEBUG2_SET 0x80008044
HW_HWECC_DEBUG2_CLR 0x80008048
HW_HWECC_DEBUG2_TOG 0x8000804C

Table 507. HW_HWECC_DEBUG2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1					SYNDROME5									SYNDROME4									SYNDROME3								

Table 508. HW_HWECC_DEBUG2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSRVD1**	RO	0x00	Reserved, always set these bits to zero.
26:18	**SYNDROME5**	RO	0x000	Syndromes visible for debug
17:9	**SYNDROME4**	RO	0x0000	Syndromes visible for debug
8:0	**SYNDROME3**	RO	0x0000	Syndromes visible for debug

DESCRIPTION:

The HW_HWECC_DEBUG2 register provides access to various internal state information that might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_HWECC_DEBUG2.U; // diagnostic programs can read and act upon various bit fields.
```

14.4.6. Hardware ECC Accelerator Debug Register 3 Description

The hardware ECC accelerator internal state machines and signals can be seen in this ECC debug register.

HW_HWECC_DEBUG3 0x80008050
HW_HWECC_DEBUG3_SET 0x80008054
HW_HWECC_DEBUG3_CLR 0x80008058
HW_HWECC_DEBUG3_TOG 0x8000805C

Table 509. HW_HWECC_DEBUG3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1					OMEGA0									SYNDROME7									SYNDROME6								

Table 510. HW_HWECC_DEBUG3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSRVD1**	RO	0x00	Reserved, always set these bits to zero.
26:18	**OMEGA0**	RO	0x000	KES Polynomials visible for debug
17:9	**SYNDROME7**	RO	0x0000	Syndromes visible for debug
8:0	**SYNDROME6**	RO	0x0000	Syndromes visible for debug

DESCRIPTION:

The HW_HWECC_DEBUG3 register provides access to various internal state information that might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_HWECC_DEBUG3.U; // diagnostic programs can read and act upon various bit fields.
```

14.4.7. Hardware ECC Accelerator Debug Register 4 Description

The hardware ECC accelerator internal state machines and signals can be seen in this ECC debug register.

HW_HWECC_DEBUG4 0x80008060
HW_HWECC_DEBUG4_SET 0x80008064
HW_HWECC_DEBUG4_CLR 0x80008068
HW_HWECC_DEBUG4_TOG 0x8000806C

Table 511. HW_HWECC_DEBUG4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1					OMEGA3									OMEGA2									OMEGA1								

Table 512. HW_HWECC_DEBUG4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSRVD1**	RO	0x00	Reserved, always set these bits to zero.
26:18	**OMEGA3**	RO	0x000	KES Polynomials visible for debug

Table 512. HW_HWECC_DEBUG4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:9	**OMEGA2**	RO	0x0000	KES Polynomials visible for debug
8:0	**OMEGA1**	RO	0x0000	KES Polynomials visible for debug

DESCRIPTION:

The HW_HWECC_DEBUG4 register provides access to various internal state information that might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_HWECC_DEBUG4.U; // diagnostic programs can read and act upon various bit fields.
```

14.4.8. Hardware ECC Accelerator Debug Register 5 Description

The hardware ECC accelerator internal state machines and signals can be seen in this ECC debug register.

HW_HWECC_DEBUG5 0x80008070
HW_HWECC_DEBUG5_SET 0x80008074
HW_HWECC_DEBUG5_CLR 0x80008078
HW_HWECC_DEBUG5_TOG 0x8000807C

Table 513. HW_HWECC_DEBUG5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1					LAMBDA2									LAMBDA1									LAMBDA0								

Table 514. HW_HWECC_DEBUG5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSRVD1**	RO	0x00	Reserved, always set these bits to zero.
26:18	**LAMBDA2**	RO	0x000	KES Polynomials visible for debug
17:9	**LAMBDA1**	RO	0x0000	KES Polynomials visible for debug
8:0	**LAMBDA0**	RO	0x0000	KES Polynomials visible for debug

DESCRIPTION:

The HW_HWECC_DEBUG5 register provides access to various internal state information that might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_HWECC_DEBUG5.U; // diagnostic programs can read and act upon various bit fields.
```

14.4.9. Hardware ECC Accelerator Debug Register 6 Description

The hardware ECC accelerator internal state machines and signals can be seen in this ECC debug register.

HW_HWECC_DEBUG6 0x80008080

HW_HWECC_DEBUG6_SET 0x80008084
HW_HWECC_DEBUG6_CLR 0x80008088
HW_HWECC_DEBUG6_TOG 0x8000808C

Table 515. HW_HWECC_DEBUG6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1														LAMBDA4										LAMBDA3							

Table 516. HW_HWECC_DEBUG6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:18	**RSRVD1**	RO	0x00	Reserved, always set these bits to zero.
17:9	**LAMBDA4**	RO	0x0000	Syndromes visible for debug
8:0	**LAMBDA3**	RO	0x0000	Syndromes visible for debug

DESCRIPTION:

The HW_HWECC_DEBUG6 register provides access to various internal state information that might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_HWECC_DEBUG6.U; // diagnostic programs can read and act upon various bit fields.
```

14.4.10. Hardware ECC Accelerator DMA Read/Write Data Register Description

The hardware ECC accelerator transfers data to and from memory using the DMA integrated into the AHB-to-APBH bridge. The DMA moves write data to the hardware ECC accelerator using standard APB write cycles targeted at the HW_HWECC_DATA register.

HW_HWECC_DATA 0x80008090
HW_HWECC_DATA_SET 0x80008094
HW_HWECC_DATA_CLR 0x80008098
HW_HWECC_DATA_TOG 0x8000809C

Table 517. HW_HWECC_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 518. HW_HWECC_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x00000000	Data written to this PIO address are used by the HWECC as the data block or the parity block.

DESCRIPTION:

In normal operation, the DMA writes to this PIO address as it reads the data block or the parity block. It writes (typically) 128 words (512 bytes) of data block information in RS mode. After writing the data block information to this register, it writes 9 bytes of parity information in 3 four byte words in RS mode. Note that the data block and parity block information can be supplied from two independent DMA commands and can therefore come from two independent buffers in system memory. Once encoding or decoding is completed, the DMA will read from this buffer the parity report or decode report, respectively.

EXAMPLE:

```
  HW_HWECC_DATA.U = 0x12345678; // diagnostic software can write to this register in place of
the DMA
```

HWECC XML Revision: 1.40

15. SYNCHRONOUS SERIAL PORT (SSP)

This chapter describes the synchronous serial port (SSP) included on the STMP36xx. It includes sections on external pins, bit rate generation, frame formats, Motorola SPI mode, Texas Instruments Synchronous Serial Interface (SSI) mode, National Semiconductor Microwire mode, SD/SDIO/MMC mode, and MS mode. Programmable registers are described in Section 15.11.

15.1. Overview

The synchronous serial port is a flexible interface for inter-IC and removable media control and communication. The SSP supports master operation of SPI, Texas Instruments SSI, National Semiconductor Microwire, 1-bit and 4-bit SD/SDIO/MMC, and MS modes. The SPI mode has enhancements to support 1-bit legacy MMC cards. The SSP also supports slave operation for the SPI, SSI, and Microwire modes. The SSP has a dedicated DMA channel in the bridge and can also be controlled directly by the CPU through PIO registers. Figure 56 shows a block diagram of the SSP port included on the STMP36xx.

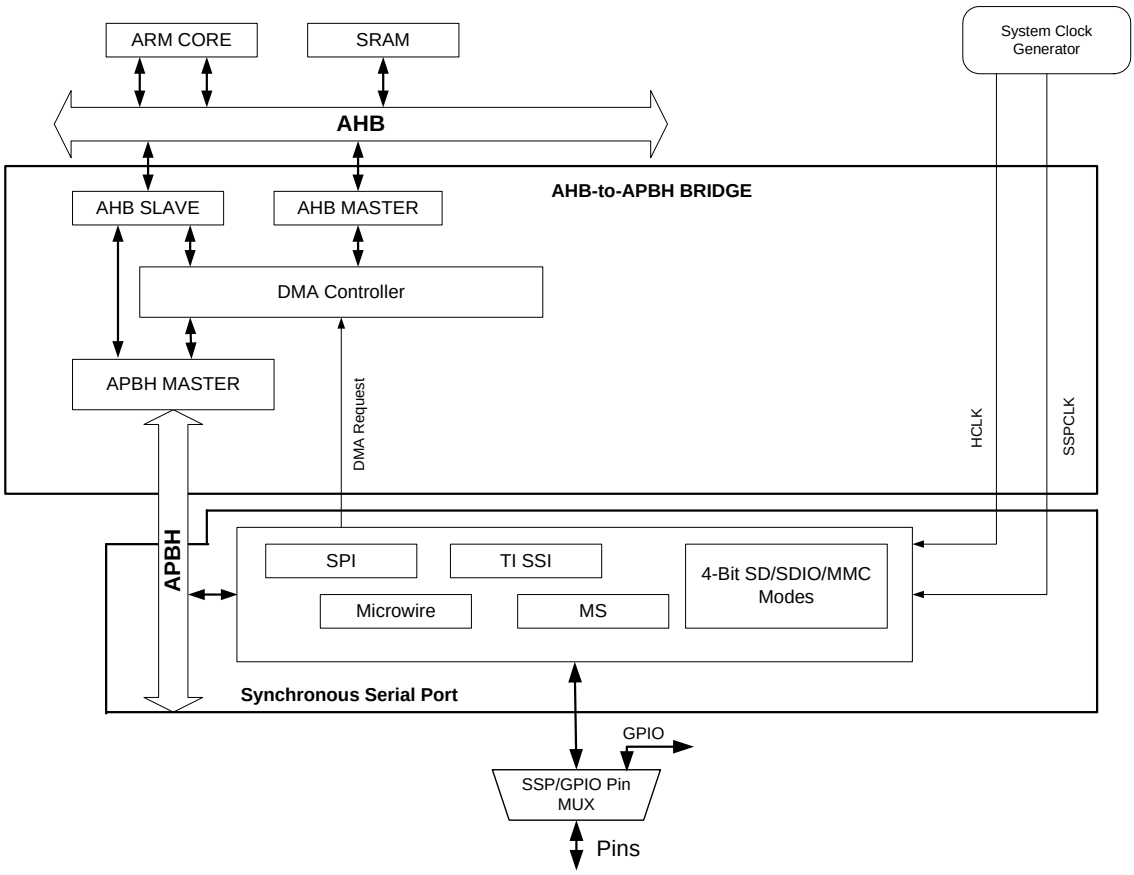


Figure 56. Synchronous Serial Port Block Diagram

15.2. External Pins

Table 519 lists the SSP pin placements for all supported modes.

Table 519. SSP Pin Matrix

PIN NAME	SPI MODE	TI SSI MODE	MICROWIRE MODE	4-BIT SD/SDIO/MMC MODE	MS MODE
SSP_SCK	SCK	CLK	CLK	CLK	CLK
SSP_CMD	MOSI	MOSI	MOSI	CMD	SDIO
SSP_DATA0	MISO	MISO	MISO	DATA0	
SSP_DATA1				DATA1/IRQ	BS
SSP_DATA2				DATA2	
SSP_DATA3	SSn	SSn	SSn	DATA3	
SSP_DETECT					

15.3. Bit Rate Generation

The serial bit rate is derived by dividing down the internal clock SSPCLK. 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 SSP_SCK is defined as follows:

$$SSP_SCK = \frac{SSPCLK}{CLOCKDIVIDE * (1+CLOCK_RATE)}$$

For example, if SSPCLK is 3.6864 MHz, and CPSDVSR=2, then SSP_SCK has a frequency range from 7.2 kHz to 1.8432 MHz. See Chapter 4, “Clock Generation and Control” on page 47, for more clock details.

15.4. Frame Format for SPI, SSI, and Microwire

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:

- Motorola SPI
- Texas Instruments Synchronous Serial Interface (SSI)
- National Semiconductor Microwire

For all three formats, the serial clock (SSP_SCK) is held inactive while the SSP is idle and transitions at the programmed frequency only during active transmissions or reception of data. The idle state of SSP_SCK is used to provide a receive timeout indication, which occurs when the receive FIFO still contains data after a timeout period.

For Motorola SPI and National Semiconductor Microwire frame formats, the serial frame (SSn) pin is active low and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial interface (SSI) frame format, the SSn 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 on data on the rising edge of SSP_SCK, 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 transmit, 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 cycle after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data can be from 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

15.5. Motorola SPI Mode

The SPI mode is used for general inter-component communication and legacy 1-bit MMC cards.

15.5.1. SPI DMA Mode

The SPI bus is inherently a full-duplex bidirectional interface. However, as most applications only require half-duplex data transmission, the STMP36xx has a single DMA channel for the SSP. It can be configured for either transmit or receive. In DMA receive mode, the SPI continuously repeats the word written to its data register. In DMA transmit mode, the SPI ignores the incoming data.

15.5.2. Motorola SPI Frame Format

The Motorola SPI interface is a four-wire interface where the SSn signal behaves as a slave select. The main feature of the Motorola SPI format is that the inactive state and phase of SSP_SCK signal are programmable through the polarity and phase bits within the HW_SSP_CTRL1.

15.5.2.1. Clock Polarity

- When the clock polarity control bit is low, it produces a steady-state low value on the SSP_SCK pin.
- When the clock polarity control bit is high, a steady-state high value is placed on the SSP_SCK pin when data is not being transferred.

15.5.2.2. Clock Phase

The phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted, by either allowing or not allowing a clock transition before the first data-capture edge.

- When the phase control bit is low, data is captured on the first clock-edge transition.
- When the clock phase control bit is high, data is captured on the second clock-edge transition.

15.5.3. Motorola SPI Format with Polarity=0, Phase=0

SIngle and continuous transmission signal sequences for Motorola SPI format with POLARITY=0, PHASE=0 are shown in Figure 57 and Figure 58.

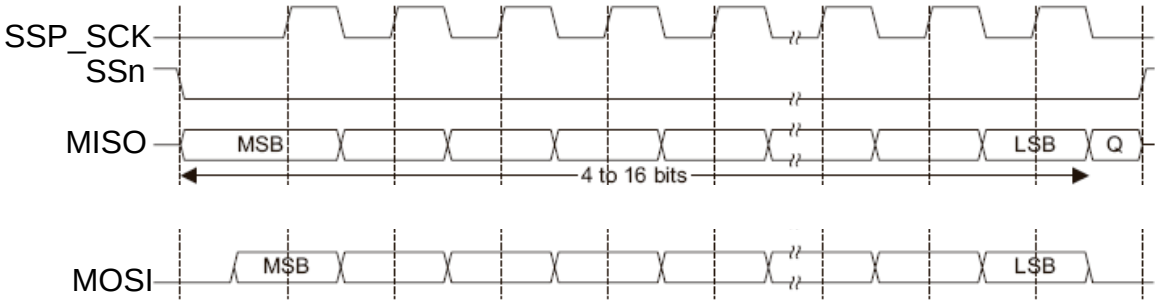


Figure 57. Motorola SPI Frame Format (Single Transfer) with POLARITY=0 and PHASE=0

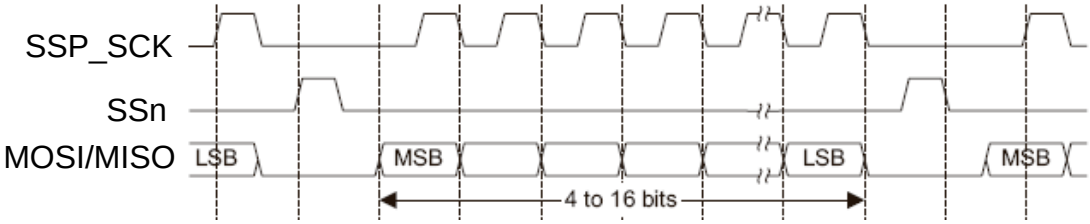


Figure 58. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=0

In this configuration, during idle periods:

- The SSP_SCK signal is forced low.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, SSP_SCK is an output.
- When the SSP is configured as a slave, SSP_SCK is an input.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of the transmission is signified by the SSn master signal being low. This causes slave data to be enabled onto the MISO input line of the master, and the enables the master MOSI output pad.

One half SSP_SCK period later, valid master data is transferred to the MOSI pin. Now that both the master and slave data have been set, the SSP_SCK master clock pin goes high after one further half SSP_SCK period.

The data is now captured on the rising and propagated on the falling edges of the SSP_SCK signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSn signal must be pulsed high between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the PHASE bit is logic zero. Therefore, the master device must raise the SSn pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSn pin is returned to its idle state one SSP_SCK period after the last bit has been captured.

15.5.4. Motorola SPI Format with Polarity=0, Phase=1

The transfer signal sequence for Motorola SPI format with POLARITY=0 and PHASE=1 is shown in Figure 59, which covers both single and continuous transfers.

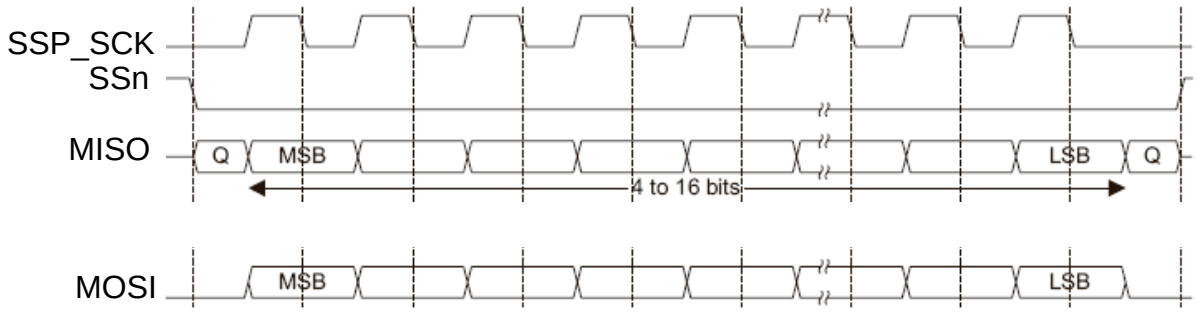


Figure 59. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=1

In this configuration, during idle periods:

- The SSP_SCK signal is forced low.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, the SSP_SCK pad is an output.
- When the SSP is configured as a slave, the SSP_SCK is an input.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of the transmission is signified by the SSn master signal being low. After a further one half SSP_SCK period, both master and slave valid data are enabled with a rising-edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSP_SCK signal.

In the case of a single word transfer, after all bits have been transferred, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

For continuous back-to-back transfers, the SSPFSOUT pin is held low between successive data words and termination is the same as that of a single word transfer.

15.5.5. Motorola SPI Format with Polarity=1, Phase=0

Single and continuous transmission signal sequences for Motorola SPI format with POLARITY=1 and PHASE=0 are shown in Figure 60 and Figure 61.

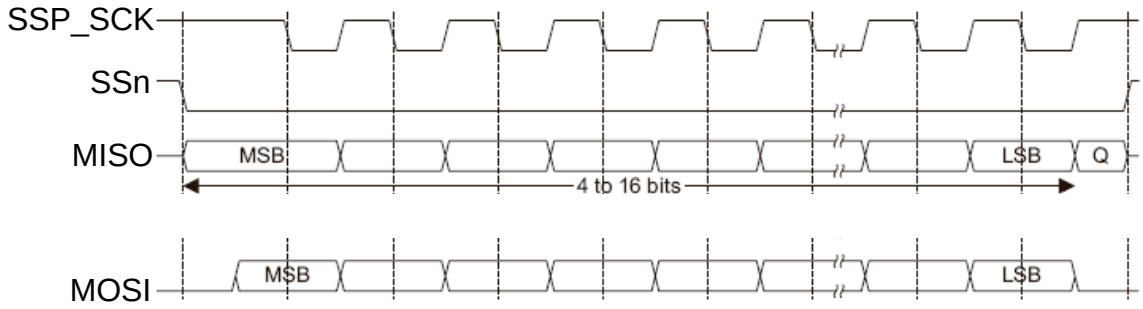


Figure 60. Motorola SPI Frame Format (Single Transfer) with POLARITY=1 and PHASE=0

Note: In Figure 60, Q is an undefined signal.

I

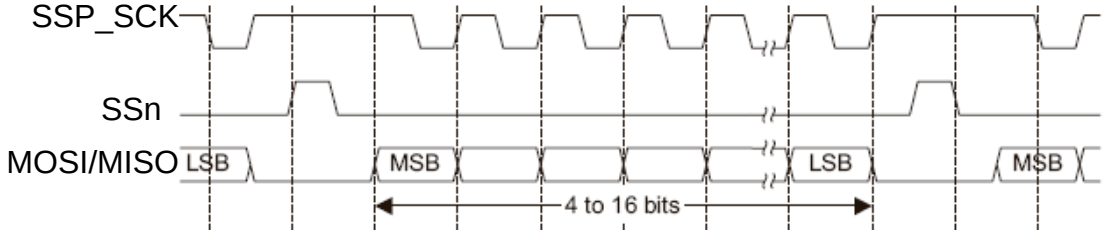


Figure 61. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=1 and PHASE=0

In this configuration, during idle periods:

- The SSP_SCK signal is forced high.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, the SSP_SCK pad is an output.
- When the SSP is configured as a slave, the SSP_SCK is an input.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSn master signal being driven low, which causes slave data to be immediately transferred onto the MISO line of the master, and enabling the master MOSI output pad.

One half-period later, valid master data is transferred to the MOSI line. Now that both master and slave data have been set, the SSP_SCK master clock pin becomes low after one further half SSP_SCK period. This means that data is captured on the falling edges and propagated on the rising edges of the SSP_SCK signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSn signal must be pulsed high between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the PHASE bit is logic zero. Therefore, the master device must raise the SSPSFSSIN pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSn pin is returned to its idle state one SSP_SCK period after the last bit has been captured.

15.5.6. Motorola SPI Format with Polarity=1, Phase=1

The transfer signal sequence for Motorola SPI format with POLARITY=1 and PHASE=1 is shown in Figure 62, which covers both single and continuous transfers.

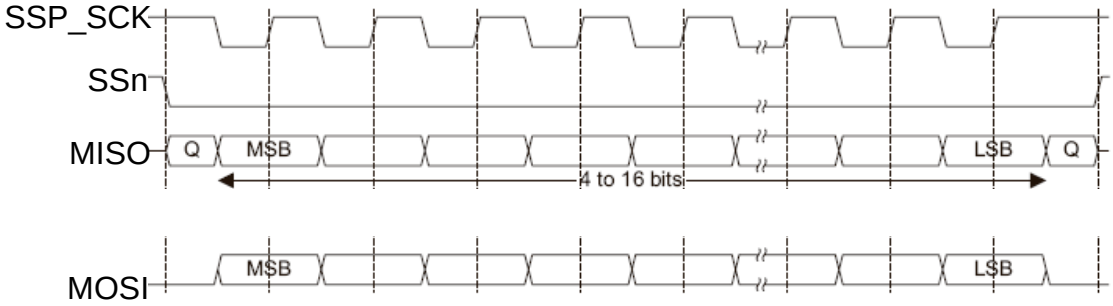


Figure 62. Motorola SPI Frame Format with POLARITY=1 and PHASE=1

Note: In Figure 62, Q is an undefined signal.

In this configuration, during idle periods:

- The SSP_SCK signal is forced high.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, the SSP_SCK pad is an output.
- When the SSP is configured as a slave, the SSP_SCK is an input.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSn master signal being driven low, and MOSI output is enabled. After a further one-half SSP_SCK period, both master and slave are enabled onto their respective transmission lines. At the same time, the SSP_SCK is enabled with a falling edge transition. Data is then captured on the rising edge and propagated on the falling edges of the SSP_SCK signal.

After all bits have been transferred, in the case of a single word transmission, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

For continuous back-to-back transmissions, the SSn pin remains in its active low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSn pin is held low between successive data words and termination is the same as that of the single word transfer.

15.6. Texas Instruments Synchronous Serial Interface (SSI) Mode

Figure 63 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

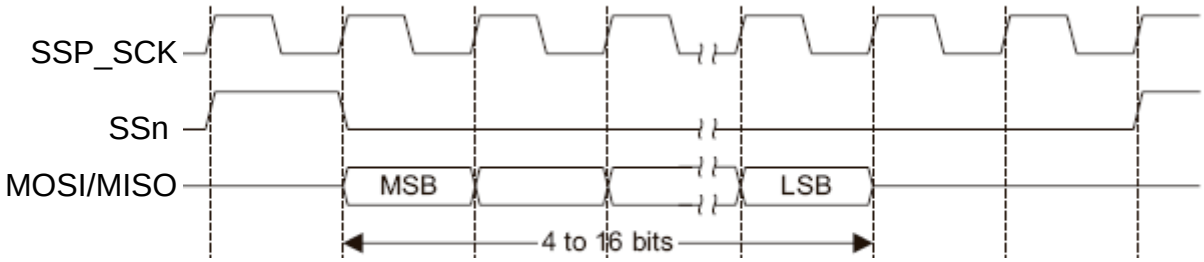


Figure 63. Texas Instruments Synchronous Serial Frame Format (Single Transfer)

In this mode, SSP_SCK and SSn are forced low, and the transmit data line MOSI is three-stated whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, SSn is pulsed high for one SSP_SCK period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSP_SCK, the MSB of the 4-to-16-bit data frame is shifted out on the SSPRXD pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSP_SCK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSP_SCK after the LSB has been latched.

Figure 64 shows the Texas Instrument synchronous serial frame format when back-to-back frames are transmitted.

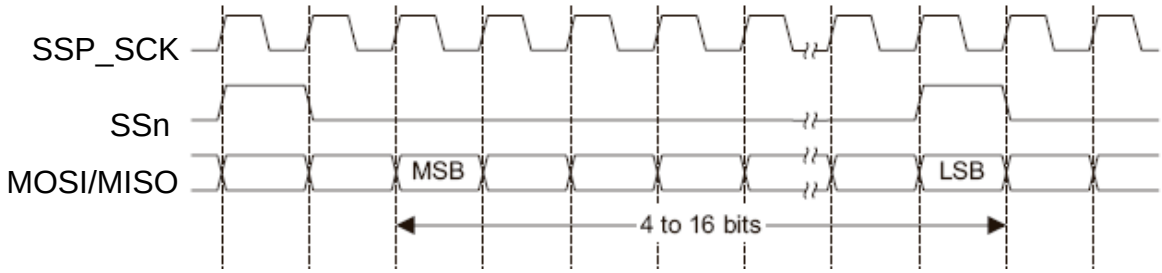


Figure 64. Texas Instruments Synchronous Serial Frame Format (Continuous Transfer)

15.7. National Semiconductor Microwire Mode

Figure 65 shows the National Semiconductor Microwire frame format, again for a single frame. Figure 66 shows the same format when back-to-back frames are transmitted.

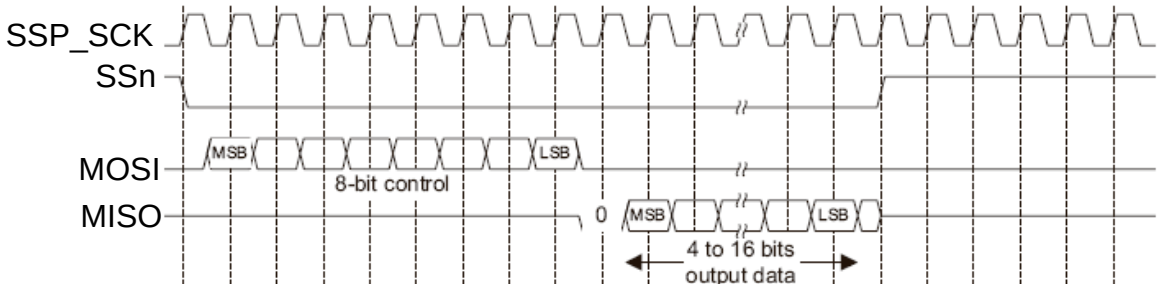


Figure 65. Microwire Frame Format (Single Transfer)

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. 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 cycle after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- The SSP_SCK signal is forced low.
- SSn is forced high.
- The Transmit data line MOSI is forced to high impedance.

A transmission is triggered by writing a control bit to the transmit FIFO. The falling edge of SSn causes the value contained in the bottom entry of the transmit FIFO to

be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the MOSI pin. SSn remains LOW for the duration of the frame transmission. The MISO pin remains three-stated during this transmission.

The off-chip serial slave device latches each control bit onto its serial shifter on the rising edge of each SSP_SCK. After the last bit is latched by the slave device, the control byte is decoded during a one clock cycle wait-state, and the slave responds by transmitting data back to the SSP. Each bit is driven onto MISO line on the falling edge of SSP_SCK. The SSP in turn latches each bit on the rising edge of SSP_SCK. At the end of the frame, for single transfers, the SSn signal is pulled high one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can three-state the receive line either on the falling edge of SSP_SCK after the LSB has been latched in the receive shifter, or when the SSn pin goes high.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSn line is continuously asserted (held low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transmitted from the receive shifter on the falling edge of SSP_SCK, after the LSB of the frame has been latched into the SSP.

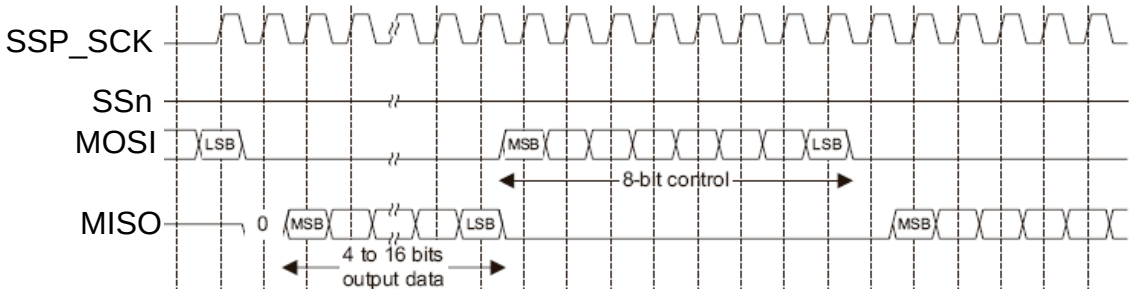


Figure 66. Microwire Frame Format (Continuous Transfer)

15.7.0.1. Setup and Hold Time Requirements on SSn with Respect to SSP_SCK in Microwire Mode

In the Microwire mode, the SSP slave samples the first bit of receive data on the rising edge of SSP_SCK after SSn has gone low. Masters that drive a free-running SSP_SCK must ensure that the SSn signal has sufficient setup and hold margins with respect to the rising edge of SSP_SCK.

Figure 67 illustrates these setup and hold time requirements. With respect to the SSP_SCK rising edge on which the first bit of receive data is to be sampled by the SSP slave, SSn must have a setup of at least two times the period of SSP_SCK on which the SSP operates. With respect to the SSP_SCK rising edge previous to this edge, SSn must have a hold of at least one SSP_SCK period.

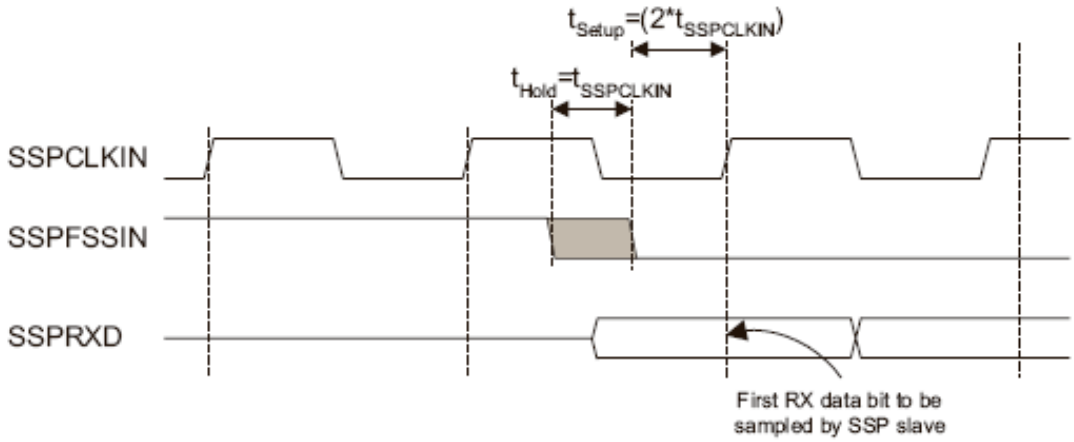


Figure 67. Microwire Frame Format (Continuous Transfer)

15.8. SD/SDIO/MMC Mode

This mode is used to provide high performance with SD, SDIO, MMC, and high-speed (4-bit) MMC cards.

SD/MMC mode supports simultaneous command and data transfers. Commands are sent to the card and responses are returned to the host on the CMD line. Register data, such as card information, is sent as a command response and is therefore on the CMD line. Block data read from or written to the card's flash is transferred on the DAT line(s). The SSP also supports the SDIO IRQ.

The SSP's SD/MMC controller can automatically perform a single block read/write or card register operation with a single PIO setup and RUN. For example, the SD/MMC controller can perform these steps with a single write to the PIO registers:

- Send command to the card.
- Receive response from the card.
- Check response for errors (and assert a CPU IRQ if there is an error).
- Wait for the DAT line(s) to be ready to transfer data (while counting for timeout)
- Transfer a block of data to/from the card.
- Check the CRC or CRC status of received/sent data (and assert IRQ if there is an error).

The SD/MMC controller is generally used with the DMA. Each DMA descriptor is set up the SD/MMC controller to perform a single complex operation as exemplified above. Multiple DMA descriptors can be chained to perform multiple card block transfers without CPU intervention.

15.8.1. SD/MMC Command/Response Transfer

SD/MMC commands are written to the HW_SSP_CMDX registers and sent on the CMD line. Command tokens consist of a start bit (0), a source bit (1), the actual

command, which is padded to 38 bits, a 7-bit CRC and a stop bit (1). The command token format is shown in Table 520.

Table 520. SD/MMC Command/Response Transfer

Line	Start	Source	Data	CRC	End
CMD	0	1 (Host)	38-bit Command	CRC7	1

SD/MMC cards transmit command words with the most significant bit first. After the card receives the command, it checks for CRC errors or invalid commands. If an error occurs, the card withholds the usual response to the command.

After transmitting the end bit, the SSP releases the CMD line to the high-Z state. A pullup resistor on the CMD node keeps it at the 1 state until the response packet is received. The slave waits to issue the reply until the SCK line is clocking again.

After the SSP sends an SD/MMC command, it optionally starts looking for a response from the card. It waits for the CMD line to go low, indicating the start of the response token. Once the SSP has received the Start and Source bits, it begins shifting the response content into the receive shift register. The SSP calculates the CRC7 of the incoming data.

If the card fails to start sending an expected response packet within 64 SCK cycles, then an error has occurred. The command may be invalid or have a bad CRC. After the SSP detects a timeout, it stops any DMA request activity and sets the RESP_TIMEOUT flag. If RESP_TIMEOUT_IRQ_EN is set, then a CPU IRQ is asserted.

The SSP calculates the CRC of the received response and compare it to the CRC received from the card. If they do not match, then the SSP sets the RESP_ERR status flag. If RESP_ERR_IRQ_EN is set, then a CPU IRQ is asserted on a command response CRC mismatch.

The SSP can also compare the 32-bit card status word, known as response R1, against a reference to check for errors. If CHECK_RESP in HW_SSP_CTRL0 is set, then the SSP XORs the response with the XOR field in the HW_SSP_COMPREF register. It then masks the results with the MASK field in the HW_SSP_COMPMASK register. If there are any differences between the masked response and the reference, then an error has occurred. The CPU asserts the RESP_ERR status flag. If RESP_ERR_IRQ_EN is set, then the RESP_ERR_IRQ is asserted. In the ISR, the CPU can read the status word to see which error flags are set.

The regular and long response tokens are shown in Table 521 and Table 522:

Table 521. SD/MMC Command Regular Response Token

Line	Start	Source	Data	CRC	End
CMD	0	0 (Card)	38-bit Response	CRC7	1

Table 522. SD/MMC Command Regular Long Response Token

Line	Start	Source	Data	CRC	End
CMD	0	0 (Card)	117-bit response	CRC16	1

15.8.2. SD/MMC Data Block Transfer

Block data is transferred on the DATA0 pin. In 1-bit I/O mode, the block data is formatted as shown in Table 523. Block data transfers typically have 512 bytes of pay-

load, plus a 16-bit CRC, a Start bit, and an End bit. The block size is programmable with the XFER_COUNT field in the HW_SSP_CTRL0 register. In SD/MMC mode, WORD_LENGTH in the HW_SSP_CTRL1 register field should always be set to 8 bits. Data is always sent Most Significant Bit of the Least Significant Byte first.

The SSP is designed to support block transfer modes only. Streaming modes may not be supported. Figure 68 shows a flowchart of SD/MMC block read and write transfers.

In block write mode, the card holds the DATA0 line low while it is busy. SSP must wait for the DATA0 line to be high for one clock cycle before starting to write a block.

In block read mode, the card begins sending the data when it is ready. The first bit transmitted by the card is a Start bit 0. Prior to the 0 Start Bit, the DATA0 bus is high. After the start bit is received, data is shifted in. The SSP bus width is set using the BUS_WIDTH bit in the HW_SSP_CTRL0 register.

In 1-bit bus mode, the block data is formatted as shown in Table 523.

Table 523. SD/MMC Data Block Transfer 1-Bit Bus Mode

Line	Start	Data	Data	Data	CRC	End
DATA0	0	Data Bit 7 Byte 0	...	Data Bit 0 Byte 511	CRC16	1

In 4-bit I/O mode, the block data is formatted as shown in Table 524.

Table 524. SD/MMC Data Block Transfer 4-Bit Bus Mode

Line	Start	Data	Data	Data	CRC	End
DATA3	0	Data Bit 7 Byte 0	...	Data Bit 3 Byte 511	CRC16	1
DATA2	0	Data Bit 6 Byte 0	...	Data Bit 2 Byte 511	CRC16	1
DATA1	0	Data Bit 5 Byte 0	...	Data Bit 1 Byte 511	CRC16	1
DATA0	0	Data Bit 4 Byte 0	...	Data Bit 0 Byte 511	CRC16	1

15.8.2.1. SD/MMC Multiple Block Transfers

The SSP supports SD/MMC multiple block transfers. The CPU or DMA will configure the SD/MMC controller to issue a Multi-Block Read or Write command. If DMA is used, then the first descriptor issues the multi-block read/write command and receives/sends the first block (512 bytes) of data. Subsequent DMA descriptors only receive/send blocks of data and do not issue new SD/MMC commands. If the card is configured for an open-ended multi-block transfer, then the last DMA descriptor needs to issue a STOP command to the card.

After each block of data has been transferred, the SSP sends/receives the CRC and checks the CRC or the CRC token. If the CRC is okay, then the SSP signals the DMA that it is done.

15.8.2.2. SD/MMC Block Transfer CRC Protection

All block data transferred over the data bus is protected by CRC16. For reads, the SSP calculates the CRC of incoming data and compares it to the CRC16 reference that is provided by the card at the end of the block. If a CRC mismatch occurs, then the block asserts the DATA_CRC_ERR status flag. If DATA_CRC_IRQ_EN is set, then a CPU IRQ is asserted.

For block write operations, the card determines if a CRC error has occurred. After the SSP has sent a block of data, it transmits the reference CRC16. The card compares that to its calculated CRC16. The card then sends a CRC status token on the

DATA bus. It sends a positive status (‘010’) if the transfer was good, and a negative status (‘101’) if the CRC16 did not match. If the SSP receives a CRC bad token, it sets the DATA_CRC_ERROR in the HW_SSP_STATUS register, and then it indicates it to the CPU if DATA_CRC_IRQ_EN is set.

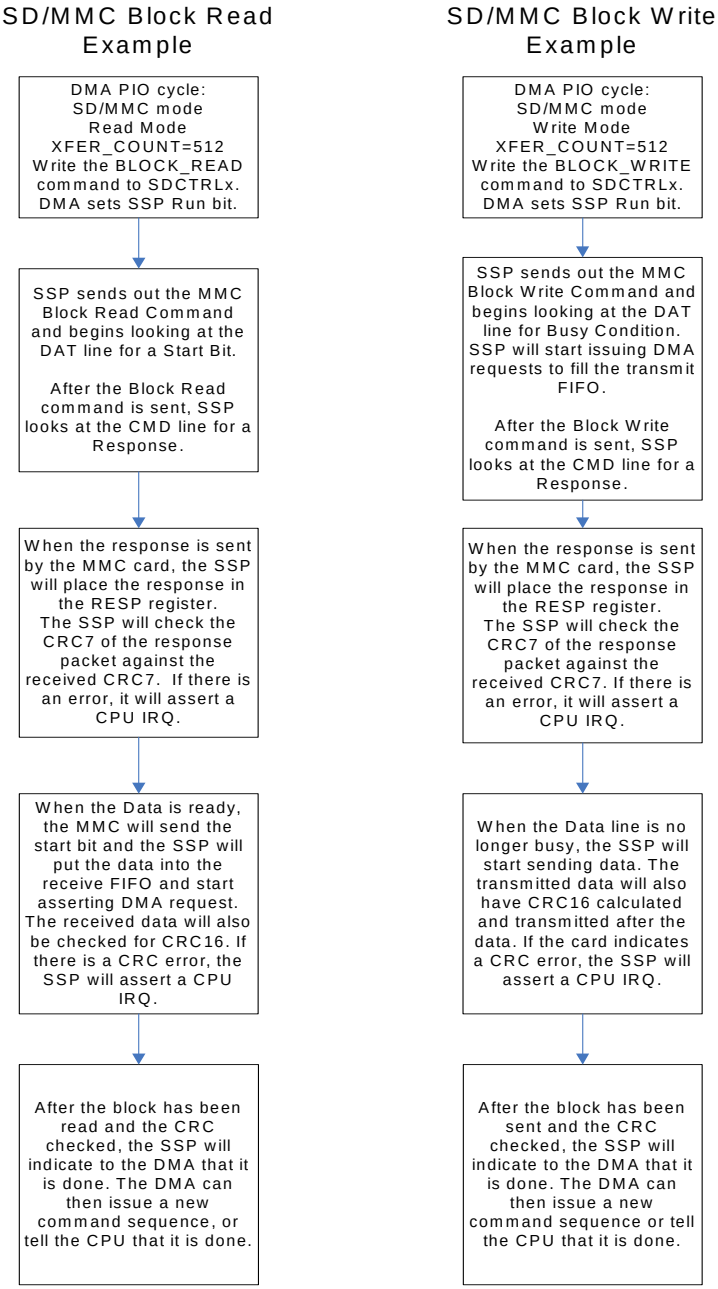


Figure 68. SD/MMC Block Transfer Flowchart

15.8.3. SDIO Interrupts

The SSP supports SDIO interrupts. When the SSP is in SD/MMC mode and the SDIO_IRQ bit in the HW_SSP_CTRL0 register is set, the SSP looks for interrupts on DATA1 during the valid IRQ periods. The valid IRQ periods are defined in the SDIO specification. If the card asserts an interrupt and SDIO_IRQ_EN is set, then the SSP sets the SDIO_IRQ status bit and asserts a CPU IRQ. Other than detecting when card IRQs are valid, the SDIO IRQ function operates independently from the rest of the SSP. After the CPU receives an IRQ, it should monitor the SSP and DMA status to determine when it should send commands to the SDIO card to handle the interrupt.

15.8.4. SD/MMC Mode Error Handling

There are several errors that can occur during SD/MMC operation. These errors can be caused by normal unexpected events, such as having a card removed or unusual events such as a card failure. The detected error cases are listed below. Please note that in all cases below, a CPU IRQ is only asserted if DATA_CRC_IRQ_ENABLE is set in HW_SSP_CTRL1 register.

- **Data Receive CRC Error**—Detected by the SSP after a block receive. If this occurs, the SSP will not indicate to the DMA that the transfer is complete. It will set the DATA_CRC_ERR status flag and assert a CPU IRQ. The ISR should reset the SSP DMA channel and instruct the DMA to re-try the block read operation.
- **Data Transmit CRC Error**—Transmit CRC error token is received from the SD/MMC card on the DAT line after a block transmit. If this occurs, the SSP will not indicate to the DMA that the transfer is complete. It will set the DATA_CRC_ERR status flag and assert a CPU IRQ. The ISR should reset the SSP DMA channel and instruct the DMA to re-try the block write operation.
- **Data Timeout Error**—The SSP TIMEOUT counter is used to detect a timeout condition during data write or read operations. The timeout counts any time that the SSP is waiting on a busy DAT bus. For read operations, the DAT line(s) indicate busy before the card sends the start bit. For write operations, the DAT line(s) may indicate busy after the block has been sent to the card. If the timeout counter expires before the DAT line(s) become ready, the SSP stops any DMA requests, sets the DATA_TIMEOUT status flag, and asserts a CPU IRQ. The ISR should check the status register to see that a data timeout has occurred. It can then reset the DMA channel and SSP to re-try the operation.
- **DMA Overflow/Underflow**—The SSP should stop SCK if the receive FIFO is full or the transmit FIFO is empty. So, a DMA underflow or overflow should not occur. However, if it does due to some unforeseen problem, the RECV_OVRFLW or XMIT_UNDRFLW status bit is set in the SSP status register.
- **Command Response Error**—The SD/MMC card returns an R1 status response after most commands. The SSP can compare the R1 response against a mask/reference pair. If any of the enabled bits are set, then an error has occurred. The SSP stops requesting any DMAs, sets the RESP_ERR status flag, and asserts a CPU IRQ. The CPU can read the SSP status register to see the RESP_ERR flag and read the HW_SSP_SDRESP0 register to get the actual response from the SD/MMC card. That response contains the specific error information. Once the error is understood, the CPU can reset the DMA channel and SSP and re-try the operation or take some other action to recover or inform the user of a non-recoverable error.

- **Command Response Timeout**—If an expected response is not received within 64 SCK cycles, then the command response has timed out. If this occurs, the SSP stops any DMA requests, stops transferring data to the card, sets the RESP_TIMEOUT status flag, and asserts the RESP_TIMEOUT_IRQ. The ISR should read the status register to find that a command response timeout has occurred. It can then decide to reset the DMA channel and SSP and re-try the operation.

15.8.5. SD/MMC Clock Control

- The serial clock (SCK) never runs when the RUN bit is not set.
- SCK runs any time that RUN is set and a data or command is active or pending. If a command has been sent and a response is expected, then SCK continues to run until the response is received. If a data operation is active or if the DAT line is busy, then SCK runs.
- SCK stops running if received command response status R1 indicates an error.
- SCK stops running if a data operation has timed out or a CRC error has occurred.
- SCK stops running after all pending commands and data operations have completed. SCK restarts when a new command or data operation has been requested.

15.9. MS Mode

The SSP MS mode supports 1-bit MS data transfers. The SSP MS mode is designed to work with the STMP36xx DMA controller. The DMA controller's linked descriptor architecture provides a high level of automatic operation without CPU intervention.

15.9.1. MS Mode I/O Pins

The MS standard defines three pins:

- **SDIO**—A bidirectional data pin used for commands and data.
- **BS**: Bus State—The Bus State pin is used to indicate to the card which state it is operating in.
- **SCLK**: Serial Clock—Data changes on falling edges and is latched in on rising edges.

15.9.2. Basic MS Mode Protocol

The MS protocol uses a hierarchy of commands and bus operations to provide access to the internal flash memory, as shown in Figure 69. At a high level, the system may read or write flash pages or access card information or configuration data. Those high-level operations are performed by executing a number of Transfer Protocol Commands, or TPCs. Each TPC is sent to the card in a four-state bus transaction.

The SSP's MS controller automates each four-state bus access associated with a TPC. The CPU or DMA provides higher-level control, putting multiple TPCs together to perform the desired compound functions.

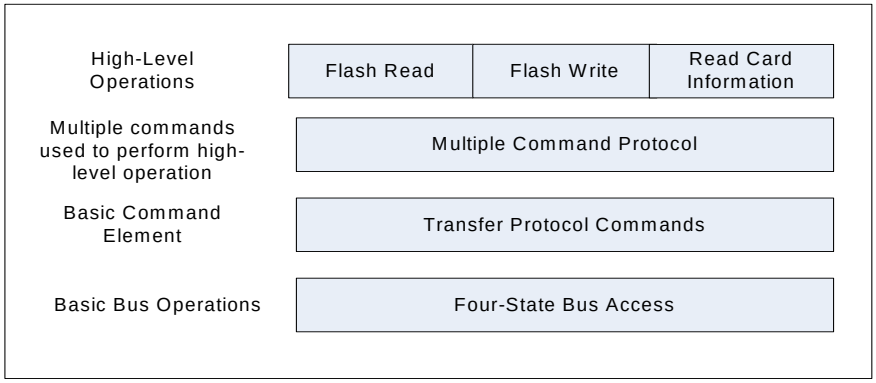


Figure 69. Basic MS Protocols

15.9.3. MS Mode High-Level Operation

The STMP36xx CPU or DMA is responsible for combining several smaller commands (TPCs) into complex operations such as reading or writing flash pages. The DMA's linked descriptor feature is particularly useful for automating MS operations without requiring CPU intervention. Each DMA descriptor commands the SSP to either send a single TPC with associated four-state bus access or wait for a card IRQ during BS0. An example flow chart of the DMA executing a MS page read operation is shown in Figure 70.

15.9.4. MS Mode Four-State Bus Protocol

The four-state bus operations are centered around transitions of the bus state. Each bus state change represents a new operating state of the card, as shown in Figure 71.

In most cases, the card starts in BS0/IRQ. When the card needs to signal the host, then it asserts the IRQ by bringing SDIO high. IRQs are asserted to signal that a requested task, such as a flash page read or write, has completed.

When the host is ready to send a command, it changes the BS signal from low to high, transitioning to BS1. On the next cycle, the host begins transmitting the Transfer Protocol Command (TPC). The TPC is eight bits wide. It includes a four-bit command and the complement of the command for error-checking. The TPCs include:

- **READ_PAGE_DATA—**Reads a 512Byte+CRC16 page from the page buffer.
- **READ_REG—**Reads from the register whose address was previously set. Data length depends on the register.
- **WRITE_PAGE_DATA—**Writes a 512Byte+CRC16 page to the page buffer.
- **WRITE_REG—**Writes to the register whose address was previously set. Data length depends on the register.
- **SET_R/W_REG_ADRS—**Sets the register accessed by READ_REG and WRITE_REG commands. Sends 4Bytes+CRC16.
- **SET_CMD—**Sets the command to be executed by the flash memory controller. Sends 8bits+CRC16. The flash memory controller starts operation with this TPC and sets an INT when it is completed.
- **GET_INT—**Requests the contents of the INT register. Returns 8bits+CRC16.

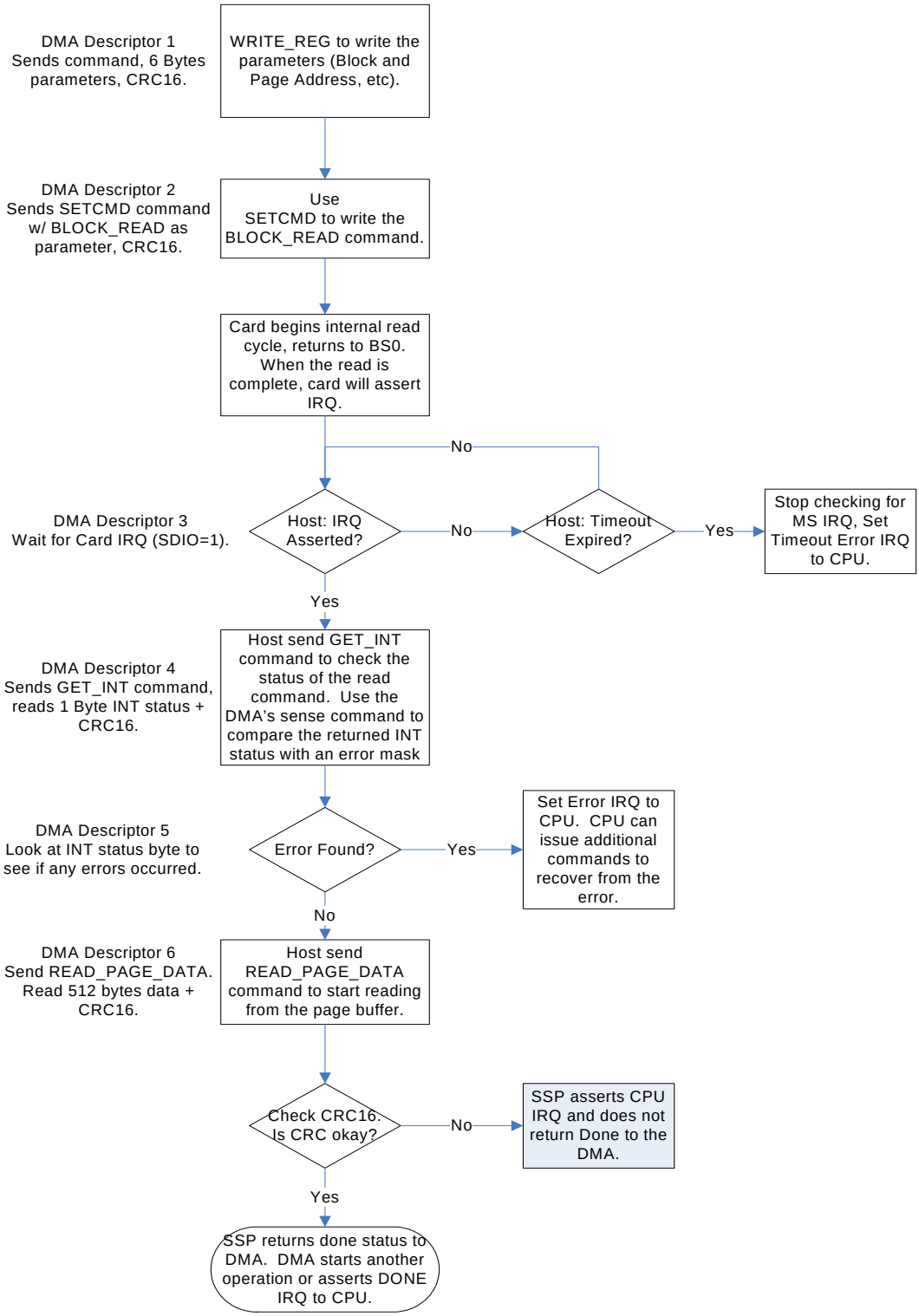


Figure 70. MS Operation Flowchart

After the TPC is sent, the host drives the BS line low again, transitioning to BS2. If the command results in data being written to the card, then the data transfer occurs during BS2. If the command results in data being read from the card, then BS2 is a “handshake” state in which the host waits for the card to indicate that it is ready to send the requested read data to the host. The card toggles the SDIO line when it is ready. In data write operations, BS3 is the handshake state. In that case, the card signals the ready state after it has processed the written data or command enough to return to BS0. In the read-data case, the data is transferred from the card to the host in BS3.

Four-State Read	BS0 IDLE/IRQ	BS1 TPC	BS2 Handshake	BS3 R-Data+CRC
Four-State Write	BS0 IDLE/IRQ	BS1 TPC	BS2 W-Data+CRC	BS3 Handshake

Figure 71. MS Four-State Read and Write

15.9.5. Wait for Card IRQ

Many MS commands, such as flash read/write/erase, take longer than the time allowed during the handshake period to complete. In these cases, the system returns to BS0 while the card is busy. After the card has completed its work (or finds an error during the attempt), then it will IRQ the host by pulling SDIO high. The host then checks the card’s status, as described in Section 15.9.6.

The SSP can be programmed to wait for the card IRQ before transitioning to BS1 and issuing the TPC. To do this, set the WAIT_FOR_IRQ bit in the HW_SSP_CTRL0 register. After the RUN bit is set, the SSP monitors the SDIO line and count for timeout. If the timeout expires before the card asserts the IRQ, then the SSP stops any DMA activity and sets the DATA_TIMEOUT status and the DATA_TIMEOUT_IRQ, if it is enabled.

15.9.6. Checking Card Status

In MS mode, card status and flash page data are transferred over the SDIO line. The card status can be retrieved by issuing a READ_REG TPC after appropriately setting the register pointers with SET_R/W_REG_ADDRS. The interrupt status register is often all that is needed. It can be easily read using the GET_INT TPC. GET_INT always returns eight bits of status while READ_REG can return as many bytes as have been configured. The SSP can check up to 32 bits of status against a reference to check for errors. To check the read data, set the CHECK_RESP bit in the HW_SSP_CTRL0 register and provide a mask and reference in the COMPMASK and COMPREF registers. When checking status, the XFER_SIZE should be set to no more than four bytes because the compare registers are 32 bits wide. If the masked card status does not match the reference, the SSP stops any DMA activity, sets the RESP_ERROR flag, and, if enabled, asserts the RESP_ERROR_IRQ.

15.9.7. MS Mode Error Conditions

There are several errors that can occur during MS operation. These errors can be caused by normal unexpected events, such as having a card removed, or unusual events, such as a card failure. The detected error cases include the following:

- **Data Receive CRC Error—**Detected by the SSP after any data receive. If this occurs, the SSP will not indicate to the DMA that the transfer is complete. It will set the DATA_CRC_ERR status flag and assert a CPU IRQ. The ISR should reset the SSP DMA channel and instruct the DMA to re-try the read operation. Note that in MS mode, a data CRC error can occur on a 512-byte flash page or on a register read. Any TPC that results in data being read from the card can result in this error.
- **Data Transmit CRC Error—**The MS card will check the CRC16 of any data that is sent from the host during BS2. This data may be a 512-byte page of data for flash write, or it may be a register parameter or command as small as one byte. If a CRC error occurs, the card will not return a ready signal during the handshake state. This will cause a timeout in the SSP waiting for the handshake to complete during BS3.
- **Data Timeout Error—**In MS mode, the timeout counter is used while waiting for an IRQ in BS0. If the timeout counter expires before an expected IRQ is asserted, the SSP will set the DATA_TIMEOUT status flag and can assert the DATA_TIMEOUT_IRQ if it is enabled. If the CPU IRQ is used, the ISR should check the status register to see that a data timeout has occurred. It can then reset the DMA channel and SSP to re-try the operation. The CPU can also read the MS Status register to see if the card has an error and/or needs to be reset.
- **Card Status Error—**After a card has signaled an IRQ to the SSP to indicate that a requested flash controller operation has completed, the DMA or CPU will instruct the SSP to issue a GET_INT command to retrieve the interrupt status from the card. The SSP will use its check response (CHECK_RESP) mode to compare the card's status with a reference. If an error is indicated, the SSP will stop DMA activity, set the RESP_ERROR flag and the RESP_ERROR_IRQ if it is enabled.

15.9.8. MS Mode Details

The SSP handles all aspects of a full four-state bus transaction. It uses the command registers that were added for SD/MMC to hold the 8-bit TPC. The data portion of the transaction is handled by the standard data path (FIFO, data register, DMA, etc.). All MS data transactions use a CRC16 for EDC. This CRC16 is different than what is used for SD/MMC. See the MS documentation for specifics.

The SSP monitors the SDIO line and detects the card handshake signal that indicates it is ready to transition from BS3 to BS4 or BS4 to BS0. The MS specification requires that the card indicate it is ready within 12 SCLK cycles. The SSP also receives four continuous toggles on SDIO before going to the next bus state. If the timeout counter expires before the ready handshake has been received, the SSP times out. This results in a DATA_TIMEOUT_IRQ to the CPU.

15.10. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

15.11. Programmable Registers

The following registers provide control for programmable elements of the SSP port.

15.11.1. SSP Control Register 0 Description

HW_SSP_CTRL0 0x80010000
HW_SSP_CTRL0_SET 0x80010004
HW_SSP_CTRL0_CLR 0x80010008
HW_SSP_CTRL0_TOG 0x8001000C

Table 525. HW_SSP_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RUN	HALF_DUPLEX	LOCK_CS	IGNORE_CRC	READ	DATA_XFER	SDIO_IRQ	BUS_WIDTH	WAIT_FOR_IRQ	WAIT_FOR_CMD	LONG_RESP	CHECK_RESP	GET_RESP	ENABLE	XFER_COUNT															

Table 526. HW_SSP_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	SSP Reset. 0: SSP is not reset. 1: SSP is held in reset. After reset, all registers are returned to their reset state.
30	**CLKGATE**	RW	0x1	Gate SSP Clocks. 0: SSP clocks are not gated. 1: SSP clocks are gated. Set this to save power while the SSP is not actively being used. Configuration state is kept while the clock is gated.
29	**RUN**	RW	0x0	SSP Run. 0: SSP is not running. 1: SSP is running. Automatically set during DMA operation.
28	**HALF_DUPLEX**	RW	0x0	1= Ignore receive data when transmitting, and force transmit data=0 when receiving. SPI mode only. (READ bit and DATA_XFER bit indicates TX or RX).
27	**LOCK_CS**	RW	0x0	In SPI mode: This affects the SSn output. 0= Deassert chip select (CS) after transfer is complete. 1= Continue to assert chip select (CS) after transfer is complete. Software must clear this bit at the end of the transfer sequence. In SD/MMC mode: 0= Look for a CRC status token from the card on DATA0 after a block write. 1= Ignore the CRC status response on DATA0 after a write operation. Note that the SD/MMC function should be used when performing MMC BUSTEST_W operation.

Table 526. HW_SSP_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26	**IGNORE_CRC**	RW	0x0	Ignore CRC - SD/MMC, MS - Ignores the Response CRC.
25	**READ**	RW	0x0	Read Mode. When this and DATA_XFER are set, the SSP will read data from the device. If this is not set, then the SSP will write data to the device.
24	**DATA_XFER**	RW	0x0	Data Transfer Mode. When set, transfer XFER_COUNT bytes of data. When not set, the SSP will not transfer any data (command or Wait for IRQ only). In MS mode, this bit selects the destination of read transfers and source for write transfers: set to 1 if using Rx/Tx FIFOs, and set to 0 if using Resp0/Cmd1 registers. When set to 0 in MS mode, the XFER_COUNT field must be 4 or less.
23	**SDIO_IRQ**	RW	0x0	SDIO IRQ Mode. When set, the SSP will check for SDIO card IRQs on DAT1 during valid IRQ periods.
22	**BUS_WIDTH**	RW	0x0	Data Bus Width. Set this bit to 0 for all modes, except for SD which can be either 0 or 1. ONE_BIT = 0x0 data bus is 1-bit wide FOUR_BIT = 0x1 data bus is 4-bits wide
21	**WAIT_FOR_IRQ**	RW	0x0	Wait for MS IRQ. In MS mode, waits for the card to assert an IRQ before switching to bus state 1 and sending the TPC. In SD/MMC mode, this signal means wait for MMC ready before sending command. (MMC is busy when databit0 is low.)
20	**WAIT_FOR_CMD**	RW	0x0	Wait for Data Done (SD/MMC Mode). 0: Send commands immediately after they are written. 1: Wait to send command until after the CRC-checking phase of a data transfer has completed successfully. This delays sending a command until a block of data is transferred. This can be used to send a stop command during an SD/MMC multi-block read.
19	**LONG_RESP**	RW	0x0	Get Long Response (SD/MMC Mode). 0: The card response will be short. 1: The card will provide a 136-bit response. Only valid if GET_RESP is set. A long response cannot be checked using CHECK_RESP.
18	**CHECK_RESP**	RW	0x0	Check Response (SD/MMC, MS Modes). If this bit is set, the SSP will XOR the result with the REFERENCE field and then mask the incoming status word with the MASK field in the COMPARE register. If there is a mismatch, then the SSP will set the RESP_ERR status bit, and, if enabled, the RESP_ERR_IRQ. This should not be used with LONG_RESP.
17	**GET_RESP**	RW	0x0	Get Response (SD/MMC, MS Modes). 0: Do not wait for a response from the card. 1: This command should receive a response from the card.

Table 526. HW_SSP_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	**ENABLE**	RW	0x0	Command Transmit Enable (SD/MMC, MS Modes). 0: Commands are not enabled. 1: Data in command registers will be sent. This is normally enabled in SD/MMC or MS mode but is disabled when waiting for a MS IRQ.
15:0	**XFER_COUNT**	RW	0x1	Number of words to transfer, as referenced in WORD_LENGTH in HW_SSP_CTRL1. The run bit and DMA request will clear after this many words have been transferred. In SD/MMC or MS mode, this should be a multiple of the block size.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.2. SD/MMC and MS Command Register 0 Description

This register is the command index and control register for SD/MMC and MS modes.

HW_SSP_CMD0 0x80010010
HW_SSP_CMD0_SET 0x80010014
HW_SSP_CMD0_CLR 0x80010018
HW_SSP_CMD0_TOG 0x8001001C

Table 527. HW_SSP_CMD0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																								CMD							

Table 528. HW_SSP_CMD0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:8	**RSVD0**	RO	0x0	Reserved
7:0	**CMD**	RW	0x0	SD/MMC Command Index (uses 5:0) or MS TPC [7:0] to be sent to card. MMC_GO_IDLE_STATE = 0x00 MMC_SEND_OP_COND = 0x01 MMC_ALL_SEND_CID = 0x02 MMC_SET_RELATIVE_ADDR = 0x03 MMC_SET_DSR = 0x04 MMC_RESERVED_5 = 0x05 MMC_SWITCH = 0x06 MMC_SELECT_DESELECT_CARD = 0x07 MMC_SEND_EXT_CSD = 0x08 MMC_SEND_CSD = 0x09 MMC_SEND_CID = 0x0A MMC_READ_DAT_UNTIL_STOP = 0x0B MMC_STOP_TRANSMISSION = 0x0C MMC_SEND_STATUS = 0x0D MMC_BUSTEST_R = 0x0E MMC_GO_INACTIVE_STATE = 0x0F MMC_SET_BLOCKLEN = 0x10 MMC_READ_SINGLE_BLOCK = 0x11 MMC_READ_MULTIPLE_BLOCK = 0x12 MMC_BUSTEST_W = 0x13 MMC_WRITE_DAT_UNTIL_STOP = 0x14 MMC_SET_BLOCK_COUNT = 0x17 MMC_WRITE_BLOCK = 0x18 MMC_WRITE_MULTIPLE_BLOCK = 0x19 MMC_PROGRAM_CID = 0x1A MMC_PROGRAM_CSD = 0x1B MMC_SET_WRITE_PROT = 0x1C MMC_CLR_WRITE_PROT = 0x1D MMC_SEND_WRITE_PROT = 0x1E MMC_ERASE_GROUP_START = 0x23 MMC_ERASE_GROUP_END = 0x24 MMC_ERASE = 0x26 MMC_FAST_IO = 0x27 MMC_GO_IRQ_STATE = 0x28 MMC_LOCK_UNLOCK = 0x2A MMC_APP_CMD = 0x37 MMC_GEN_CMD = 0x38 SD_GO_IDLE_STATE = 0x00 SD_ALL_SEND_CID = 0x02 SD_SEND_RELATIVE_ADDR = 0x03 SD_SET_DSR = 0x04 SD_IO_SEND_OP_COND = 0x05 SD_SELECT_DESELECT_CARD = 0x07 SD_SEND_CSD = 0x09 SD_SEND_CID = 0x0A SD_STOP_TRANSMISSION = 0x0C SD_SEND_STATUS = 0x0D SD_GO_INACTIVE_STATE = 0x0F SD_SET_BLOCKLEN = 0x10 SD_READ_SINGLE_BLOCK = 0x11 SD_READ_MULTIPLE_BLOCK = 0x12 SD_WRITE_BLOCK = 0x18 SD_WRITE_MULTIPLE_BLOCK = 0x19 SD_PROGRAM_CSD = 0x1B SD_SET_WRITE_PROT = 0x1C SD_CLR_WRITE_PROT = 0x1D SD_SEND_WRITE_PROT = 0x1E SD_ERASE_WR_BLK_START = 0x20 SD_ERASE_WR_BLK_END = 0x21 SD_ERASE_GROUP_START = 0x23 SD_ERASE_GROUP_END = 0x24 SD_ERASE = 0x26 SD_LOCK_UNLOCK = 0x2A SD_IO_RW_DIRECT = 0x34 SD_IO_RW_EXTENDED = 0x35 SD_APP_CMD = 0x37 SD_GEN_CMD = 0x38

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.3. SD/MMC Command Register 1 Description

This register is the command argument register for SD/MMC mode.

HW_SSP_CMD1 0x80010020

Table 529. HW_SSP_CMD1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_ARG																															

Table 530. HW_SSP_CMD1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**CMD_ARG**	RW	0x0	SD/MMC or MS Command Argument. See SSP_CTRL0_DATA_XFER bit description.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.4. SD/MMC and MS Compare Reference Register Description

This register is the status response reference for MMC/SD and MS modes.

HW_SSP_COMPREF 0x80010030

Table 531. HW_SSP_COMPREF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
REFERENCE																															

Table 532. HW_SSP_COMPREF Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**REFERENCE**	RW	0x0	SD/MMC, MS Compare Mode Reference. If CHECK_RESP is set, the response will be XORed with this value. The results will be masked by the MASK bitfield. If there are any differences, then the SSP will indicate an error state to the DMA.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.5. SD/MMC and MS Compare Mask Register Description

This register is the status response mask for MMC/SD and MS modes.

HW_SSP_COMPMASK 0x80010040

Table 533. HW_SSP_COMPMASK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MASK																															

Table 534. HW_SSP_COMPMASK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**MASK**	RW	0x0	SD/MMC, MS Compare Mode Mask. If CHECK_RESP is set, the response is compared to REFERENCE, and the results are masked by this bitfield.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.6. SSP Timing Register Description

This register is used to configure the timing for the SSP.

HW_SSP_TIMING 0x80010050

Table 535. HW_SSP_TIMING

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TIMEOUT																CLOCK_DIVIDE								CLOCK_RATE							

Table 536. HW_SSP_TIMING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**TIMEOUT**	RW	0x0	Timeout counter, in serial clock cycles. This timeout is used for data transfer/write operations in SD/MMC and MS modes.

Table 536. HW_SSP_TIMING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	**CLOCK_DIVIDE**	RW	0x0	Clock Pre-Divider. CLOCK_DIVIDE must be an even value from 2 to 254.
7:0	**CLOCK_RATE**	RW	0x0	Serial Clock Rate. The value CLOCK_RATE is used to generate the transmit and receive bit rate of the SSP. The bit rate is SSPCLK / (CLOCK_DIVIDE x (1+ CLOCK_RATE)). CLOCK_RATE is a value from 0 to 255.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.7. SSP Control Register 1 Description

Control Register 1.

HW_SSP_CTRL1 0x80010060
HW_SSP_CTRL1_SET 0x80010064
HW_SSP_CTRL1_CLR 0x80010068
HW_SSP_CTRL1_TOG 0x8001006C

Table 537. HW_SSP_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SDIO_IRQ	SDIO_IRQ_EN	RESP_ERR_IRQ	RESP_ERR_IRQ_EN	RESP_TIMEOUT_IRQ	RESP_TIMEOUT_IRQ_EN	DATA_TIMEOUT_IRQ	DATA_TIMEOUT_IRQ_EN	DATA_CRC_IRQ	DATA_CRC_IRQ_EN	XMIT_IRQ	XMIT_IRQ_EN	RECV_IRQ	RECV_IRQ_EN	RECV_TIMEOUT_IRQ	RECV_TIMEOUT_IRQ_EN	RECV_OVRFLW_IRQ	RECV_OVRFLW_IRQ_EN	DMA_ENABLE	LOOPBACK	SLAVE_OUT_DISABLE	PHASE	POLARITY	SLAVE_MODE	WORD_LENGTH				SSP_MODE			

Table 538. HW_SSP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SDIO_IRQ**	RW	0x0	If this is set, an SDIO card interrupt has occurred and an IRQ, if enabled, has been sent to the interrupt collector (ICOLL). Write a one to the SCT Clear address to reset this interrupt request status bit.
30	**SDIO_IRQ_EN**	RW	0x0	SDIO Card Interrupt IRQ Enable. 0: SDIO card IRQs masked. 1: SDIO card IRQs will be sent to the ICOLL.

Table 538. HW_SSP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29	**RESP_ERR_IRQ**	RW	0x0	When the CHECK_RESP bit in CTRL0 is set, if an unexpected response is received from the card, this bit will be set. Write a one to the SCT Clear address to reset this interrupt request status bit. Note that the SSP block must be reset after a CRC error or timeout IRQ.
28	**RESP_ERR_IRQ_EN**	RW	0x0	SD/MMC Card Error IRQ Enable. 0: Card Error IRQ is Masked. 1: Card Error IRQ is enabled. When set to 1, if an SD/MMC card indicates a card error (bit is set in both the SD/MMC Error Mask and R1 Card Status response), then a CPU IRQ will be asserted.
27	**RESP_TIMEOUT_IRQ**	RW	0x0	If this is set, a command response timeout has occurred, and an IRQ, if enabled, has been sent to the IRQ Collector. Write a one to the SCT Clear address to reset this interrupt request status bit. Note that the SSP block must be reset after a CRC error or timeout IRQ.
26	**RESP_TIMEOUT_IRQ_EN**	RW	0x0	SD/MMC, MS Card Command Respone Timeout Error IRQ Enable. 0: Response Timeout IRQ is Masked. 1: Response Timeout IRQ is enabled. When set to 1, if an SD/MMC card does not respond to a command within 64 cycles or a MS card does not return ready during the handshake bus state within 32 cycles, then this CPU IRQ will be asserted.
25	**DATA_TIMEOUT_IRQ**	RW	0x0	Data Transmit/Receive Timeout Error IRQ. If the timeout counter expires before the DAT bus is ready for write or sends read data, then a data timeout has occurred. Only Valid For SD/MMC and MS Modes. Write a one to the SCT Clear address to reset this interrupt request status bit. Note that the SSP block must be reset after a CRC error or timeout IRQ.
24	**DATA_TIMEOUT_IRQ_EN**	RW	0x0	Data Transmit/Receive Timeout Error IRQ Enable. If the timeout counter expires before the DAT bus is ready for write or sends read data, then a data timeout has occurred. Only valid for SD/MMC and MS modes.
23	**DATA_CRC_IRQ**	RW	0x0	Data Transmit/Receive CRC Error IRQ. Only valid for SD/MMC and MS modes. Write a one to the SCT Clear address to reset this interrupt request status bit. Note that the SSP block must be reset after a CRC error or timeout IRQ.
22	**DATA_CRC_IRQ_EN**	RW	0x0	Data Transmit/Receive CRC Error IRQ Enable. Only valid for SD/MMC and MS modes.
21	**XMIT_IRQ**	RW	0x1	Transmit FIFO half empty or less IRQ. If enabled and the transmit FIFO is half empty or lesss, an IRQ will be generated. Write a one to the SCT Clear address to reset this interrupt request status bit.
20	**XMIT_IRQ_EN**	RW	0x0	Transmit FIFO half empty or less IRQ Enable. If set and the transmit FIFO is half empty or lesss, an IRQ will be generated. Not for use with DMA.

Table 538. HW_SSP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19	**RECV_IRQ**	RW	0x0	Receive FIFO at least Half Filled IRQ. If enabled and the receive FIFO is half filled, an IRQ will be generated. Write a one to the SCT Clear address to reset this interrupt request status bit.
18	**RECV_IRQ_EN**	RW	0x0	Receive FIFO at least Half Filled IRQ Enable. If set and the receive FIFO is half filled, an IRQ will be generated. This should not be used with the DMA.
17	**RECV_TIMEOUT_IRQ**	RW	0x0	Data Timeout Interrupt. If enabled and the Receive FIFO is not empty, an IRQ will be generated if 128 HCLK Cycles Pass before the Data register is read. Write a one to the SCT Clear address to reset this interrupt request status bit.
16	**RECV_TIMEOUT_IRQ_EN**	RW	0x0	Receive Timeout. If set and the Receive FIFO is not empty, an IRQ will be generated if 128 HCLK cycles pass before the Data register is read.
15	**RECV_OVRFLW_IRQ**	RW	0x0	Receiver Overflow Interrupt. Indicates that the receive FIFO has been written to while full. Write a one to the SCT Clear address to reset this interrupt request status bit.
14	**RECV_OVRFLW_IRQ_EN**	RW	0x0	Receiver Overflow Interrupt Enable. If set, an IRQ will be generated if the receive FIFO is written to while full.
13	**DMA_ENABLE**	RW	0x0	DMA Enable. This signal enables DMA request and DMA Command End signals to be asserted.
12	**LOOPBACK**	RW	0x0	Loopback mode is not supported. Clear this bit to 0.
11	**SLAVE_OUT_DISABLE**	RW	0x0	Slave Output Disable. 0: SSP can drive MISO in Slave Mode. 1: SSP does not drive MISO in slave mode.
10	**PHASE**	RW	0x0	Serial Clock Phase. For SPI mode only.
9	**POLARITY**	RW	0x0	Serial Clock Polarity. For SPI and SD modes only. In SD mode, 0: Command and transmit data change after rising edge of SCK, 1: Command and transmit data change after falling edge of SCK.
8	**SLAVE_MODE**	RW	0x0	Slave Mode. 0: SSP is in master mode. 1: SSP is in slave mode. Set to one for SD/MMC and MS modes.

Table 538. HW_SSP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:4	**WORD_LENGTH**	RW	0x8	Word Length in bits per word. 0x0 to 0x2 are reserved and undefined. 0x3 is 4 bits per word...0xF is 16 bits per word. Always use 8 bits per word in SD/MMC and MS modes. RESERVED0 = 0x0 0x0 is reserved and undefined RESERVED1 = 0x1 0x1 is reserved and undefined RESERVED2 = 0x2 0x2 is reserved and undefined FOUR_BITS = 0x3 use 4 bits per word EIGHT_BITS = 0x7 use 8 bits per word SIXTEEN_BITS = 0xF use 16 bits per word
3:0	**SSP_MODE**	RW	0x0	Operating Mode. 0x0 = Motorola SPI Mode, 0x1 = TI Syncronous Serial Interface mode, 0x2 = National Microwire mode, 0x3 = SD/MMC Card mode, 0x4 = MS mode. Before changing SSP_MODE, a soft reset must be issued to clear the FIFOs. SPI = 0x0 Motorola SPI mode SSI = 0x1 Texas Instruments SSI mode MICROWIRE = 0x2 National Semiconductor Microwire mode SD_MMC = 0x3 SD/MMC mode MS = 0x4 MS mode

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.8. SSP Data Register Description

HW_SSP_DATA 0x80010070

Table 539. HW_SSP_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 540. HW_SSP_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x0	Data Register. Holds one, two, three, or four words, depending on the WORD_LENGTH. If WORD_LENGTH is not 8, 16, or 32, the words are padded to those lengths. Data is right-justified. When the run bit is set, reads will cause the receive FIFO read pointer to increment and writes will cause the transmit FIFO write pointer to increment. Byte-writes are supported only for the least significant byte. Half-word (16-bit) writes are supported only for the lower half-word.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.9. SD/MMC Card Response Register 0 Description

HW_SSP_SDRESP0 0x80010080

Table 541. HW_SSP_SDRESP0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RESP0																															

Table 542. HW_SSP_SDRESP0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RESP0**	RO	0x0	SD/MMC Response Status Bits[31:0]. In MS mode, device read data can be placed here. See SSP_CTRL0_DATA_XFER bit description.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.10. SD/MMC Card Response Register 1 Description

HW_SSP_SDRESP1 0x80010090

Table 543. HW_SSP_SDRESP1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RESP1																															

Table 544. HW_SSP_SDRESP1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RESP1**	RO	0x0	SD/MMC Short Response Command Index [37:32] or Long Response [63:32]

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.11. SD/MMC Card Response Register 2 Description

HW_SSP_SDRESP2 0x800100A0

Table 545. HW_SSP_SDRESP2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RESP2																															

Table 546. HW_SSP_SDRESP2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RESP2**	RO	0x0	SD/MMC Long Response [95:64]

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.12. SD/MMC Card Response Register 3 Description

HW_SSP_SDRESP3 0x800100B0

Table 547. HW_SSP_SDRESP3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RESP3																															

Table 548. HW_SSP_SDRESP3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**RESP3**	RO	0x0	SD/MMC Long Response [127:96]

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.13. SSP Status Register Description

SSP Read-Only Status Register.

HW_SSP_STATUS 0x800100C0

Table 549. HW_SSP_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PRESENT	MS_PRESENT	SD_PRESENT	CARD_DETECT	RECV_COUNT				XMIT_COUNT				DMAREQ	DMAEND	SDIO_IRQ	RESP_CRC_ERR	RESP_ERR	RESP_TIMEOUT	DATA_CRC_ERR	TIMEOUT	RECV_TIMEOUT_STAT	RECV_DATA_STAT	RECV_OVRFLW	RECV_FULL	RECV_NOT_EMPTY	XMIT_NOT_FULL	XMIT_EMPTY	XMIT_UNDRFLW	CMD_BUSY	DATA_BUSY	DATA_XFER	BUSY

Table 550. HW_SSP_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**PRESENT**	RO	0x1	SSP Present Bit. 0: SSP is not present in this product. 1: SSP is present.
30	**MS_PRESENT**	RO	0x1	MS Controller Present. 0: MS controller is not present in this product. 1: MS controller is present.
29	**SD_PRESENT**	RO	0x1	SD/MMC Controller Present. 0: SD/MMC controller is not present in this product. 1: SD/MMC controller is present.
28	**CARD_DETECT**	RO	0x0	Reflects the state of the SSP_DETECT input pin.
27:24	**RECV_COUNT**	RO	0x0	Receive FIFO Count. Number of bytes of valid data in receive FIFO. When zero, use the full bit to differentiate between full and empty.
23:20	**XMIT_COUNT**	RO	0x0	Transmit FIFO Count. Number of bytes of valid data in transmit FIFO. When zero, use the full bit to differentiate between full and empty.
19	**DMAREQ**	RO	0x0	Reflects the state of the ssp_dmareq output port. This is a toggle signal.
18	**DMAEND**	RO	0x0	Reflects the state of the ssp_dmaend output port. This is a toggle signal.
17	**SDIO_IRQ**	RO	0x0	SDIO IRQ has been detected.
16	**RESP_CRC_ERR**	RO	0x0	SD/MMC, MS Response failed CRC check.
15	**RESP_ERR**	RO	0x0	SD/MMC, MS Card Responded to Command with an Error Condition.
14	**RESP_TIMEOUT**	RO	0x0	SD/MMC, MS Card Expected Command Response not received within 64 CLK cycles (16 for MS). This indicates a card error, bad command, or command that failed CRC check.
13	**DATA_CRC_ERR**	RO	0x0	Data CRC Error
12	**TIMEOUT**	RO	0x0	In SD/MMC mode, the timeout counter expired before data bus was ready. In MS mode, the timeout expired waiting for interrupt from card.

Table 550. HW_SSP_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11	**RECV_TIMEOUT_STAT**	RO	0x0	Raw Receive Timeout Status. Indicates that no read has occurred to non-empty receive data FIFO for 128 cycles
10	**RECV_DATA_STAT**	RO	0x0	Raw Receive FIFO at least Half Filled Interrupt Status.
9	**RECV_OVRFLW**	RO	0x0	Raw Receiver Overflow Interrupt.
8	**RECV_FULL**	RO	0x0	Receive FIFO Full
7	**RECV_NOT_EMPTY**	RO	0x0	Receive FIFO Not Empty
6	**XMIT_NOT_FULL**	RO	0x1	Transmit FIFO Not Full
5	**XMIT_EMPTY**	RO	0x1	Transmit FIFO Empty.
4	**XMIT_UNDRFLW**	RO	0x0	Transmit underflow has occurred.
3	**CMD_BUSY**	RO	0x0	SD/MMC or MS command controller is busy sending a command or receiving a response.
2	**DATA_BUSY**	RO	0x0	Card indicates data line is busy (SD/MMC or MS)
1	**DATA_XFER**	RO	0x1	Data Transfer Active
0	**BUSY**	RO	0x0	SSP is busy.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

15.11.14.SSP Debug Register Description

This register is a read-only debug register for the SSP.

HW_SSP_DEBUG 0x80010100

Table 551. HW_SSP_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATACRC_ERR				DATA_STALL	DAT_SM			MSTK_SM				CMD_OE	CMD_SM			CLK_OE	MMC_SM			DAT0_OE	DAT321_OE	SSP_CMD	SSP_RESP	SSP_TXD				SSP_RXD			

Table 552. HW_SSP_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**DATACRC_ERR**	RO	0x0	Data CRC error
27	**DATA_STALL**	RO	0x0	MS/MMC mode: FIFO transfer not ready.

Table 552. HW_SSP_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26:24	**DAT_SM**	RO	0x0	MS/MMC data transfer state machine DSM_IDLE = 0x0 DSM_START = 0x1 DSM_WORD = 0x2 DSM_CRC1 = 0x3 DSM_CRC2 = 0x4 DSM_END = 0x5 DSM_RXDLY = 0x6
23:20	**MSTK_SM**	RO	0x0	MS state machine MSTK_IDLE = 0x0 MSTK_CKON = 0x1 MSTK_BS1 = 0x2 MSTK_TPC = 0x3 MSTK_BS2 = 0x4 MSTK_HDSHK = 0x5 MSTK_BS3 = 0x6 MSTK_RW = 0x7 MSTK_CRC1 = 0x8 MSTK_CRC2 = 0x9 MSTK_BS0 = 0xA MSTK_DONE = 0xB
19	**CMD_OE**	RO	0x0	Enable for SSP_CMD
18:16	**CMD_SM**	RO	0x0	MMC command_state machine CSM_IDLE = 0x0 CSM_INDEX = 0x1 CSM_ARG = 0x2 CSM_CRC = 0x3
15	**CLK_OE**	RO	0x0	Enable for SSP_CLKOUT
14:12	**MMC_SM**	RO	0x0	MMC_state machine MMC_IDLE = 0x0 MMC_CMD = 0x1 MMC_TRC = 0x2 MMC_RESP = 0x3 MMC_RPRX = 0x4 MMC_TX = 0x5 MMC_CTOK = 0x6 MMC_RX = 0x7
11	**DAT0_OE**	RO	0x0	Enable for SSP_TXD0
10	**DAT321_OE**	RO	0x0	Enable for SSP_TXD321
9	**SSP_CMD**	RO	0x0	SSP_CMD
8	**SSP_RESP**	RO	0x0	SSP_RESP
7:4	**SSP_TXD**	RO	0x0	SSP_TXD
3:0	**SSP_RXD**	RO	0x0	SSP_RXD

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

SSP XML Revision: 1.38

16. LCD INTERFACE (LCDIF)

This chapter describes the LCD interface included on the STMP36xx and includes operation examples. Programmable registers are described in Section 16.5.

16.1. Overview

Many products based on the STMP36xx include an LCD panel with an integrated controller/driver. These smart LCDs are available in a range of sizes and capabilities, from simple text-only displays to QVGA, 16bpp color TFT panels. The integrated controllers include a frame buffer and logic to generate the appropriate LCD waveforms, including any frame rate modulation for STN displays. Smart displays have an asynchronous parallel interface that is used for setup and to write to the frame buffer. In general, it is not necessary to read data from the LCD controller.

The LCD interface provided on the STMP36xx is shown in Figure 72.

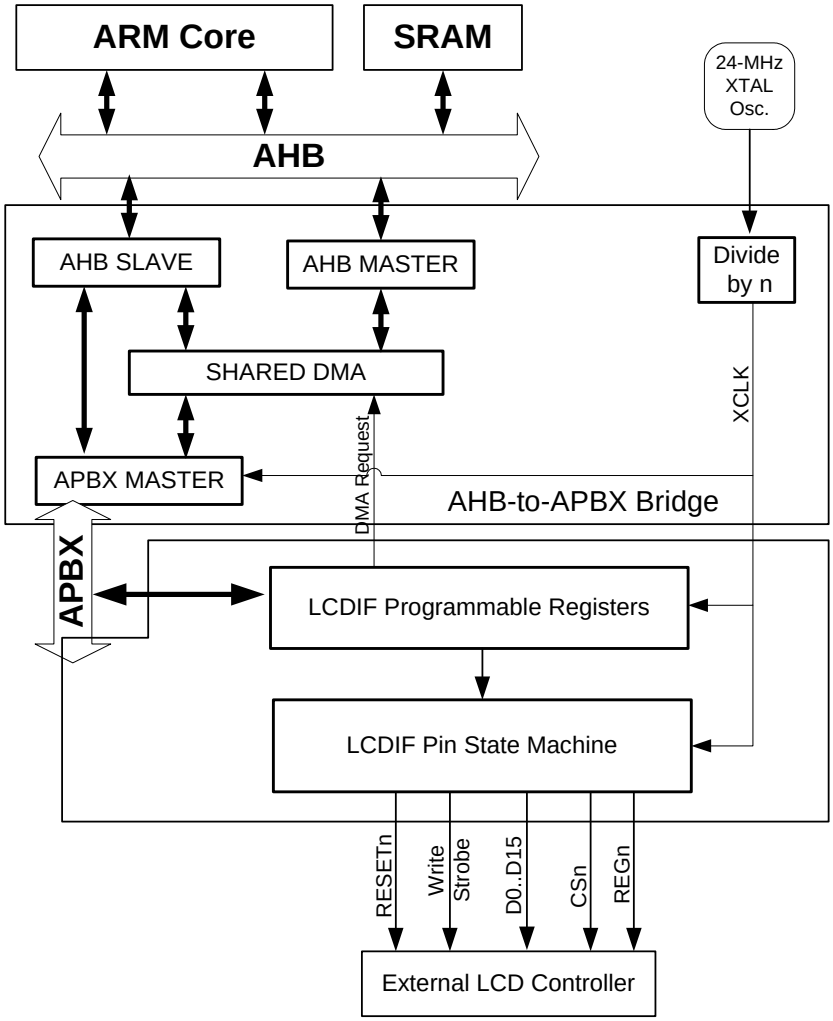


Figure 72. LCD Interface Block Diagram

The LCD interface on the STMP36xx has several major features:

- DMA data transfers allow minimal CPU overhead.
- 8 and 16-bit data bus widths are supported.
- Programmable timing supports a wide range of controllers.

The LCDIF provides an efficient mechanism to control an external smart LCD with an integrated controller. It resides on the APBX bus. It has memory-mapped control and data registers. The AHB-APBX bridge DMA can be used to move data from a memory-mapped frame buffer to the LCD's internal frame buffer. The CPU can directly send commands or data by setting up a non-DMA transfer and writing directly to the data register.

Since the LCDIF uses the AHB-APBX bridge DMA, the software designer can take advantage of the DMA's linked descriptors. This enables substantial flexibility for setting up frame buffers. It can be configured for 8 or 16 bit transfers. In 8-bit mode, pins corresponding to the upper bits will output 0, unless they have been configured to connect to another peripheral (such as GPIO).

The LCDIF receives little-endian input, but can transform the output through the use of the DATA_SWIZZLE field in the HW_LCDIF_CTRL register. The data swizzle manipulates the out-going data based on the following values:

- 00 (0): No swizzle (little-endian)
- 01 (1): Swap bytes zero and three, swap bytes 1 and 2 (big-endian)
- 10 (2): Swap half words
- 11 (3): Swap bytes within each half-word

The data register (HW_LCDIF_DATA) uses the bus byte enables, which are used to determine how many valid bytes are in each written word. For example, if the entire 32-bit word is valid, the LCDIF transmits two (16-bit mode) or four (8-bit mode) LCD data operations. The DMA sends full 32-bit words as much as possible. The last word will be shorter if the DMA transfer length is not a multiple of four bytes. If system software writes to the data register directly, care should be taken to ensure that the correct byte enables are asserted (by using the appropriate store command). Otherwise, the LCDIF may send invalid cycles to the display.

The LCDIF provides a request signal to the central DMA; see Section 11.2. "APBX DMA" on page 258. The request signal is asserted any time the LCDIF is enabled and its data FIFO has space for more data. The DMA request signal is also visible in the LCDIF control and status register. Software writing directly to the LCDIF data register (as opposed to using DMA) should monitor the FIFO_STATUS bit and only write new data when the FIFO is not full.

The LCDIF has a control output line that can be used to select which register is being written to in the LCD's controller. This register-select line is typically used to switch between command and data modes.

16.2. LCD Interface Operation Example

If using DMA, see Chapter 11 for more information on using the DMA engine.

The typical usage of the LCDIF block in software mode (also called soft DMA) would flow similarly to that outlined in Section 16.2.1 and Section 16.2.2.

16.2.1. Initialization Steps

1. To set the output pins for driving the LCD panel, set the appropriate bits in the HW_PINCTRL_MUXSELx registers in the PINCTRL block. (See Chapter 17, "Pin Control and GPIO" on page 429.)
2. Bring the LCDIF out of soft reset and clock gate.
3. Reset the LCD controller by setting LCDIF_CTRL_RESET bit appropriately, being careful to observe the reset requirements of the controller.
4. Set the BUSY_ENABLE bit in the HW_LCDIF_CTRL register if connected to an LCD controller that implements a busy line. Otherwise, the busy line is ignored.
5. Set the timings in the HW_LCDIF_TIMING register appropriately. CMD_HOLD is the hold time in cycles for the DCn signal. CMD_SETUP is the setup time in cycles for the DCn signal. DATA_HOLD is the hold time in cycles for the WEn signal. DATA_SETUP is the setup time in cycles for the WEn signal. Also note that all four of these fields must be non-zero. The LCDIF does not function if any of these signals are set to zero.
6. Set the DATA_SWIZZLE according to the endianness of the LCD controller.
7. Set the MODE86 register based on whether the data strobe should be active high (1) or low (0).
8. Set the DATA_SELECT register based on whether the data to be sent is in command mode (0) or data mode (1). Note that the idle state for the DCn signal is high, regardless of the programming of the DATA_SELECT register.
9. Set the WORD_LENGTH field appropriately—0 = 16-bit bus, 1 = 8-bit bus.

16.2.2. Run Time Steps

1. Set the COUNT register with the amount of data transfer units to send. The transfer unit size is based on WORD_LENGTH, so programming 100 into COUNT with a WORD_LENGTH of 0 will send 100 half-words.
2. When the above setup is completed and software is ready to send data, the RUN bit is set to 1. The LCDIF is now ready to receive data through writes to the HW_LCDIF_DATA register. Note that, while in soft DMA mode, the software will need to poll the FIFO_STATUS bit to ensure that it does not overflow the LCDIF's data buffers.

The FIFO_STATUS bit indicates the FIFO full state when it is 0. Therefore, when the FIFO_STATUS bit is 1, the data register can be safely written with a word, half-word, or byte as required. Writing to the data register when the FIFO_STATUS bit is 0 will result in incorrect operation. The RUN bit is cleared automatically when the LCDIF has received all the data and completed the transfer to the panel. The current transfer can be canceled (or aborted) if the RUN bit is manually set to 0. If the RUN bit is set to 0 during the middle of a transfer, the LCDIF transmits out all data it has received to that point—that is, it will flush the FIFO. In this case, the transfer count is not satisfied, but the transfer will finish normally, and the LCDIF will return to the idle state and be ready to be programmed for the next transfer.

16.3. LCDIF Pin Timing Diagrams

The LCDIF has flexible pin and strobe timings, shown in Figure 73, which enable it to optimally support a wide range of LCDs.

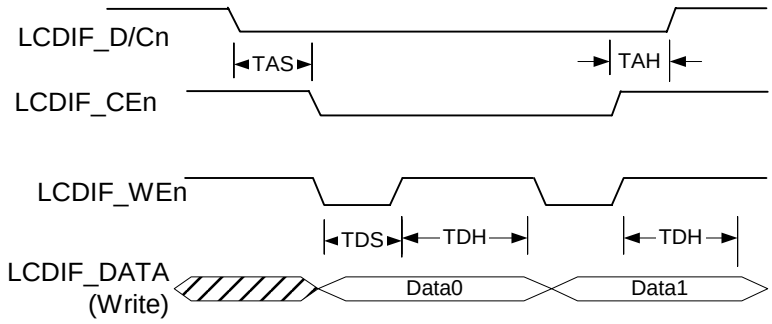


Figure 73. LCDIF Timing (Command Cycle)

The LCDIF has four basic timing parameters: Setup and Hold for the Command/Data register selection (TCS, TCH) and Setup and Hold for the Data bus (TDS, TDH). These parameters are expressed in XCLK cycles. The LCDIF data strobe polarity can be selected to active high (6800 mode) or active low (8080 mode). In 8080 mode, new data is written at the falling edge of the write strobe, WEn. WEn is asserted low for TDS XCLK clock cycles. The LCD controller latches the data with the rising edge of WEn. WEn remains high for TDH XCLK cycles. The CEn signal also remains asserted (low) for at least TDH XCLK cycles. The Data/Command register select signal is asserted TAS XCLK cycles before CEn and remains asserted for TAH cycles after CEn is no longer active.

The minimum cycle time is two XCLK cycles (TDS=TDH=1). This results in a maximum LCD data rate of 12MB/sec when XCLK is 24 MHz. TDS and TDH are 8-bit values so minimum LCDIF rate is at 510 XCLK cycles (47 kHz with a 24-MHz XCLK).

The timings are not automatically adjusted if the XCLK frequency changes, so it may be necessary to adjust the timings if XCLK changes.

16.4. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

16.5. Programmable Registers

The LCD interface block contains the following directly programmable registers.

16.5.1. LCD Interface Control and Status Register Description

The LCD Interface Control and Status Register provides overall control of the LCD parallel interface.

HW_LCDIF_CTRL 0x80060000
HW_LCDIF_CTRL_SET 0x80060004
HW_LCDIF_CTRL_CLR 0x80060008
HW_LCDIF_CTRL_TOG 0x8006000C

Table 553. HW_LCDIF_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	PRESENT	RSRVD1			BUSY_ENABLE	FIFO_STATUS	DMA_REQ	DATA_SWIZZLE		RESET	MODE86	DATA_SELECT	WORD_LENGTH	RUN	COUNT															

Table 554. HW_LCDIF_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	This bit must be set to zero to enable normal operation of the LCDIF. When set to one, it forces a block-level reset.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29	**PRESENT**	RO	0x1	0: LCDIF not present in this product 1: LCDIF is present.
28:26	**RSRVD1**	RO	0x0	Reserved bits, write as 0.
25	**BUSY_ENABLE**	RW	0x0	This bit enables the use of the interface's busy signal input. This should be enabled for LCD controllers that implement a busy line (to stall the LCDIF from sending more data until ready). Otherwise this bit should be cleared. BUSY_DISABLED = 0x0 The busy signal from the LCD controller will be ignored. BUSY_ENABLED = 0x1 Enable the use of the busy signal from the LCD controller.
24	**FIFO_STATUS**	RO	0x0	LCDIF Data FIFO Status. This bit indicates whether the FIFO is full or not full. FIFO_FULL = 0x0 LCDIF data FIFO is full, and LCDIF Data Register must not be written. FIFO_OK = 0x1 OK to write the LCDIF Data Register.
23	**DMA_REQ**	RO	0x0	Reflects the current state of the DMA Request line for the LCDIF. The DMA Request line toggles for each new request.

Table 554. HW_LCDIF_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22:21	**DATA_SWIZZLE**	RW	0x0	This field specifies how to swap the bytes in the HW_LCDIF_DATA register before transmitting them over the LCD interface bus. The data is always transmitted with the least significant byte/hword (half-word) first after the swizzle takes place. The swizzle function is independent of the WORD_LENGTH bit. See the explanation of the HW_LCDIF_DATA for names and definitions of data register fields. The supported swizzle configurations are: NO_SWAP = 0x0 No byte swapping.(Little endian) LITTLE_ENDIAN = 0x0 Little Endian byte ordering (same as NO_SWAP). BIG_ENDIAN_SWAP = 0x1 Big Endian swap (swap bytes 0,3 and 1,2). SWAP_ALL_BYTES = 0x1 Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). HWD_SWAP = 0x2 Swap half-words. HWD_BYTE_SWAP = 0x3 Swap bytes within each half-word.
20	**RESET**	RW	0x0	Reset bit for the external LCD controller. This bit can be changed at any time. LCDRESET_LOW = 0x0 LCD_RESET output signal is low. LCDRESET_HIGH = 0x1 LCD_RESET output signal is high.
19	**MODE86**	RW	0x0	Data Strobe Polarity. This bit should only be changed when RUN = 0. 8080_MODE = 0x0 Data Strobe is active low. 6800_MODE = 0x1 Data Strobe is active high.
18	**DATA_SELECT**	RW	0x0	Command Mode Polarity. This bit should only be changed when RUN = 0. CMD_MODE = 0x0 Command Mode. DCn signal is Low. DATA_MODE = 0x1 Data Mode. DCn signal is High.
17	**WORD_LENGTH**	RW	0x0	Data bus transfer width. 16_BIT = 0x0 16-bit data bus mode. 8_BIT = 0x1 8-bit data bus mode.
16	**RUN**	RW	0x0	When this bit is set by software, the LCDIF will send COUNT words (whether 8 or 16 bits) of data as data is written to the Data Register. The RUN bit is cleared by hardware only after COUNT words have been written to the Data Register.
15:0	**COUNT**	RW	0x0	This field tells the LCDIF how much data will be sent for this frame, or transaction. Count refers to the number of words of data. The word size is specified in the WORD_LENGTH field (8 or 16 bit words).

DESCRIPTION:

The LCD Interface Control and Status Register provides a variety of control and status functions to the programmer. These functions allow the interface to be very flexible to work with a variety of LCD controllers, and to minimize overhead and increase performance of LCD programming.

EXAMPLE:

```
Empty Example.
```

16.5.2. LCD Interface Timing Register Description

The LCD Interface Timing Register controls the various setup and hold times enforced by the LCD interface.

HW_LCDIF_TIMING 0x80060010

Table 555. HW_LCDIF_TIMING

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CMD_HOLD								CMD_SETUP								DATA_HOLD								DATA_SETUP							

Table 556. HW_LCDIF_TIMING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**CMD_HOLD**	RW	0x00	Number of XCLK cycles that the DCn signal is active after CEn is deasserted.
23:16	**CMD_SETUP**	RW	0x00	Number of XCLK cycles that the the DCn signal is active before CEn is asserted.
15:8	**DATA_HOLD**	RW	0x00	Data bus hold time in XCLK cycles. Also the time that the data strobe is deasserted in a cycle
7:0	**DATA_SETUP**	RW	0x00	Data bus setup time in XCLK cycles. Also the time that the data strobe is asserted in a cycle.

DESCRIPTION:

The values used in this register are dependent on the particular LCD controller used; consult the user's manual for the particular controller for required timings. Each field of the register must be non-zero, therefore the minimum value is: 0x01010101. NOTE: The timings are not automatically adjusted if the XCLK frequency changes; it may be necessary to adjust the timings if XCLK changes. NOTE: Each field in this register must be non-zero or the LCD interface will not function.

EXAMPLE:

```
Empty Example.
```

16.5.3. LCD Interface Data Register Description

The data sent to an external LCD controller is written to this register. Data can be written to this register (from the processor's perspective) as bytes, half-words (16 bits), or words (32 bits), as appropriate.

HW_LCDIF_DATA 0x80060020

Table 557. HW_LCDIF_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA_THREE								DATA_TWO								DATA_ONE								DATA_ZERO							

Table 558. HW_LCDIF_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**DATA_THREE**	RW	0x00	In 16-bit mode, this field contains the higher order byte of a 16-bit transfer. In 8-bit mode, this byte, if written, is sent last.
23:16	**DATA_TWO**	RW	0x00	In 16-bit mode, this field contains the lower order byte of a 16-bit transfer. In 8-bit mode, this byte, if written, is sent third.
15:8	**DATA_ONE**	RW	0x00	In 16-bit mode, this field contains the higher order byte of a 16-bit transfer. In 8-bit mode, this byte, if written, is sent second.
7:0	**DATA_ZERO**	RW	0x00	In 16-bit mode, this field contains the lower order byte of a 16-bit transfer. In 8-bit mode, this byte, if written, is sent first.

DESCRIPTION:

In 16-bit mode, either one or two 16-bit data words can be written to the Data register at once. In 8-bit transfer mode, one, two or four bytes can be written to this register.

EXAMPLE:

```
Empty Example.
```

16.5.4. LCD Interface Debug Register Description

The LCD Interface Debug Register provides a diagnostic view of the state machine and other useful internal signals.

HW_LCDIF_DEBUG 0x80060030

Table 559. HW_LCDIF_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1				BUSY	LAST_SUBWORD	SUBWORD_POSITION		EMPTY_WORD	STATE							DATA_COUNT															

Table 560. HW_LCDIF_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	**RSRVD1**	RO	0x0	Reserved bits. Write as 0.
27	**BUSY**	RO	0x00	Read-only view of the input busy signal from the external LCD controller.
26	**LAST_SUBWORD**	RO	0x00	Read-only view of signal indicating last sub-word in current word is being transmitted.
25:24	**SUBWORD_POSITION**	RO	0x00	Read-only view of pointer to the current sub-word being transmitted.
23	**EMPTY_WORD**	RO	0x00	Read-only view of the empty word signal.
22:16	**STATE**	RO	0x00	Read-only view of the current state machine state.
15:0	**DATA_COUNT**	RO	0x00	Read-only view of the current state of the transmit word counter.

DESCRIPTION:

The LCD Interface Debug Register is for diagnostic use only.

EXAMPLE:

```
Empty Example.
```

LCDIF XML Revision: 1.33

17. PIN CONTROL AND GPIO

This chapter describes the pin control and general-purpose input/output (GPIO) pin interface implemented on the STMP36xx. It includes sections on pin multiplexing and drive strength selection, followed by a description of the GPIO interface operation. Figure 74 and Figure 75, along with Table 561, Table 562, Table 563, and Table 564, illustrate the pin multiplexing plan. Programmable registers are described in Section 17.6.

17.1. Overview

The STMP36xx has 109 digital interface pins in the BGA package, of which 48 are available on the QFP package. (In the context of this chapter, “digital pin” means the 3.3-V standard digital interface pins. This does *not* include JTAG, TESTMODE, or digital radio interface pins.)

Each digital pin may be dynamically programmed at any time to be in one of the following states:

- High-impedance (for input, three-state, or open-drain applications)
- Low
- High
- Controlled by one of 'n' chip hardware interface blocks, where 'n' is a pin-dependent number between 1 and 3, as described in Section 17.2.

Additionally, the state of each pin may be read at any time (no matter how it is configured), and its drive strength may be configured to be 4 or 8 ma (or to be 4 or 16 ma for high current pins). Each pin may also be used as an interrupt input, and the interrupt trigger type may be configured to be low level-sensitive, high level-sensitive, rising edge-sensitive, or falling edge-sensitive.

For programming purposes, these 109 pins are divided into four banks of up to 32 pins each. The following sections show how to use all the features of each pin, and the pin register definitions are included in Section 17.6.

17.2. Pin Interface Multiplexing

The STMP36xx is somewhat pin-limited in the BGA package, and severely pin-limited in the QFP package. It contains a rich set of specialized hardware interfaces (SPI, NAND flash, NOR flash, SDRAM, etc.), but does not have enough pins to allow use of all signals of all interfaces simultaneously. Consequently, a pin multiplexing scheme (the “pin mux”) is employed to allow customers to choose which specialized interfaces to enable for their applications. In addition to these specialized hardware interfaces, the STMP36xx allows any digital pin to be used as a GPIO pin. This capability supports custom interfacing requirements, such as the ability to communicate with LEDs, digital buttons, and other devices that are not directly supported by any of the STMP36xx specialized hardware interfaces.

Each pin is connected to 1, 2, or 3 specialized hardware interfaces in addition to the GPIO block. The description of each pin in Chapter 35, “Pin Descriptions” on page 809 contains full details on which specialized hardware interfaces are attached to that pin. For example, pin PWM0 (GPIO bank 3, bit 10) is shared between the PWM, ETM, and Debug UART hardware interfaces, so care must be taken when designing a system to ensure that these functions are not required simultaneously.

Programs define which of the available hardware interfaces controls each pin by writing a two-bit field for that pin into one of the HW_PINCTRL_MUXSELx registers, as shown in Table 561 through Table 564. Pin names are shown in the last row under each register. See also Figure 74 and Figure 75.

Table 561. Relationship of MUXSELx Registers to GPIO Bits to Pin Names: Bank 0

GPIO BANK 0, PINS 15-0 (REG MUXSEL0)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
GPMI_D15		GPMI_D14		GPMI_D13		GPMI_D12		GPMI_D11		GPMI_D10		GPMI_D09		GPMI_D08		GPMI_D07		GPMI_D06		GPMI_D05		GPMI_D04		GPMI_D03		GPMI_D02		GPMI_D01		GPMI_D00	
GPIO BANK 0, PINS 31-16 (REG MUXSEL1)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	
SSP_ DATA3		SSP_ DATA2		SSP_ DATA1		SSP_ DATA0		SSP_SCK		SSP_CMD		SSP_ DETECT		GPMI_A2		GPMI_A1		GPMI_A0		GPMI_ WRN		GPMI_ RDY2		GPMI_ RDY3		GPMI_ RDY		GPMI_ RDN		GPMI_IRQ	

Table 562. Relationship of MUXSELx Registers to GPIO Bits to Pin Names: Bank 1

GPIO BANK 1, PINS 15-0 (REG MUXSEL2)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
LCD_D15		LCD_D14		LCD_D13		LCD_D12		LCD_D11		LCD_D10		LCD_D09		LCD_D08		LCD_D07		LCD_D06		LCD_D05		LCD_D04		LCD_D03		LCD_D02		LCD_D01		LCD_D00	
GPIO BANK 1, PINS 25-16 (REG MUXSEL3)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	
Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		UARTAP_ TX		UARTAP_ RX		UARTAP_ RTS		UARTAP_ CTS		LCD_ BUSY		GMPI_ RESET M		LCD_CS		LCD_WR		LCD_RS		LCD_ RESET	

Table 563. Relationship of MUXSELx Registers to GPIO Bits to Pin Names: Bank 2

GPIO BANK 2, PINS 15-0 (REG MUXSEL4)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
EMI_D15		EMI_D14		EMI_D13		EMI_D12		EMI_D11		EMI_D10		EMI_D09		EMI_D08		EMI_D07		EMI_D06		EMI_D05		EMI_D04		EMI_D03		EMI_D02		EMI_D01		EMI_D00	
GPIO BANK 2, PINS 31-16 (REG MUXSEL5)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	
EMI_RASN		EMI_A14		EMI_A13		EMI_A12		EMI_A11		EMI_A10		EMI_A09		EMI_A08		EMI_A07		EMI_A06		EMI_A05		EMI_A04		EMI_A03		EMI_A02		EMI_A01		EMI_A00	

Table 564. Relationship of MUXSELx Registers to GPIO Bits to Pin Names: Bank 3

GPIO BANK 3, PINS 15-0 (REG MUXSEL6)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
ROTARYA		PWM4		PWM3		PWM2		PWM1		PWM0		EMI_WEN		EMI_DOM1		EMI_DOM0		EMI_CASN		EMI_CKE		EMI_CLK		EMI_CE3N		SMI_CE2N		EMI_CE1N		EMI_CE0N	
GPIO BANK 3, PINS 18-16 (REG MUXSEL7)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	
Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		I2C_SDA		I2C_SCL		ROTARYB	

Bank 0

GPIO0				7	6	5	4	3	2	1	0
MUXSEL0				15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0	0	GPMI_D7	GPMI_D6	GPMI_D5	GPMI_D4	GPMI_D3	GPMI_D2	GPMI_D1	GPMI_D0
Pin Function 1	MUXSEL=	0	1								
Pin Function 2	MUXSEL=	1	0								
Pin Function 3	MUXSEL=	1	1	EREPAIR	TM2	TM1	TM0	BM3	BM2	BM1	BM0

GPIO0				15	14	13	12	11	10	9	8
MUXSEL0				31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0	0	GPMI_D15	GPMI_D14	GPMI_D13	GPMI_D12	GPMI_D11	GPMI_D10	GPMI_D9	GPMI_D8
Pin Function 1	MUXSEL=	0	1	EMI_A22	EMI_A21	EMI_A20	EMI_A19	EMI_A18	EMI_A17	EMI_A16	EMI_A15
Pin Function 2	MUXSEL=	1	0	GPMI_CE3n*	GPMI_CE2n*	GPMI_CE1n*	GPMI_CE0n*				
Pin Function 3	MUXSEL=	1	1								

GPIO0				23	22	21	20	19	18	17	16
MUXSEL1				15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0	0	GPMI_A1	GPMI_A0	GPMI_WRn	GPMI_RDY2	GPMI_RDY3	GPMI_RDY	GPMI_RDn	GPMI_IRQ
Pin Function 1	MUXSEL=	0	1	EMI_A24	EMI_A23			EMI_OEn			
Pin Function 2	MUXSEL=	1	0								
Pin Function 3	MUXSEL=	1	1								

GPIO0				31	30	29	28	27	26	25	24
MUXSEL1				31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0	0	SSP_DATA3	SSP_DATA2	SSP_DATA1	SSP_DATA0	SSP_SCK	SSP_CMD	SSP_DETECT	GPMI_A2
Pin Function 1	MUXSEL=	0	1								EMI_A25
Pin Function 2	MUXSEL=	1	0							RTCK	
Pin Function 3	MUXSEL=	1	1								

Bank 1

GPIO1				7	6	5	4	3	2	1	0
MUXSEL2				15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0	0	LCD_D7	LCD_D6	LCD_D5	LCD_D4	LCD_D3	LCD_D2	LCD_D1	LCD_D0
Pin Function 1	MUXSEL=	0	1	ETM_DA7	ETM_DA6	ETM_DA5	ETM_DA4	ETM_DA3	ETM_DA2	ETM_DA1	ETM_DA0
Pin Function 2	MUXSEL=	1	0								
Pin Function 3	MUXSEL=	1	1								

GPIO1				15	14	13	12	11	10	9	8
MUXSEL2				31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0	0	LCD_D15	LCD_D14	LCD_D13	LCD_D12	LCD_D11	LCD_D10	LCD_D9	LCD_D8
Pin Function 1	MUXSEL=	0	1	ETM_DB7	ETM_DB6	ETM_DB5	ETM_DB4	ETM_DB3	ETM_DB2	ETM_DB1	ETM_DB0
Pin Function 2	MUXSEL=	1	0	RTCK							
Pin Function 3	MUXSEL=	1	1								

GPIO1				23	22	21	20	19	18	17	16
MUXSEL3				15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0	0	UARTAPP_RTS	UARTAPP_CTS	LCD_BUSY	GPMI_RESETn	LCD_CS	LCD_WR	LCD_RS	LCD_RESET
Pin Function 1	MUXSEL=	0	1	RTCK			EMI_RESET	ETM_TCLK	ETM_PSA2	ETM_PSA0	ETM_PSA1
Pin Function 2	MUXSEL=	1	0	IR_CLK							
Pin Function 3	MUXSEL=	1	1								

GPIO1				31	30	29	28	27	26	25	24
MUXSEL3				31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0	0							UARTAPP_TX	UARTAPP_RX
Pin Function 1	MUXSEL=	0	1								
Pin Function 2	MUXSEL=	1	0							IR_TX	IR_RX
Pin Function 3	MUXSEL=	1	1								

KEY

APP UART Pins	IR Pins	TIMROT Pins	SPDIF Pin	Power-Up Pin Function (General-Purpose I/O)	GPMI Pins	EMI Pins
ETM/JTAG Pins	I^2C Pins	PWM Pins	DBG UART Pins	Unused	SSP Pins	LCD Pins

Figure 74. Pin Control Mux Chart (Banks 0 and 1)

Bank 2

GPIO2			7	6	5	4	3	2	1	0
MUXSEL4			15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0 0	EMI_D7	EMI_D6	EMI_D5	EMI_D4	EMI_D3	EMI_D2	EMI_D1	EMI_D0
Pin Function 1	MUXSEL=	0 1								
Pin Function 2	MUXSEL=	1 0								
Pin Function 3	MUXSEL=	1 1								

GPIO2			15	14	13	12	11	10	9	8
MUXSEL4			31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0 0	EMI_D15	EMI_D14	EMI_D13	EMI_D12	EMI_D11	EMI_D10	EMI_D9	EMI_D8
Pin Function 1	MUXSEL=	0 1								
Pin Function 2	MUXSEL=	1 0								
Pin Function 3	MUXSEL=	1 1								

GPIO2			23	22	21	20	19	18	17	16
MUXSEL5			15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0 0	EMI_A7	EMI_A6	EMI_A5	EMI_A4	EMI_A3	EMI_A2	EMI_A1	EMI_A0
Pin Function 1	MUXSEL=	0 1								
Pin Function 2	MUXSEL=	1 0								
Pin Function 3	MUXSEL=	1 1								

GPIO2			31	30	29	28	27	26	25	24
MUXSEL5			31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0 0	EMI_RASn	EMI_A14	EMI_A13	EMI_A12	EMI_A11	EMI_A10	EMI_A9	EMI_A08
Pin Function 1	MUXSEL=	0 1								
Pin Function 2	MUXSEL=	1 0								
Pin Function 3	MUXSEL=	1 1								

Bank 3

GPIO3			7	6	5	4	3	2	1	0
MUXSEL6			15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0 0	EMI_DQM0	EMI_CASn	EMI_CKE	EMI_CLK	EMI_CE3n	EMI_CE2n	EMI_CE1n	EMI_CE0n
Pin Function 1	MUXSEL=	0 1					GPMI_CE3n	GPMI_CE2n	GPMI_CE1n	GPMI_CE0n
Pin Function 2	MUXSEL=	1 0								
Pin Function 3	MUXSEL=	1 1								

GPIO3			15	14	13	12	11	10	9	8
MUXSEL6			31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0 0	ROTARYA	PWM4	PWM3	PWM2	PWM1	PWM0	EMI_WEn	EMI_DQM1
Pin Function 1	MUXSEL=	0 1		ETM_PSB1	ETM_PSB0	ETM_PSB2	ETM_TSYNCB	ETM_TSYNCA		
Pin Function 2	MUXSEL=	1 0			SPKR_SPDIF	RTCK	UARTDBG_TX	UARTDBG_RX		
Pin Function 3	MUXSEL=	1 1								

GPIO3			23	22	21	20	19	18	17	16
MUXSEL7			15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Pin Function 0	MUXSEL=	0 0						I2C_SDA	I2C_SCL	ROTARYB
Pin Function 1	MUXSEL=	0 1								
Pin Function 2	MUXSEL=	1 0								
Pin Function 3	MUXSEL=	1 1								

GPIO3			31	30	29	28	27	26	25	24
MUXSEL7			31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Pin Function 0	MUXSEL=	0 0								
Pin Function 1	MUXSEL=	0 1								
Pin Function 2	MUXSEL=	1 0								
Pin Function 3	MUXSEL=	1 1								

KEY

APP UART Pins	IR Pins	TIMROT Pins	SPDIF Pin	Power-Up Pin Function (General-Purpose I/O)	GPMI Pins	EMI Pins
ETM/JTAG Pins	I^2C Pins	PWM Pins	DBG UART Pins	Unused	SSP Pins	LCD Pins

Figure 75. Pin Control Mux Chart (Banks 2 and 3)

Readback registers are never affected by the operation of the HW_PINCTRL_MUXSELx registers and always sense the actual value on the data pin.

For example, if a pin is programmed to be a GPIO output and then driven high, any specialized hardware interfaces that are actively monitoring that pin will read the high logic value. Conversely, if the pin mux is programmed to give a specialized hardware interface such as the EMI block control of a particular pin, the current state of that pin can be read through its GPIO read register at any time, even while active EMI cycles are in progress.

Because the pin mux configuration is independent for each individual pin, any pin not required for a given active interface can be reused as a GPIO pin. For example, the EMI_CE0N pin can be configured and controlled as a GPIO pin, while the other EMI interface pins are controlled by the EMI block.

17.3. Pin Drive Strength Selection

Each digital pin can be programmed to drive at either 4 or 8 mA by setting the bit corresponding to that pin in one of the HW_PINCTRL_DRIVEx registers. There are two exceptions to this behavior. The PWM3 and PWM4 pins have higher drive capability and will drive at 16 mA when the 8-mA setting is selected.

17.4. GPIO Interface

The registers discussed in the following sections exist within each of these four banks to configure the chip's digital pins. Some pins only exist in the 169-pin package options. The registers that control those pins exist but perform no useful function when in a 100-pin package.

17.4.1. Output Operation

Programming and controlling a digital pin as a GPIO output is accomplished by programming the appropriate bits in four registers, as shown in Figure 76.

- After setting the field in the HW_PINCTRL_MUXSELx to program for GPIO control, the HW_PINCTRL_DRIVEx register bit is set for the desired drive strength.
- The HW_PINCTRL_DOUTx register bit is then loaded with the level that will initially be driven on the pin.
- Finally, the HW_PINCTRL_DOEx register bit is set.
- Once set, the logic value the HW_PINCTRL_DOUTx bit will be driven on the pin and the value can be toggled with repeated writes.

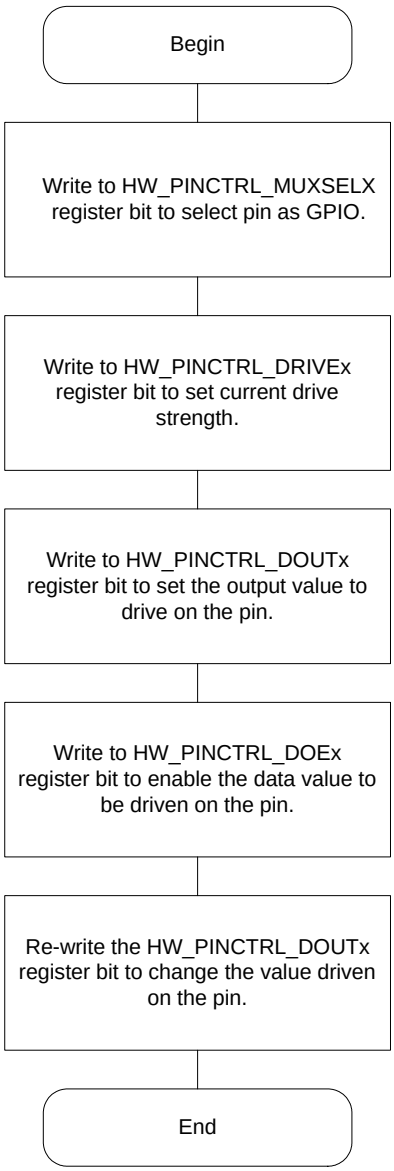


Figure 76. GPIO Output Setup Flowchart

17.4.2. Input Operation

Any digital pin may be used as a GPIO input by programming its HW_PINCTRL_MUXSELx field to 3 to enable GPIO mode, programming its HW_PINCTRL_DOEx field to 0 to disable output, and then reading from the HW_PINCTRL_DINx register, as shown in Figure 77. Note that because of clock synchronization issues, the logic levels read from the HW_PINCTRL_DINx registers are delayed from the pins by two APBX clock cycles.

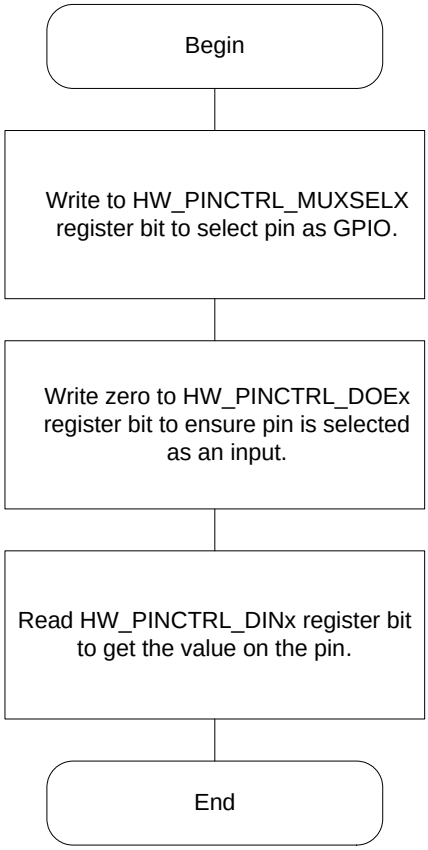


Figure 77. GPIO Input Setup Flowchart

17.4.3. Input Interrupt Operation

Programming and controlling a digital pin as a GPIO interrupt input is accomplished by programming the appropriate bits in six registers, as shown in Figure 78.

- After setting the HW_PINCTRL_MUXSELx register for GPIO, the HW_PINCTRL_IRQLEVELx and HW_PINCTRL_IRQPOLx registers set the interrupt trigger mode. A GPIO interrupt pin can be programmed in one of four trigger detect modes: positive edge, negative edge, positive level, and negative level triggered.
- The HW_PINCTRL_IRQSTATx register bit should then be cleared to ensure that there are no interrupts pending when enabled.
- Setting the HW_PINCTRL_PIN2IRQx register bit will then set up the pin to be an interrupt pin.
- At this point, if an interrupt event occurs on the pin, it will be sensed and recorded in the appropriate HW_PINCTRL_IRQSTATx bit.
- However, the interrupt will not be communicated back to the interrupt collector until the HW_PINCTRL_IRQENx register bit is enabled.

Figure 79 shows the logic diagram for the interrupt-generation circuit.

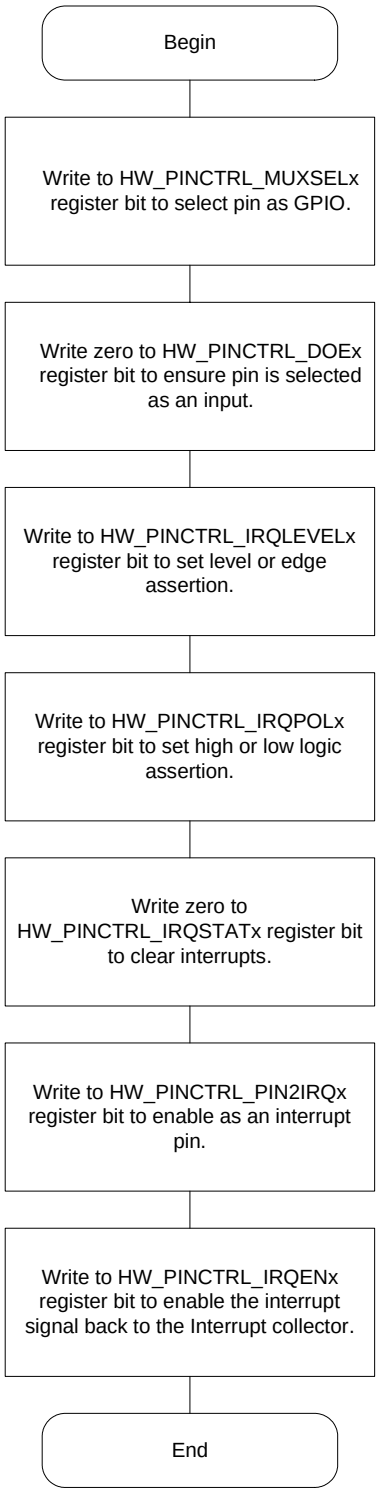


Figure 78. GPIO Interrupt Flowchart

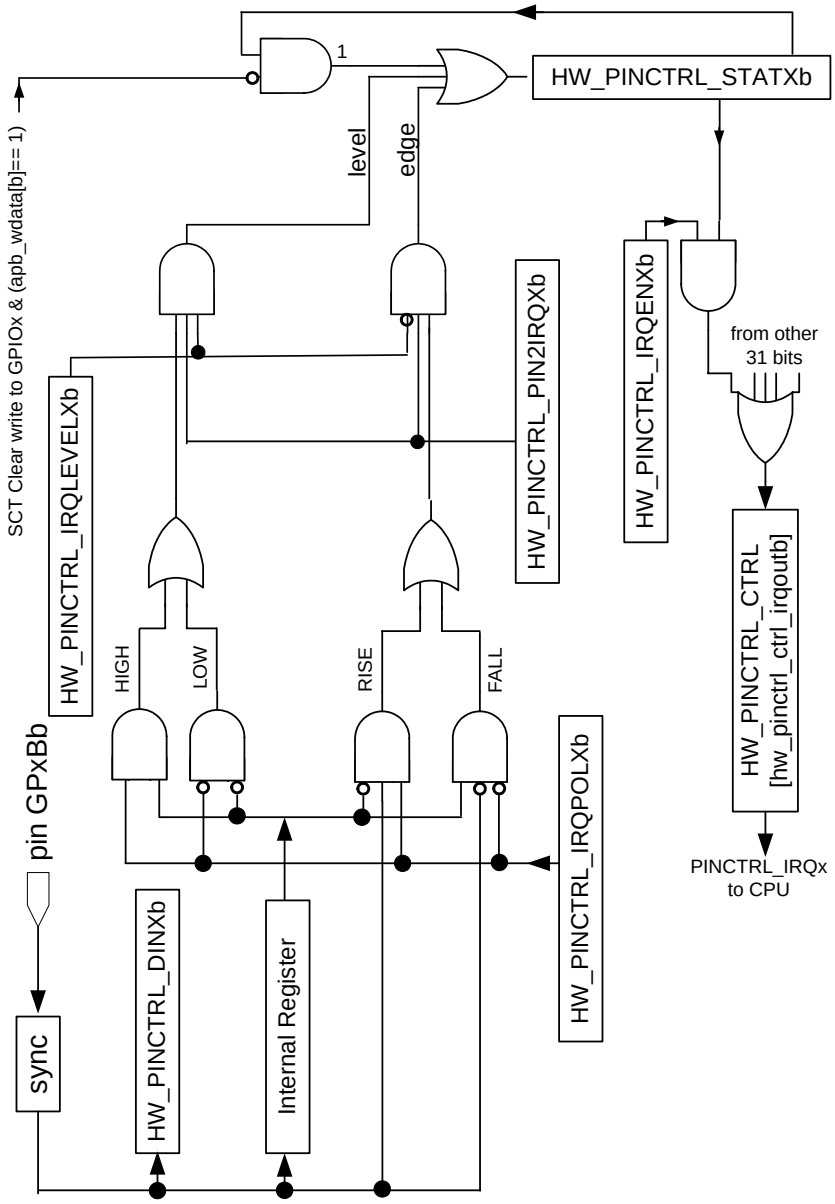


Figure 79. GPIO Interrupt Generation

17.5. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

17.6. Programmable Registers

The following programmable registers are available for controlling the pin control and GPIO interface of the STMP36xx.

17.6.1. PINCTRL Block Control Register Description

The PINCTRL Block Control Register contains the block control bits and combined interrupt output status for each PINCTRL bank.

HW_PINCTRL_CTRL 0x80018000
HW_PINCTRL_CTRL_SET 0x80018004
HW_PINCTRL_CTRL_CLR 0x80018008
HW_PINCTRL_CTRL_TOG 0x8001800C

Table 565. HW_PINCTRL_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	PRESENT3	PRESENT2	PRESENT1	PRESENT0	RSRVD1																						IRQOUT3	IRQOUT2	IRQOUT1	IRQOUT0

Table 566. HW_PINCTRL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	This bit must be set to zero to enable operation of any of the PINCTRL banks. When set to one, it forces a block-level reset.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it disables the block clock.
29	**PRESENT3**	RO	0x1	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 3 is not present in this product. 1: GPIO functionality for Bank 3 is present.
28	**PRESENT2**	RO	0x1	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 2 is not present in this product. 1: GPIO functionality for Bank 2 is present.
27	**PRESENT1**	RO	0x1	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 1 is not present in this product. 1: GPIO functionality for Bank 1 is present.
26	**PRESENT0**	RO	0x1	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 0 is not present in this product. 1: GPIO functionality for Bank 0 is present.
25:4	**RSRVD1**	RO	0x000000	Always write zeroes to this field.
3	**IRQOUT3**	RO	0x0	Read-only view of the interrupt collector GPIO3 signal, sourced from the combined IRQ outputs from bank 3.
2	**IRQOUT2**	RO	0x0	Read-only view of the interrupt collector GPIO2 signal, sourced from the combined IRQ outputs from bank 2.

Table 566. HW_PINCTRL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**IRQOUT1**	RO	0x0	Read-only view of the interrupt collector GPIO1 signal, sourced from the combined IRQ outputs from bank 1.
0	**IRQOUT0**	RO	0x0	Read-only view of the interrupt collector GPIO0 signal, sourced from the combined IRQ outputs from bank 0.

DESCRIPTION:

This register contains block-wide control bits and combined bank interrupt status bits. For normal operation, write a 0x00000000 into this register.

EXAMPLE:

```
Empty Example.
```

17.6.2. PINCTRL Bank 0 Lower Pin Mux Select Register Description

The PINCTRL Bank 0 Lower Pin Mux Select Register provides pin function selection for pins 0 through 15 of bank 0.

HW_PINCTRL_MUXSEL0 0x80018010
HW_PINCTRL_MUXSEL0_SET 0x80018014
HW_PINCTRL_MUXSEL0_CLR 0x80018018
HW_PINCTRL_MUXSEL0_TOG 0x8001801C

Table 567. HW_PINCTRL_MUXSEL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNC_SEL																															

Table 568. HW_PINCTRL_MUXSEL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FUNC_SEL**	RW	0xffffffff	This field selects which hardware interface block controls each of the first 16 pins in bank 0. This field is divided into sixteen 2-bit subfields, with bits [1:0] corresponding to pin 0, bits [3:2] corresponding to pin 1, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the first sixteen pins in bank 0. For example, if this register is set to 0x0000002C, the second pin in the bank (GPIO0[1]) will be set to GPIO mode, the third pin in the bank will be set to its second alternate function mode, and bank pins 0 and 3-15 will be set to their primary function modes.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin-to-GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.3. PINCTRL Bank 0 Upper Pin Mux Select Register Description

The PINCTRL Bank 0 Upper Pin Mux Select Register provides pin function selection for pins 16 through 31 of bank 0.

HW_PINCTRL_MUXSEL1 0x80018020
HW_PINCTRL_MUXSEL1_SET 0x80018024
HW_PINCTRL_MUXSEL1_CLR 0x80018028
HW_PINCTRL_MUXSEL1_TOG 0x8001802C

Table 569. HW_PINCTRL_MUXSEL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNC_SEL																															

Table 570. HW_PINCTRL_MUXSEL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FUNC_SEL**	RW	0xffffffff	This field selects which hardware interface block controls each of the last 16 pins in bank 0. This field is divided into sixteen 2-bit subfields, with bits [1:0] corresponding to pin 16, bits [3:2] corresponding to pin 17, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the last sixteen pins in bank 0. For example, if this register is set to 0x00000003, the sixteenth pin in the bank (GPIO0[16]) will be set to GPIO mode and bank pins 17-31 will be set to their primary function mode.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin-to-GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.4. PINCTRL Bank 0 Drive Strength Register Description

The PINCTRL Bank 0 Drive Strength Register selects the current drive strength for pins in bank 0.

HW_PINCTRL_DRIVE0 0x80018030
HW_PINCTRL_DRIVE0_SET 0x80018034
HW_PINCTRL_DRIVE0_CLR 0x80018038

HW_PINCTRL_DRIVE0_TOG 0x8001803C

Table 571. HW_PINCTRL_DRIVE0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DRIVE8MA																															

Table 572. HW_PINCTRL_DRIVE0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DRIVE8MA**	RW	0x00000000	This field selects the drive strength for pins configured as outputs. Each bit in this register corresponds to one of the 32 pins in bank 0: 0= 4-mA drive strength 1= 8-mA drive strength

DESCRIPTION:

The PINCTRL Bank 0 Drive Strength Register selects the drive strength (4 mA or 8 mA) for pins in bank 0 that are configured for output. For example, if this register is set to 0x10000004, then bank 0 pins 2 and 28 will be set to 8-mA drive strength and the rest of the pins in the bank will be set to 4-mA drive strength.

EXAMPLE:

```
Empty Example.
```

17.6.5. PINCTRL Bank 0 Data Output Register Description

The Bank 0 Data Output register provides data for all pins in bank 0 that are configured for GPIO output mode.

HW_PINCTRL_DOUT0 0x80018050
HW_PINCTRL_DOUT0_SET 0x80018054
HW_PINCTRL_DOUT0_CLR 0x80018058
HW_PINCTRL_DOUT0_TOG 0x8001805C

Table 573. HW_PINCTRL_DOUT0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATAOUT																															

Table 574. HW_PINCTRL_DOUT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATAOUT**	RW	0x00000000	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 32 pins in bank 0.

DESCRIPTION:

This register contains the data that will be driven out all bank 0 pins that are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then GPIO0[0] will

be driven with the value from bit 0 of this register, GPIO0[1] will not be driven, and GPIO0[2:15] will be controlled by the associated primary interfaces.

EXAMPLE:

```
Empty Example.
```

17.6.6. PINCTRL Bank 0 Data Input Register Description

The current value of all bank 0 pins may be read from the PINCTRL Bank 0 Data Input Register.

HW_PINCTRL_DIN0 0x80018060
HW_PINCTRL_DIN0_SET 0x80018064
HW_PINCTRL_DIN0_CLR 0x80018068
HW_PINCTRL_DIN0_TOG 0x8001806C

Table 575. HW_PINCTRL_DIN0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATAIN																															

Table 576. HW_PINCTRL_DIN0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATAIN**	RO	0x00000000	Each bit in this read-only register corresponds to one of the 32 pins in bank 0. The current state of each pin in bank 0, synchronized to HCLK, may be read here.

DESCRIPTION:

This register reflects the current values of all the bank 0 pins. The register accurately reflects the state of the pin, regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers. But generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to ensure that the chip is not driving the pin.

For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then pin GPIO0[1] will be an input pin, and bit 1 of this register will reflect its current state.

EXAMPLE:

```
Empty Example.
```

17.6.7. PINCTRL Bank 0 Output Enable Register Description

The PINCTRL Bank 0 Output Enable Register controls the output enable signal for all pins in bank 0 that are configured for GPIO mode.

HW_PINCTRL_DOE0 0x80018070
HW_PINCTRL_DOE0_SET 0x80018074
HW_PINCTRL_DOE0_CLR 0x80018078
HW_PINCTRL_DOE0_TOG 0x8001807C

Table 577. HW_PINCTRL_DOE0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATAOE																															

Table 578. HW_PINCTRL_DOE0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATAOE**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 0. Setting a bit in this register to one allows the STMP36xx to drive the corresponding pin in GPIO mode.

DESCRIPTION:

For pins in bank 0 that are configured as GPIOs (by setting the pin's control field in HW_PINCTRL_MUXSEL0/1 to 3), a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUT0 to be driven out the pin, and a 0 in this register will disable the corresponding driver. For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then pin GPIO0[0] will be driven with the value from HW_PINCTRL_DOUT0 bit 0, pin GPIO0[1] will be three-stated, and pins GPIO0[2-15] will be controlled by the default peripheral interface associated with each of those pins.

EXAMPLE:

```
Empty Example.
```

17.6.8. *PINCTRL Bank 0 Interrupt Select Register Description*

The Bank 0 Interrupt Select register selects which of the bank 0 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ0 0x80018080
HW_PINCTRL_PIN2IRQ0_SET 0x80018084
HW_PINCTRL_PIN2IRQ0_CLR 0x80018088
HW_PINCTRL_PIN2IRQ0_TOG 0x8001808C

Table 579. HW_PINCTRL_PIN2IRQ0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ENABLE2IRQ																															

Table 580. HW_PINCTRL_PIN2IRQ0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ENABLE2IRQ**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 0: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 0 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL0 and HW_PINCTRL_IRQPOL0 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT0 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN0 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO0.

For example, if this register contains 0x00000014, then pins GPIO0[2] and GPIO0[4] can be used as interrupt pins, and no other pins in bank 0 will cause bits to be set in the HW_PINCTRL_IRQSTAT0 register.

EXAMPLE:

```
Empty Example.
```

17.6.9. PINCTRL Bank 0 Interrupt Mask Register Description

The PINCTRL Bank 0 Interrupt Mask Register contains interrupt enable masks for the pins in bank 0.

HW_PINCTRL_IRQEN0 0x80018090
HW_PINCTRL_IRQEN0_SET 0x80018094
HW_PINCTRL_IRQEN0_CLR 0x80018098
HW_PINCTRL_IRQEN0_TOG 0x8001809C

Table 581. HW_PINCTRL_IRQEN0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IRQENABLE																															

Table 582. HW_PINCTRL_IRQEN0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**IRQENABLE**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 0: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 0. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT0, an interrupt will be propagated to the interrupt collector as interrupt GPIO0.

For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT0 (corresponding to pins GPIO0[2] and GPIO0[4]) will cause interrupts from bank 0.

EXAMPLE:

```
Empty Example.
```

17.6.10. PINCTRL Bank 0 Interrupt Level/Edge Register Description

The PINCTRL Bank 0 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 0.

HW_PINCTRL_IRQLEVEL0 0x800180A0
HW_PINCTRL_IRQLEVEL0_SET 0x800180A4
HW_PINCTRL_IRQLEVEL0_CLR 0x800180A8
HW_PINCTRL_IRQLEVEL0_TOG 0x800180AC

Table 583. HW_PINCTRL_IRQLEVEL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITIRQLEVEL																															

Table 584. HW_PINCTRL_IRQLEVEL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITIRQLEVEL**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 0: 1= Level detection 0= Edge detection

DESCRIPTION:

This register selects level or edge detection for interrupt generation. Each pin in bank 0 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL0 (see below) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.11. PINCTRL Bank 0 Interrupt Polarity Register Description

The PINCTRL Bank 0 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 0.

HW_PINCTRL_IRQPOL0 0x800180B0
HW_PINCTRL_IRQPOL0_SET 0x800180B4
HW_PINCTRL_IRQPOL0_CLR 0x800180B8
HW_PINCTRL_IRQPOL0_TOG 0x800180BC

Table 585. HW_PINCTRL_IRQPOL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IRQPOL																															

Table 586. HW_PINCTRL_IRQPOL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**IRQPOL**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 0: 0= Low or falling edge 1= High or rising edge

DESCRIPTION:

This register selects the polarity for interrupt generation. Each pin in bank 0 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL0 (see above) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.12. PINCTRL Bank 0 Interrupt Status Register Description

The PINCTRL Bank 0 Interrupt Status Register reflects pending interrupt status for the pins in bank 0.

HW_PINCTRL_IRQSTAT0 0x800180C0
HW_PINCTRL_IRQSTAT0_SET 0x800180C4
HW_PINCTRL_IRQSTAT0_CLR 0x800180C8
HW_PINCTRL_IRQSTAT0_TOG 0x800180CC

Table 587. HW_PINCTRL_IRQSTAT0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
STAT																															

Table 588. HW_PINCTRL_IRQSTAT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**STAT**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 0: 0= No interrupt pending 1= Interrupt pending

DESCRIPTION:

This register reflects the pending interrupt status for pins in bank 0. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 0 pin that has been enabled as an interrupts source in the HW_PINCTRL_PIN2IRQ0 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT0_CLR. Status bits for pins configured as level-sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ0.

If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN0 mask register, then the GPIO0 interrupt will be asserted to the interrupt collector.

EXAMPLE:

```
Empty Example.
```

17.6.13. PINCTRL Bank 1 Lower Pin Mux Select Register Description

The PINCTRL Bank 1 Lower Pin Mux Select Register provides pin function selection for pins 0 through 15 of bank 1.

HW_PINCTRL_MUXSEL2 0x80018110
HW_PINCTRL_MUXSEL2_SET 0x80018114
HW_PINCTRL_MUXSEL2_CLR 0x80018118
HW_PINCTRL_MUXSEL2_TOG 0x8001811C

Table 589. HW_PINCTRL_MUXSEL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNC_SEL																															

Table 590. HW_PINCTRL_MUXSEL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FUNC_SEL**	RW	0xffffffff	This field selects which hardware interface block controls each of the first 16 pins in bank 1. This field is divided into sixteen 2-bit subfields, with bits [1:0] corresponding to pin 0, bits [3:2] corresponding to pin 1, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the first sixteen pins in bank 1. For example, if this register is set to 0x0000002C, the second pin in the bank (GPIO1[1]) will be set to GPIO mode, the third pin in the bank will be set to its second alternate function mode, and bank 1 pins 0 and 3-15 will be set to their primary function modes.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin to GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.14. PINCTRL Bank 1 Upper Pin Mux Select Register Description

The PINCTRL Bank 1 Upper Pin Mux Select Register provides pin function selection for pins 16 through 25 of bank 1.

HW_PINCTRL_MUXSEL3 0x80018120

HW_PINCTRL_MUXSEL3_SET 0x80018124
HW_PINCTRL_MUXSEL3_CLR 0x80018128
HW_PINCTRL_MUXSEL3_TOG 0x8001812C

Table 591. HW_PINCTRL_MUXSEL3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1												FUNC_SEL																			

Table 592. HW_PINCTRL_MUXSEL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	**RSRVD1**	RO	0x0	Always write zeroes to this field.
19:0	**FUNC_SEL**	RW	0xfffff	This field selects which hardware interface block controls each of the last 10 pins in bank 1. This field is divided into ten 2 bit subfields, with bits [1:0] corresponding to pin 16, bits [3:2] corresponding to pin 17, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the last 10 pins in bank 1. For example, if this register is set to 0x00000003, the sixteenth pin in the bank (GPIO1[16]) will be set to GPIO mode and bank 1 pins 17-25 will be set to their primary function mode.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin to GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.15. PINCTRL Bank 1 Drive Strength Register Description

The PINCTRL Bank 1 Drive Strength Register selects the current drive strength for pins in bank 1.

HW_PINCTRL_DRIVE1 0x80018130
HW_PINCTRL_DRIVE1_SET 0x80018134
HW_PINCTRL_DRIVE1_CLR 0x80018138
HW_PINCTRL_DRIVE1_TOG 0x8001813C

Table 593. HW_PINCTRL_DRIVE1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						DRIVE8MA																									

Table 594. HW_PINCTRL_DRIVE1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**DRIVE8MA**	RW	0x0	This field selects the drive strength for pins configured as outputs. This field is segmented into 1 bit per pin subfields, with bit 0 corresponding to bank 1 pin 0, bit 1 corresponding to pin 1, etc. Subfield definitions: 0= 4-mA drive strength 1= 8-mA drive strength

DESCRIPTION:

The PINCTRL Bank 1 Drive Strength Register selects the drive strength (4 mA or 8 mA) for pins in bank 1 that are configured for output.

EXAMPLE:

```
Empty Example.
```

17.6.16. PINCTRL Bank 1 Data Output Register Description

The PINCTRL Bank 1 Data Output Register provides data for all pins in bank 1 that are configured for GPIO output mode.

HW_PINCTRL_DOUT1 0x80018150
HW_PINCTRL_DOUT1_SET 0x80018154
HW_PINCTRL_DOUT1_CLR 0x80018158
HW_PINCTRL_DOUT1_TOG 0x8001815C

Table 595. HW_PINCTRL_DOUT1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						DATAOUT																									

Table 596. HW_PINCTRL_DOUT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**DATAOUT**	RW	0x0	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 26 pins in bank 1.

DESCRIPTION:

This register contains the data that will be driven out all bank 1 pins which are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL2 contains 0x0000000F and HW_PINCTRL_DOE1 contains 0x00000001, then GPIO1[0] will be driven with the value from bit 0 of this register, GPIO1[1] will not be driven, and GPIO1[2:15] will be controlled by the associated primary interfaces.

EXAMPLE:

```
Empty Example.
```

17.6.17. PINCTRL Bank 1 Data Input Register Description

The current value of all bank 1 pins may be read from the PINCTRL Bank 1 Data Input Register.

HW_PINCTRL_DIN1 0x80018160
HW_PINCTRL_DIN1_SET 0x80018164
HW_PINCTRL_DIN1_CLR 0x80018168
HW_PINCTRL_DIN1_TOG 0x8001816C

Table 597. HW_PINCTRL_DIN1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						DATAIN																									

Table 598. HW_PINCTRL_DIN1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**DATAIN**	RO	0x0	Each bit in this read-only register corresponds to one of the 26 pins in bank 1. The current state of each pin in bank 1, synchronized to HCLK, may be read here.

DESCRIPTION:

This register reflects the current values of all the bank 1 pins. The register accurately reflects the state of the pin, regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers. But generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to ensure that the chip is not driving the pin.

For example, if HW_PINCTRL_MUXSEL2 contains 0x0000000F and HW_PINCTRL_DOE1 contains 0x00000001, then pin GPIO1[1] will be an input pin, and bit 1 of this register will reflect its current state.

EXAMPLE:

```
Empty Example.
```

17.6.18. PINCTRL Bank 1 Output Enable Register Description

The PINCTRL Bank 1 Output Enable Register controls the output enable signal for all pins in bank 1 that are configured for GPIO mode.

HW_PINCTRL_DOE1 0x80018170
HW_PINCTRL_DOE1_SET 0x80018174
HW_PINCTRL_DOE1_CLR 0x80018178
HW_PINCTRL_DOE1_TOG 0x8001817C

Table 599. HW_PINCTRL_DOE1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						DATAOE																									

Table 600. HW_PINCTRL_DOE1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**DATAOE**	RW	0x0	Each bit in this register corresponds to one of the 26 pins in bank 1. Setting a bit in this register to one allows the STMP36xx to drive the corresponding pin in GPIO mode.

DESCRIPTION:

For pins in bank 1 that are configured as GPIOs (by setting the pin's control field in HW_PINCTRL_MUXSEL2/3 to 3), a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUT1 to be driven on the pin, and a 0 in this register will disable the corresponding driver. For example, if HW_PINCTRL_MUXSEL2 contains 0x0000000F and HW_PINCTRL_DOE1 contains 0x00000001, then pin GPIO1[0] will be driven with the value from HW_PINCTRL_DOUT1 bit 0, pin GPIO1[1] will be three-stated, and pins GPIO1[2-15] will be controlled by the default peripheral interface associated with each of those pins.

EXAMPLE:

```
Empty Example.
```

17.6.19. PINCTRL Bank 1 Interrupt Select Register Description

The PINCTRL Bank 1 Interrupt Select Register selects which of the bank 1 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ1 0x80018180
HW_PINCTRL_PIN2IRQ1_SET 0x80018184

HW_PINCTRL_PIN2IRQ1_CLR 0x80018188
HW_PINCTRL_PIN2IRQ1_TOG 0x8001818C

Table 601. HW_PINCTRL_PIN2IRQ1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						ENABLE2IRQ																									

Table 602. HW_PINCTRL_PIN2IRQ1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**ENABLE2IRQ**	RW	0x0	Each bit in this register corresponds to one of the 26 pins in bank 1: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 1 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL1 and HW_PINCTRL_IRQPOL1 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT1 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN1 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO1.

For example, if this register contains 0x00000014, then pins GPIO1[2] and GPIO1[4] can be used as interrupt pins, and no other pins in bank 1 will cause bits to be set in the HW_PINCTRL_IRQSTAT1 register.

EXAMPLE:

```
Empty Example.
```

17.6.20. PINCTRL Bank 1 Interrupt Mask Register Description

The PINCTRL Bank 1 Interrupt Mask Register contains interrupt enable masks for the pins in bank 1.

HW_PINCTRL_IRQEN1 0x80018190
HW_PINCTRL_IRQEN1_SET 0x80018194
HW_PINCTRL_IRQEN1_CLR 0x80018198
HW_PINCTRL_IRQEN1_TOG 0x8001819C

Table 603. HW_PINCTRL_IRQEN1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						IRQENABLE																									

Table 604. HW_PINCTRL_IRQEN1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**IRQENABLE**	RW	0x0	Each bit in this register corresponds to one of the 26 pins in bank 1: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT1. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT1.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 1. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT1, an interrupt will be propagated to the interrupt collector as interrupt GPIO1.

For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT1 (corresponding to pins GPIO1[2] and GPIO1[4]) will cause interrupts from bank 1.

EXAMPLE:

```
Empty Example.
```

17.6.21. PINCTRL Bank 1 Interrupt Level/Edge Register Description

The PINCTRL Bank 1 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 1.

HW_PINCTRL_IRQLEVEL1 0x800181A0
HW_PINCTRL_IRQLEVEL1_SET 0x800181A4
HW_PINCTRL_IRQLEVEL1_CLR 0x800181A8
HW_PINCTRL_IRQLEVEL1_TOG 0x800181AC

Table 605. HW_PINCTRL_IRQLEVEL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						BITIRQLEVEL																									

Table 606. HW_PINCTRL_IRQLEVEL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**BITIRQLEVEL**	RW	0x0	Each bit in this register corresponds to one of the 26 pins in bank 1: 1= Level detection 0= Edge detection

DESCRIPTION:

This register selects level or edge detection for interrupt generation. Each pin in bank 1 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL1 (see below) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.22. PINCTRL Bank 1 Interrupt Polarity Register Description

The PINCTRL Bank 1 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 1.

HW_PINCTRL_IRQPOL1 0x800181B0
HW_PINCTRL_IRQPOL1_SET 0x800181B4
HW_PINCTRL_IRQPOL1_CLR 0x800181B8
HW_PINCTRL_IRQPOL1_TOG 0x800181BC

Table 607. HW_PINCTRL_IRQPOL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						IRQPOL																									

Table 608. HW_PINCTRL_IRQPOL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**IRQPOL**	RW	0x0	Each bit in this register corresponds to one of the 26 pins in bank 1: 0= Low or falling edge 1= High or rising edge

DESCRIPTION:

This register selects the polarity for interrupt generation. Each pin in bank 1 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL1 (see above) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.23. PINCTRL Bank 1 Interrupt Status Register Description

The PINCTRL Bank 1 Interrupt Status Register reflects pending interrupt status for the pins in bank 1.

HW_PINCTRL_IRQSTAT1 0x800181C0
HW_PINCTRL_IRQSTAT1_SET 0x800181C4
HW_PINCTRL_IRQSTAT1_CLR 0x800181C8
HW_PINCTRL_IRQSTAT1_TOG 0x800181CC

Table 609. HW_PINCTRL_IRQSTAT1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1						STAT																									

Table 610. HW_PINCTRL_IRQSTAT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD1**	RO	0x0	Always write zeroes to this field.
25:0	**STAT**	RW	0x0	Each bit in this register corresponds to one of the 26 pins in bank 1: 0= No interrupt pending 1= Interrupt pending

DESCRIPTION:

This register reflects the pending interrupt status for pins in bank 1. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 1 pin that has been enabled as an interrupt source in the HW_PINCTRL_PIN2IRQ1 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT1_CLR. Status bits for pins configured as level-sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ1.

If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN1 mask register, then the GPIO1 interrupt will be asserted to the interrupt collector.

EXAMPLE:

```
Empty Example.
```

17.6.24. PINCTRL Bank 2 Lower Pin Mux Select Register Description

The PINCTRL Bank 2 Lower Pin Mux Select Register provides pin function selection for pins 0 through 15 of bank 2.

HW_PINCTRL_MUXSEL4 0x80018210
HW_PINCTRL_MUXSEL4_SET 0x80018214
HW_PINCTRL_MUXSEL4_CLR 0x80018218

HW_PINCTRL_MUXSEL4_TOG 0x8001821C

Table 611. HW_PINCTRL_MUXSEL4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNC_SEL																															

Table 612. HW_PINCTRL_MUXSEL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FUNC_SEL**	RW	0xffffffff	This field selects which hardware interface block controls each of the first 16 pins in bank 2. This field is divided into sixteen 2 bit subfields, with bits [1:0] corresponding to pin 0, bits [3:2] corresponding to pin 1, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the first sixteen pins in bank 2. For example, if this register is set to 0x0000002C, the second pin in the bank (GPIO2[1]) will be set to GPIO mode, the third pin in the bank will be set to its second alternate function mode, and bank 2 pins 0 and 3-15 will be set to their primary function modes.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin to GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.25. PINCTRL Bank 2 Upper Pin Mux Select Register Description

The PINCTRL Bank 2 Upper Pin Mux Select Register provides pin function selection for pins 16 through 31 of bank 2.

HW_PINCTRL_MUXSEL5 0x80018220
HW_PINCTRL_MUXSEL5_SET 0x80018224
HW_PINCTRL_MUXSEL5_CLR 0x80018228
HW_PINCTRL_MUXSEL5_TOG 0x8001822C

Table 613. HW_PINCTRL_MUXSEL5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNC_SEL																															

Table 614. HW_PINCTRL_MUXSEL5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FUNC_SEL**	RW	0xffffffff	This field selects which hardware interface block controls each of the last 16 pins in bank 2. This field is divided into sixteen 2 bit subfields, with bits [1:0] corresponding to pin 16, bits [3:2] corresponding to pin 17, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the last sixteen pins in bank 2. For example, if this register is set to 0x00000003, the sixteenth pin in the bank (GPIO2[16]) will be set to GPIO mode and bank 2 pins 17-31 will be set to their primary function mode.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin to GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.26. PINCTRL Bank 2 Drive Strength Register Description

The PINCTRL Bank 2 Drive Strength Register selects the current drive strength for pins in bank 2.

HW_PINCTRL_DRIVE2 0x80018230
HW_PINCTRL_DRIVE2_SET 0x80018234
HW_PINCTRL_DRIVE2_CLR 0x80018238
HW_PINCTRL_DRIVE2_TOG 0x8001823C

Table 615. HW_PINCTRL_DRIVE2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DRIVE8MA																															

Table 616. HW_PINCTRL_DRIVE2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DRIVE8MA**	RW	0x00000000	This field selects the drive strength for pins configured as outputs. This field is segmented into 1 bit per pin subfields, with bit 0 corresponding to bank 3 pin 0, bit 1 corresponding to pin 1, etc. Subfield definitions: 0= 4-mA drive strength 1= 8-mA drive strength

DESCRIPTION:

The Drive Strength register selects the drive strength (4 mA or 8 mA) for pins in bank 2 that are configured for output.

EXAMPLE:

```
Empty Example.
```

17.6.27. PINCTRL Bank 2 Data Output Register Description

The PINCTRL Bank 2 Data Output Register provides data for all pins in bank 2 that are configured for GPIO output mode.

HW_PINCTRL_DOUT2 0x80018250
HW_PINCTRL_DOUT2_SET 0x80018254
HW_PINCTRL_DOUT2_CLR 0x80018258
HW_PINCTRL_DOUT2_TOG 0x8001825C

Table 617. HW_PINCTRL_DOUT2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATAOUT																															

Table 618. HW_PINCTRL_DOUT2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATAOUT**	RW	0x00000000	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 32 pins in bank 2.

DESCRIPTION:

This register contains the data that will be driven out all bank 2 pins that are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL4 contains 0x0000000F and HW_PINCTRL_DOE2 contains 0x00000001, then GPIO2[0] will be driven with the value from bit 0 of this register, GPIO2[1] will not be driven, and GPIO2[2:15] will be controlled by the associated primary interfaces.

EXAMPLE:

```
Empty Example.
```

17.6.28. PINCTRL Bank 2 Data Input Register Description

The current value of all bank 2 pins may be read from the PINCTRL Bank 2 Data Input Register.

HW_PINCTRL_DIN2 0x80018260
HW_PINCTRL_DIN2_SET 0x80018264
HW_PINCTRL_DIN2_CLR 0x80018268
HW_PINCTRL_DIN2_TOG 0x8001826C

Table 619. HW_PINCTRL_DIN2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATAIN																															

Table 620. HW_PINCTRL_DIN2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATAIN**	RO	0x00000000	Each bit in this read-only register corresponds to one of the 32 pins in bank 2. The current state of each pin in bank 2, synchronized to HCLK, may be read here.

DESCRIPTION:

This register reflects the current values of all the bank 2 pins. The register accurately reflects the state of the pin regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers. But generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to ensure that the chip is not driving the pin.

For example, if HW_PINCTRL_MUXSEL4 contains 0x0000000F and HW_PINCTRL_DOE2 contains 0x00000001, then pin GPIO2[1] will be an input pin, and bit 1 of this register will reflect its current state.

EXAMPLE:

```
Empty Example.
```

17.6.29. PINCTRL Bank 2 Output Enable Register Description

The PINCTRL Bank 2 Output Enable Register controls the output enable signal for all pins in bank 2 that are configured for GPIO mode.

HW_PINCTRL_DOE2 0x80018270
HW_PINCTRL_DOE2_SET 0x80018274
HW_PINCTRL_DOE2_CLR 0x80018278
HW_PINCTRL_DOE2_TOG 0x8001827C

Table 621. HW_PINCTRL_DOE2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATAOE																															

Table 622. HW_PINCTRL_DOE2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATAOE**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 2. Setting a bit in this register to one allows the STMP36xx to drive the corresponding pin in GPIO mode.

DESCRIPTION:

For pins in bank 2 that are configured as GPIOs (by setting the pin's control field in HW_PINCTRL_MUXSEL4/5 to 3), a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUT2 to be driven out the pin, and a 0 in this register will disable the corresponding driver. For example, if HW_PINCTRL_MUXSEL4 contains 0x0000000F and HW_PINCTRL_DOE2 contains 0x00000001, then pin GPIO2[0] will be driven with the value from HW_PINCTRL_DOUT2 bit 0, pin

GPIO1[1] will be three-stated, and pins GPIO2[2-15] will be controlled by the default peripheral interface associated with each of those pins.

EXAMPLE:

```
Empty Example.
```

17.6.30. PINCTRL Bank 2 Interrupt Select Register Description

The PINCTRL Bank 2 Interrupt Select Register selects which of the bank 2 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ2 0x80018280
HW_PINCTRL_PIN2IRQ2_SET 0x80018284
HW_PINCTRL_PIN2IRQ2_CLR 0x80018288
HW_PINCTRL_PIN2IRQ2_TOG 0x8001828C

Table 623. HW_PINCTRL_PIN2IRQ2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ENABLE2IRQ																															

Table 624. HW_PINCTRL_PIN2IRQ2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**ENABLE2IRQ**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 2: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 2 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL2 and HW_PINCTRL_IRQPOL2 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT2 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN2 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO2.

For example, if this register contains 0x00000014, then pins GPIO2[2] and GPIO2[4] can be used as interrupt pins, and no other pins in bank 2 will cause bits to be set in the HW_PINCTRL_IRQSTAT2 register.

EXAMPLE:

```
Empty Example.
```

17.6.31. PINCTRL Bank 2 Interrupt Mask Register Description

The PINCTRL Bank 2 Interrupt Mask Register contains interrupt enable masks for the pins in bank 2.

HW_PINCTRL_IRQEN2 0x80018290
HW_PINCTRL_IRQEN2_SET 0x80018294
HW_PINCTRL_IRQEN2_CLR 0x80018298
HW_PINCTRL_IRQEN2_TOG 0x8001829C

Table 625. HW_PINCTRL_IRQEN2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IRQENABLE																															

Table 626. HW_PINCTRL_IRQEN2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**IRQENABLE**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 2: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT2. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT2.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 2. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT2, an interrupt will be propagated to the interrupt collector as interrupt GPIO2.

For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT2 (corresponding to pins GPIO2[2] and GPIO2[4]) will cause interrupts from bank 2.

EXAMPLE:

```
Empty Example.
```

17.6.32. PINCTRL Bank 2 Interrupt Level/Edge Register Description

The PINCTRL Bank 2 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 2.

HW_PINCTRL_IRQLEVEL2 0x800182A0
HW_PINCTRL_IRQLEVEL2_SET 0x800182A4
HW_PINCTRL_IRQLEVEL2_CLR 0x800182A8
HW_PINCTRL_IRQLEVEL2_TOG 0x800182AC

Table 627. HW_PINCTRL_IRQLEVEL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITIRQLEVEL																															

Table 628. HW_PINCTRL_IRQLEVEL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITIRQLEVEL**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 2: 1= Level detection 0= Edge detection

DESCRIPTION:

This register selects level or edge detection for interrupt generation. Each pin in bank 2 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL2 (see below) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.33. PINCTRL Bank 2 Interrupt Polarity Register Description

The PINCTRL Bank 2 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 2.

HW_PINCTRL_IRQPOL2 0x800182B0
HW_PINCTRL_IRQPOL2_SET 0x800182B4
HW_PINCTRL_IRQPOL2_CLR 0x800182B8
HW_PINCTRL_IRQPOL2_TOG 0x800182BC

Table 629. HW_PINCTRL_IRQPOL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IRQPOL																															

Table 630. HW_PINCTRL_IRQPOL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**IRQPOL**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 2: 0= Low or falling edge 1= High or rising edge

DESCRIPTION:

This register selects the polarity for interrupt generation. Each pin in bank 2 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL2 (see above) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.34. PINCTRL Bank 2 Interrupt Status Register Description

The PINCTRL Bank 2 Interrupt Status Register reflects pending interrupt status for the pins in bank 2.

HW_PINCTRL_IRQSTAT2 0x800182C0
HW_PINCTRL_IRQSTAT2_SET 0x800182C4
HW_PINCTRL_IRQSTAT2_CLR 0x800182C8
HW_PINCTRL_IRQSTAT2_TOG 0x800182CC

Table 631. HW_PINCTRL_IRQSTAT2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
STAT																															

Table 632. HW_PINCTRL_IRQSTAT2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**STAT**	RW	0x00000000	Each bit in this register corresponds to one of the 32 pins in bank 2: 0= No interrupt pending 1= Interrupt pending

DESCRIPTION:

This register reflects the pending interrupt status for pins in bank 2. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 2 pin that has been enabled as an interrupt source in the HW_PINCTRL_PIN2IRQ2 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT2_CLR. Status bits for pins configured as level-sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ2.

If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN2 mask register, then the GPIO2 interrupt will be asserted to the interrupt collector.

EXAMPLE:

```
Empty Example.
```

17.6.35. PINCTRL Bank 3 Lower Pin Mux Select Register Description

The PINCTRL Bank 3 Lower Pin Mux Select Register provides pin function selection for pins 0 through 15 of bank 3.

HW_PINCTRL_MUXSEL6 0x80018310
HW_PINCTRL_MUXSEL6_SET 0x80018314
HW_PINCTRL_MUXSEL6_CLR 0x80018318
HW_PINCTRL_MUXSEL6_TOG 0x8001831C

Table 633. HW_PINCTRL_MUXSEL6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
FUNC_SEL																															

Table 634. HW_PINCTRL_MUXSEL6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**FUNC_SEL**	RW	0xffffffff	This field selects which hardware interface block controls each of the first 16 pins in bank 3. This field is divided into sixteen 2 bit subfields, with bits [1:0] corresponding to pin 0, bits [3:2] corresponding to pin 1, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the first sixteen pins in bank 3. For example, if this register is set to 0x0000002C, the second pin in the bank (GPIO3[1]) will be set to GPIO mode, the third pin in the bank will be set to its second alternate function mode, and bank 3 pins 0 and 3-15 will be set to their primary function modes.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin-to-GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.36. PINCTRL Bank 3 Upper Pin Mux Select Register Description

The PINCTRL Bank 3 Upper Pin Mux Select Register provides pin function selection for pins 16 through 18 of bank 3.

HW_PINCTRL_MUXSEL7 0x80018320
HW_PINCTRL_MUXSEL7_SET 0x80018324
HW_PINCTRL_MUXSEL7_CLR 0x80018328
HW_PINCTRL_MUXSEL7_TOG 0x8001832C

Table 635. HW_PINCTRL_MUXSEL7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																										FUNC_SEL					

Table 636. HW_PINCTRL_MUXSEL7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:6	**RSRVD1**	RO	0x0	Always write zeroes to this field.
5:0	**FUNC_SEL**	RW	0x3f	This field selects which hardware interface block controls each of the last 3 pins in bank 3. This field is divided into three 2 bit subfields, with bits [1:0] corresponding to pin 16, bits [3:2] corresponding to pin 17, etc. Subfield definitions: 00= Default peripheral 01= Alternate peripheral1 or undefined 10= Alternate peripheral2 or undefined 11= GPIO

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the last three pins in bank 3. For example, if this register is set to 0x00000003, the sixteenth pin in the bank (GPIO0[16]) will be set to GPIO mode and bank pins 17-18 will be set to their primary function mode.

See the table in the Pin Interface Multiplexing section earlier in this chapter for information about pin-to-GPIO bank mapping.

EXAMPLE:

```
Empty Example.
```

17.6.37. PINCTRL Bank 3 Drive Strength Register Description

The PINCTRL Bank 3 Drive Strength Register selects the current drive strength for pins in bank 3.

HW_PINCTRL_DRIVE3 0x80018330
HW_PINCTRL_DRIVE3_SET 0x80018334
HW_PINCTRL_DRIVE3_CLR 0x80018338
HW_PINCTRL_DRIVE3_TOG 0x8001833C

Table 637. HW_PINCTRL_DRIVE3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													DRIVE8MA																		

Table 638. HW_PINCTRL_DRIVE3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**DRIVE8MA**	RW	0x0	This field selects the drive strength for pins configured as outputs. This field is segmented into 1 bit per pin subfields, with bit 0 corresponding to bank 3 pin 0, bit 1 corresponding to pin 1, etc. Subfield definitions: 0= 4-mA drive strength 1= 8-mA drive strength (16-mA for pins PWM3 and PWM4) Note that two pins, PWM3 and PWM4, corresponding to bits 13 and 14 repectively have stronger drivers than the other pins. When asserting these bits their pins have 16-mA drive strength instead of 8-mA.

DESCRIPTION:

The PINCTRL Bank 3 Drive Strength Register selects the drive strength (4 mA or 8 mA) for pins in bank 3 that are configured for output. Note that two pins, PWM3 and PWM4, corresponding to bits 13 and 14 repectively have stronger drivers than the other pins. When asserting these bits, their pins have 16-mA drive strength instead of 8-mA.

EXAMPLE:

```
Empty Example.
```

17.6.38. PINCTRL Bank 3 Data Output Register Description

The PINCTRL Bank 3 Data Output Register provides data for all pins in bank 3 that are configured for GPIO output mode.

HW_PINCTRL_DOUT3 0x80018350
HW_PINCTRL_DOUT3_SET 0x80018354
HW_PINCTRL_DOUT3_CLR 0x80018358
HW_PINCTRL_DOUT3_TOG 0x8001835C

Table 639. HW_PINCTRL_DOUT3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													DATAOUT																		

Table 640. HW_PINCTRL_DOUT3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**DATAOUT**	RW	0x0	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 19 pins in bank 3.

DESCRIPTION:

This register contains the data that will be driven out all bank 3 pins that are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL6 contains 0x0000000F and HW_PINCTRL_DOE3 contains 0x00000001, then GPIO3[0] will be driven with the value from bit 0 of this register, GPIO3[1] will not be driven, and GPIO3[2:15] will be controlled by the associated primary interfaces.

EXAMPLE:

```
Empty Example.
```

17.6.39. PINCTRL Bank 3 Data Input Register Description

The current value of all bank 3 pins may be read from the PINCTRL Bank 3 Data Input Register.

HW_PINCTRL_DIN3 0x80018360
HW_PINCTRL_DIN3_SET 0x80018364
HW_PINCTRL_DIN3_CLR 0x80018368
HW_PINCTRL_DIN3_TOG 0x8001836C

Table 641. HW_PINCTRL_DIN3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													DATAIN																		

Table 642. HW_PINCTRL_DIN3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**DATAIN**	RO	0x0	Each bit in this read-only register corresponds to one of the 19 pins in bank 2. The current state of each pin in bank 2, synchronized to HCLK, may be read here.

DESCRIPTION:

This register reflects the current values of all the bank 3 pins. The register accurately reflects the state of the pin regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers. But generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to ensure that the chip is not driving the pin.

For example, if HW_PINCTRL_MUXSEL6 contains 0x0000000F and HW_PINCTRL_DOE3 contains 0x00000001, then pin GPIO3[1] will be an input pin, and bit 1 of this register will reflect its current state.

EXAMPLE:

```
Empty Example.
```

17.6.40. PINCTRL Bank 3 Output Enable Register Description

The PINCTRL Bank 3 Output Enable Register controls the output enable signal for all pins in bank 3 that are configured for GPIO mode.

HW_PINCTRL_DOE3 0x80018370
HW_PINCTRL_DOE3_SET 0x80018374
HW_PINCTRL_DOE3_CLR 0x80018378
HW_PINCTRL_DOE3_TOG 0x8001837C

Table 643. HW_PINCTRL_DOE3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													DATAOE																		

Table 644. HW_PINCTRL_DOE3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**DATAOE**	RW	0x0	Each bit in this register corresponds to one of the 19 pins in bank 3. Setting a bit in this register to one allows the STMP36xx to drive the corresponding pin in GPIO mode.

DESCRIPTION:

For pins in bank 3 that are configured as GPIOs (by setting the pin's control field in HW_PINCTRL_MUXSEL6/7 to 3), a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUT3 to be driven out the pin, and a 0 in this register will disable the corresponding driver. For example, if HW_PINCTRL_MUXSEL6 contains 0x0000000F and HW_PINCTRL_DOE3 contains 0x00000001, then pin GPIO3[0] will be driven with the value from HW_PINCTRL_DOUT3 bit 0, pin GPIO3[1] will be three-stated, and pins GPIO3[2-15] will be controlled by the default peripheral interface associated with each of those pins.

EXAMPLE:

```
Empty Example.
```

17.6.41. PINCTRL Bank 3 Interrupt Select Register Description

The PINCTRL Bank 3 Interrupt Select Register selects which of the bank 3 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ3 0x80018380
HW_PINCTRL_PIN2IRQ3_SET 0x80018384
HW_PINCTRL_PIN2IRQ3_CLR 0x80018388
HW_PINCTRL_PIN2IRQ3_TOG 0x8001838C

Table 645. HW_PINCTRL_PIN2IRQ3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													ENABLE2IRQ																		

Table 646. HW_PINCTRL_PIN2IRQ3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**ENABLE2IRQ**	RW	0x0	Each bit in this register corresponds to one of the 19 pins in bank 0: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 3 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL3 and HW_PINCTRL_IRQPOL3 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT3 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN3 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO3.

For example, if this register contains 0x00000014, then pins GPIO3[2] and GPIO3[4] can be used as interrupt pins, and no other pins in bank 3 will cause bits to be set in the HW_PINCTRL_IRQSTAT3 register.

EXAMPLE:

```
Empty Example.
```

17.6.42. PINCTRL Bank 3 Interrupt Mask Register Description

The PINCTRL Bank 3 Interrupt Mask Register contains interrupt enable masks for the pins in bank 3.

HW_PINCTRL_IRQEN3 0x80018390
HW_PINCTRL_IRQEN3_SET 0x80018394
HW_PINCTRL_IRQEN3_CLR 0x80018398
HW_PINCTRL_IRQEN3_TOG 0x8001839C

Table 647. HW_PINCTRL_IRQEN3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													IRQENABLE																		

Table 648. HW_PINCTRL_IRQEN3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**IRQENABLE**	RW	0x0	Each bit in this register corresponds to one of the 19 pins in bank 3: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT3. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT3.

DESCRIPTION:

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 3. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT3, an interrupt will be propagated to the interrupt collector as interrupt GPIO3.

For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT3 (corresponding to pins GPIO3[2] and GPIO3[4]) will cause interrupts from bank 3.

EXAMPLE:

```
Empty Example.
```

17.6.43. PINCTRL Bank 3 Interrupt Level/Edge Register Description

The PINCTRL Bank 3 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 3.

HW_PINCTRL_IRQLEVEL3 0x800183A0
HW_PINCTRL_IRQLEVEL3_SET 0x800183A4
HW_PINCTRL_IRQLEVEL3_CLR 0x800183A8
HW_PINCTRL_IRQLEVEL3_TOG 0x800183AC

Table 649. HW_PINCTRL_IRQLEVEL3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													BITIRQLEVEL																		

Table 650. HW_PINCTRL_IRQLEVEL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**BITIRQLEVEL**	RW	0x0	Each bit in this register corresponds to one of the 19 pins in bank 3: 1= Level detection 0= Edge detection

DESCRIPTION:

This register selects level or edge detection for interrupt generation. Each pin in bank 3 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL3 (see below) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.44. PINCTRL Bank 3 Interrupt Polarity Register Description

The PINCTRL Bank 3 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 3.

HW_PINCTRL_IRQPOL3 0x800183B0
HW_PINCTRL_IRQPOL3_SET 0x800183B4
HW_PINCTRL_IRQPOL3_CLR 0x800183B8
HW_PINCTRL_IRQPOL3_TOG 0x800183BC

Table 651. HW_PINCTRL_IRQPOL3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													IRQPOL																		

Table 652. HW_PINCTRL_IRQPOL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**IRQPOL**	RW	0x0	Each bit in this register corresponds to one of the 19 pins in bank 3: 0= Low or falling edge 1= High or rising edge

DESCRIPTION:

This register selects the polarity for interrupt generation. Each pin in bank 3 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL3 (see above) appropriately.

EXAMPLE:

```
Empty Example.
```

17.6.45. PINCTRL Bank 3 Interrupt Status Register Description

The PINCTRL Bank 3 Interrupt Status Register reflects pending interrupt status for the pins in bank 3.

HW_PINCTRL_IRQSTAT3 0x800183C0
HW_PINCTRL_IRQSTAT3_SET 0x800183C4
HW_PINCTRL_IRQSTAT3_CLR 0x800183C8
HW_PINCTRL_IRQSTAT3_TOG 0x800183CC

Table 653. HW_PINCTRL_IRQSTAT3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1													STAT																		

Table 654. HW_PINCTRL_IRQSTAT3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	**RSRVD1**	RO	0x0	Always write zeroes to this field.
18:0	**STAT**	RW	0x0	Each bit in this register corresponds to one of the 19 pins in bank 3: 0= No interrupt pending 1= Interrupt pending

DESCRIPTION:

This register reflects the pending interrupt status for pins in bank 3. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 3 pin that has been enabled as an interrupt source in the HW_PINCTRL_PIN2IRQ3 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT3_CLR. Status bits for pins configured as level-sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ3.

If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN3 mask register, then the GPIO3 interrupt will be asserted to the interrupt collector.

EXAMPLE:

```
Empty Example.
```

PINCTRL XML Revision: 1.20

18. TIMERS AND ROTARY DECODER

This chapter describes the timers and rotary decoder included on the STMP36xx. Programmable registers are described in Section 18.4.

18.1. Overview

The STMP36xx implements four timers and a rotary decoder, as shown in Figure 80. The timers and decoder can take their inputs from any of the pins defined for PWM, rotary encoders, or certain divisions from the 32-kHz clock input. Thus, the PWM pins can be inputs or outputs, depending on the application.

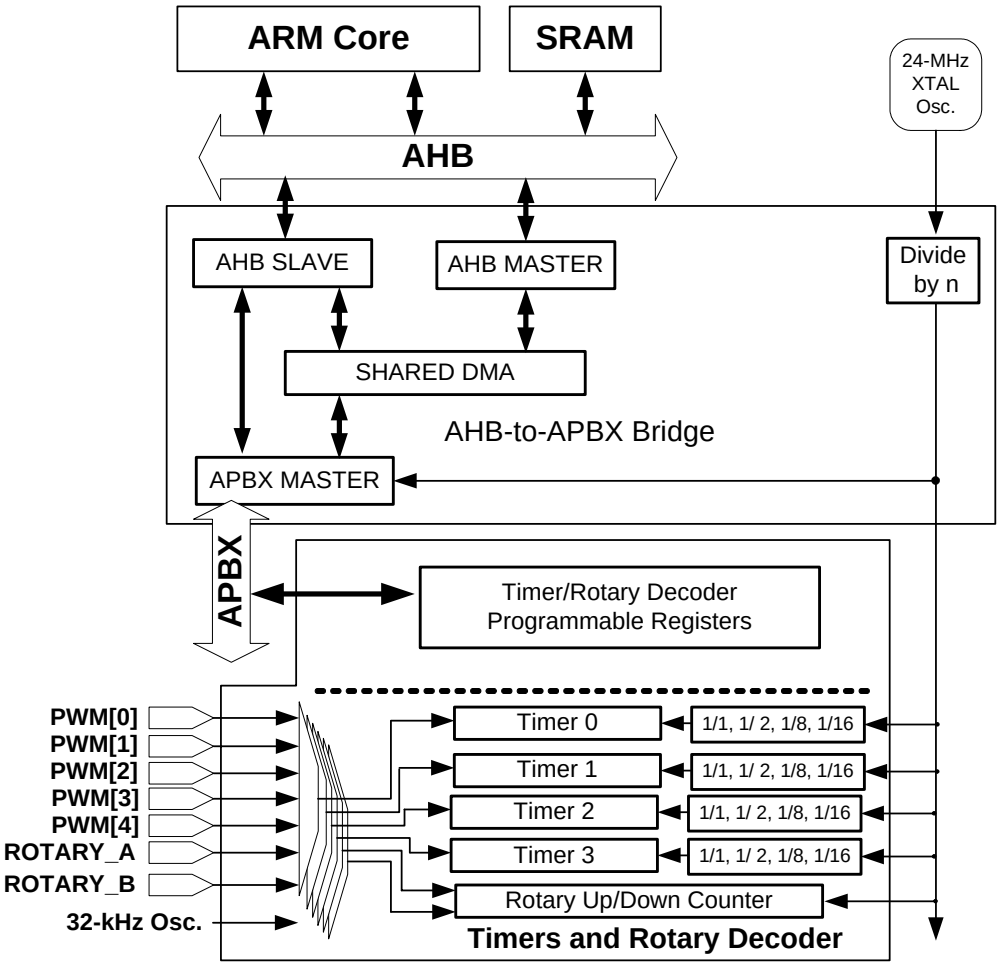


Figure 80. Timers and Rotary Decoder Block Diagram

The timer/rotary decoder block is a programmed I/O interface connected to the APBX bus. Recall that the APBX typically runs at a divided clock rate from the 24-MHz crystal clock (6 MHz). Each timer and rotary channel can sample at a rate that is further subdivided from the APBX clock. Each timer can select a different pre-scaler value.

18.2. Timers

Each of the four timers consists of a 16-bit fixed count value and a 16-bit free-running count value. In most cases, the free-running count decrements to zero. When it decrements to zero, it sets an interrupt status bit associated with the counter.

- If the RELOAD bit is set to one, then the fixed count is automatically copied to the free-running counter and the count continues.
- If the RELOAD bit is not set, the timer stalls when it reaches zero.

Figure 81 shows a detailed view of either Timer 0, Timer 1, or Timer 2. Timer 3 has additional functionality, which is shown in Figure 82.

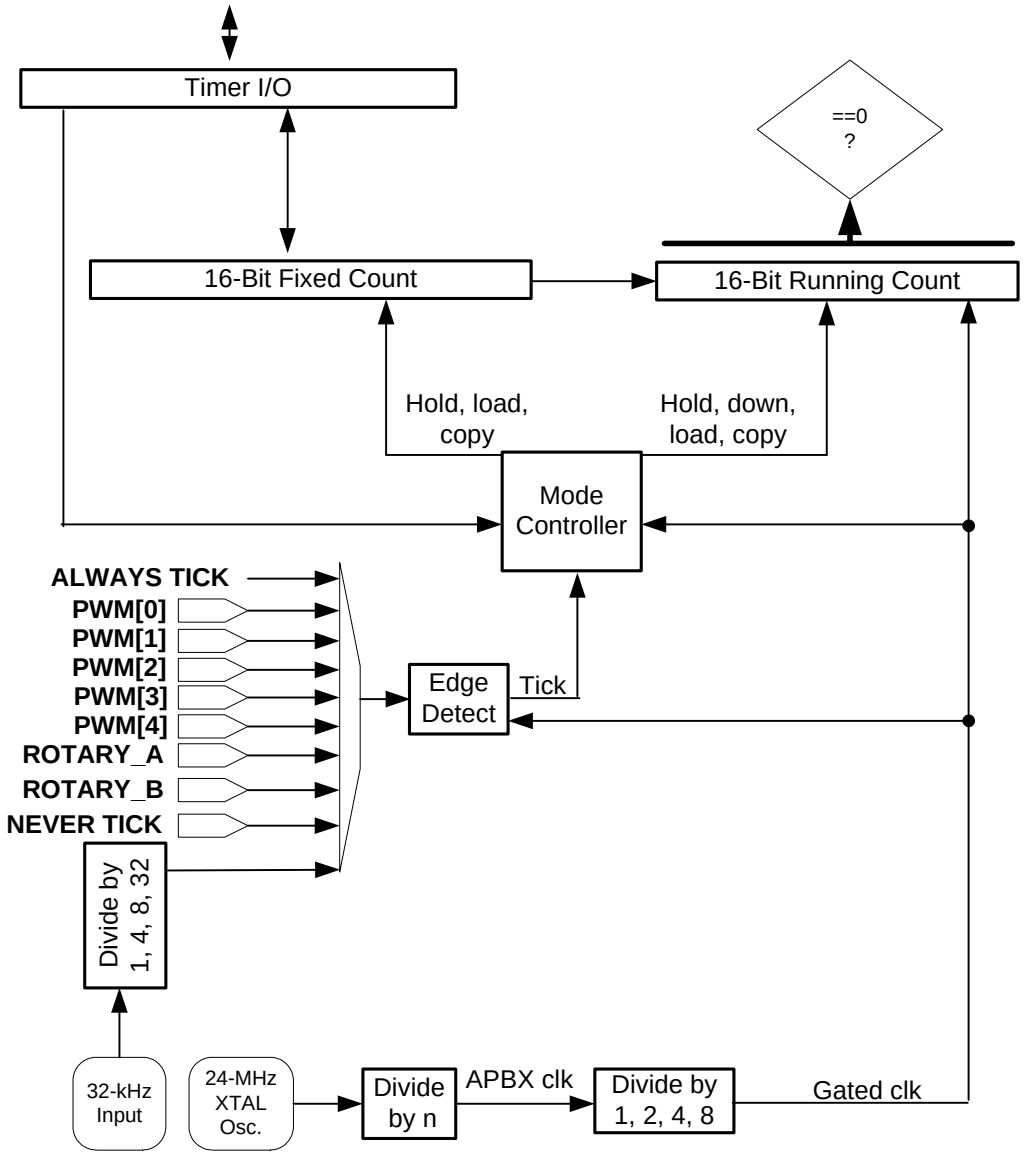


Figure 81. Timer 0, Timer 1, or Timer 2 Detail

Each timer has an UPDATE bit that controls whether the free-running-counter is loaded at the same time the fixed-count register is written from the CPU. The output of each timer's source select has a polarity control that allows the timer to operate on either edge.

Table 655 lists the timer state machine transitions.

Table 655. Timer State Machine Transitions

UPDATE	RELOAD	RUNNING
0	0	PIO writes to the fixed-count bit field have no effect on the running count.
0	1	The value written to the fixed count is used to reload the running count the next time it reaches zero. When the fixed count has been written with a value of zero and the running count reaches zero, it continuously copies the fixed count value to the running count. Thus, writing a non-zero value to the fixed count register kicks off a continuous count and update operation.
1	0	The value written to the fixed count bit field is copied, immediately, to the running count, restarting any existing running count operation. When the new running count reaches zero, it freezes.
1	1	The value written to the fixed count bit field is copied, immediately, to the running count, restarting any existing running count operation. When the new running count reaches zero, it is reloaded from the value in the fixed count bit field, thus running continuously using the newly supplied fixed count.

When generating a periodic timer interrupt using the RELOAD bit, the user must compute the proper fixed-count value (count_value) based on clock speeds and clock divider settings. Note that, in this case, the actual value written to the FIXED_COUNT register field should be count_value – 1. For one-shot interrupts (RELOAD bit not set), the value written should be count_value.

For proper detection of the input source signal, it should be much slower than the pre-scaled APBX clock (no greater than one-third the frequency of the pre-scaled APBX clock).

Selecting the ALWAYS tick causes the timer to decrement continuously at the rate established by the pre-scaled APBX clock. The NEVER TICK selection causes the timer to stall. Setting the fixed-count to 0xFFFF and setting the RELOAD bit causes the timer to operate in a continuous-count 65536 count mode.

The state of the 16-bit free-running count can be read by the CPU for each timer.

18.2.1. Using External Signals as Inputs

External signals can be used as inputs to the block. They can be used as either the test signal or sampling input signals (duty cycle or normal timer mode). This can be accomplished by using the rotary input pins or any unused PWM pins. If PWM pins are being used for this purpose, conflicts with the PWM or other blocks that could drive the pins as outputs must be avoided. In this case, the PWM pins being used should be programmed as GPIO inputs. (See Chapter 17, "Pin Control and GPIO" on page 429 for details.) Then, the external signal can be wired to the pin, and the PWM number selected in the appropriate TIMROT registers.

18.2.2. Timer 3 and Duty Cycle Mode

Timer 3 can operate in the same modes as Timer 0, Timer 1, and Timer 2. However, it has an additional duty cycle measurement mode. Figure 82 shows a detailed view of Timer 3.

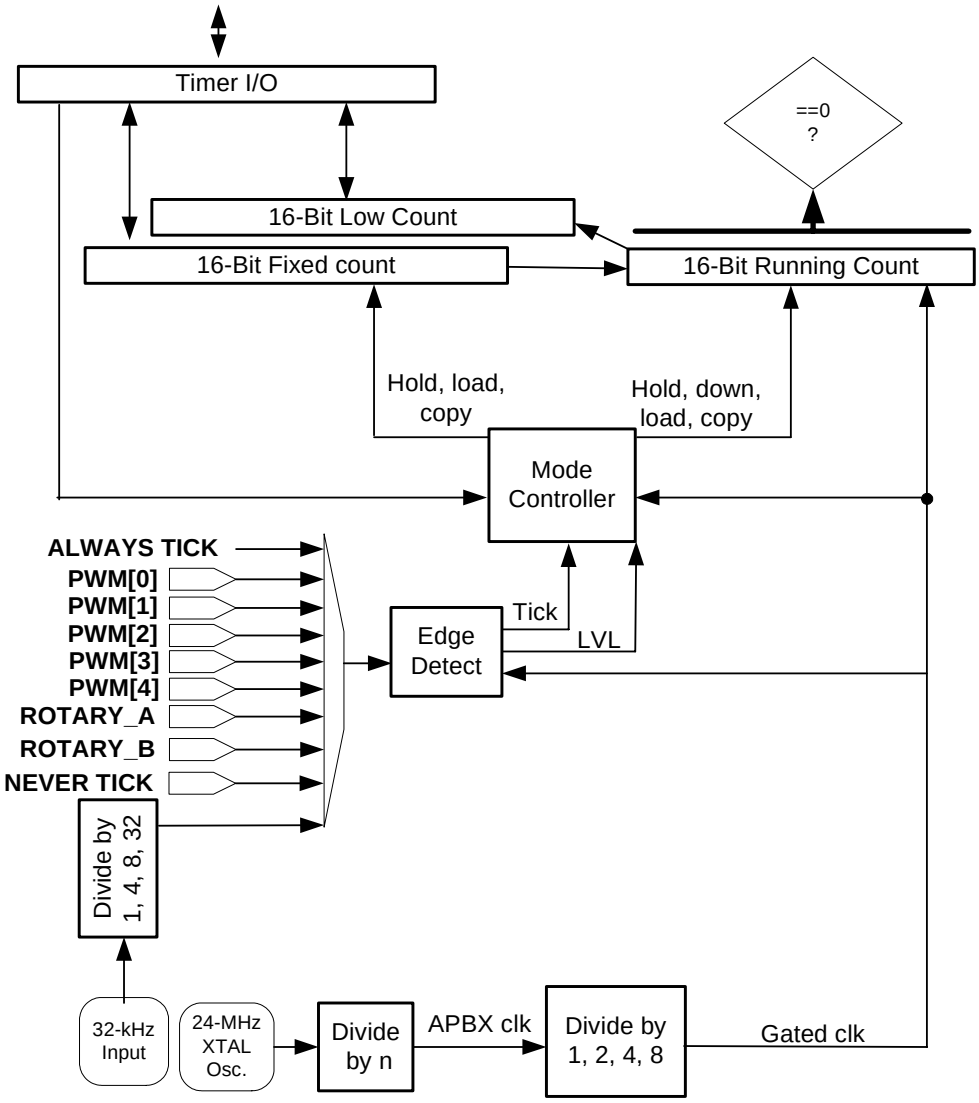


Figure 82. Timer 3 Detail

In the duty cycle mode, Timer 3 samples the free-running counter at the rising and falling edges of the input test signal, resetting the free-running counter on the same clock that is sampled.

- On the rising edge of the test signal, the free-running count is copied to the LOW_RUNNING_COUNT bit field of the HW_TIMROT_TIMCOUNT3 register.
- On the falling edge of the source clock, the free-running count is copied to the HIGH_FIXED_COUNT bit field (as shown in Figure 83).

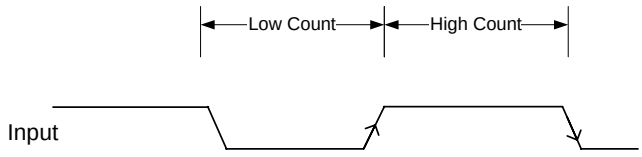


Figure 83. Pulse-Width Measurement Mode

- Once duty cycle mode is programmed and the input signal is stable, software should poll the DUTY_VALID bit in the HW_TIMROT_TIMCTRL3 register.
- This bit is automatically set and cleared by the hardware. When this bit is set, count values in the HW_TIMROT_TIMCOUNT3 register are stable and ready to be read.

Refer to the Timer 3 control and status register, HW_TIMROT_TIMCTRL3, where the DUTY_CYCLE bit controls whether HW_TIMROT_TIMCOUNT3 register's LOW_RUNNING_COUNT bit field reads back the running count or the low count of a duty cycle measurement. The DUTY_CYCLE bit also controls whether the HIGH_FIXED_COUNT bit field reads back the fixed-count value used in normal timer operations or the duty cycle high-time measurement.

It should be noted that for duty cycle mode to function properly, the timer "tick" source selected (SELECT field of the HW_TIMROT_TIMCTRL3 register) should be an appropriate frequency to sample the test signal. The NEVER_TICK value should never be used in this mode, as it will yield incorrect count results.

18.2.3. Testing Timer 3 Duty Cycle Modes

To test the duty cycle modes of Time r3, select PWM1 as the input. PWM1 can generate waveforms of arbitrary duty cycle suitable for testing the duty cycle measurement capability.

18.3. Rotary Decoder

The rotary decoder uses two input selectors and edge detectors, as shown in Figure 84. It includes a debounce circuit for each input, as shown in Figure 85. This figure shows the debounce circuit for input A, though the circuit is identical for input B.

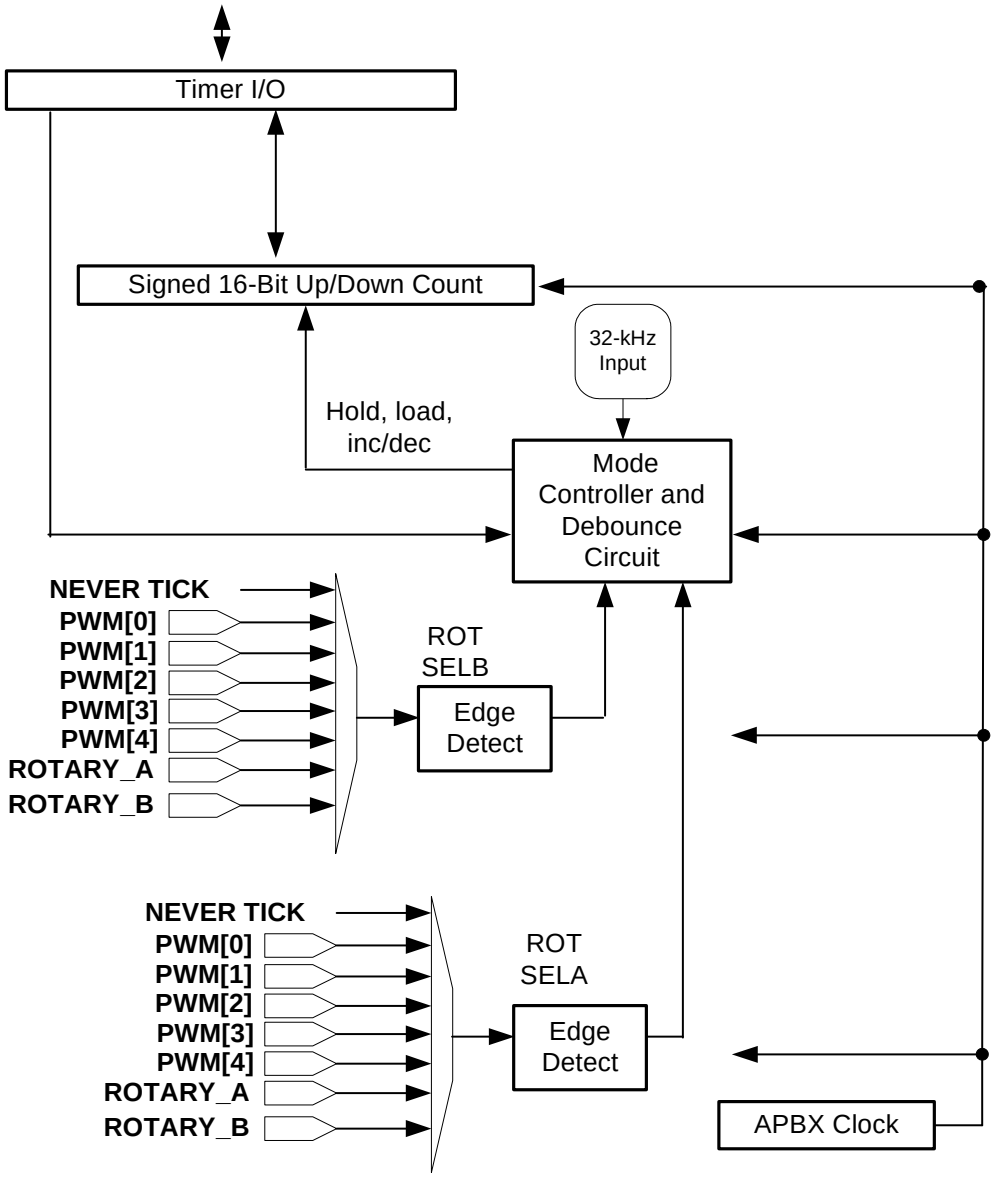


Figure 84. Detail of Rotary Decoder

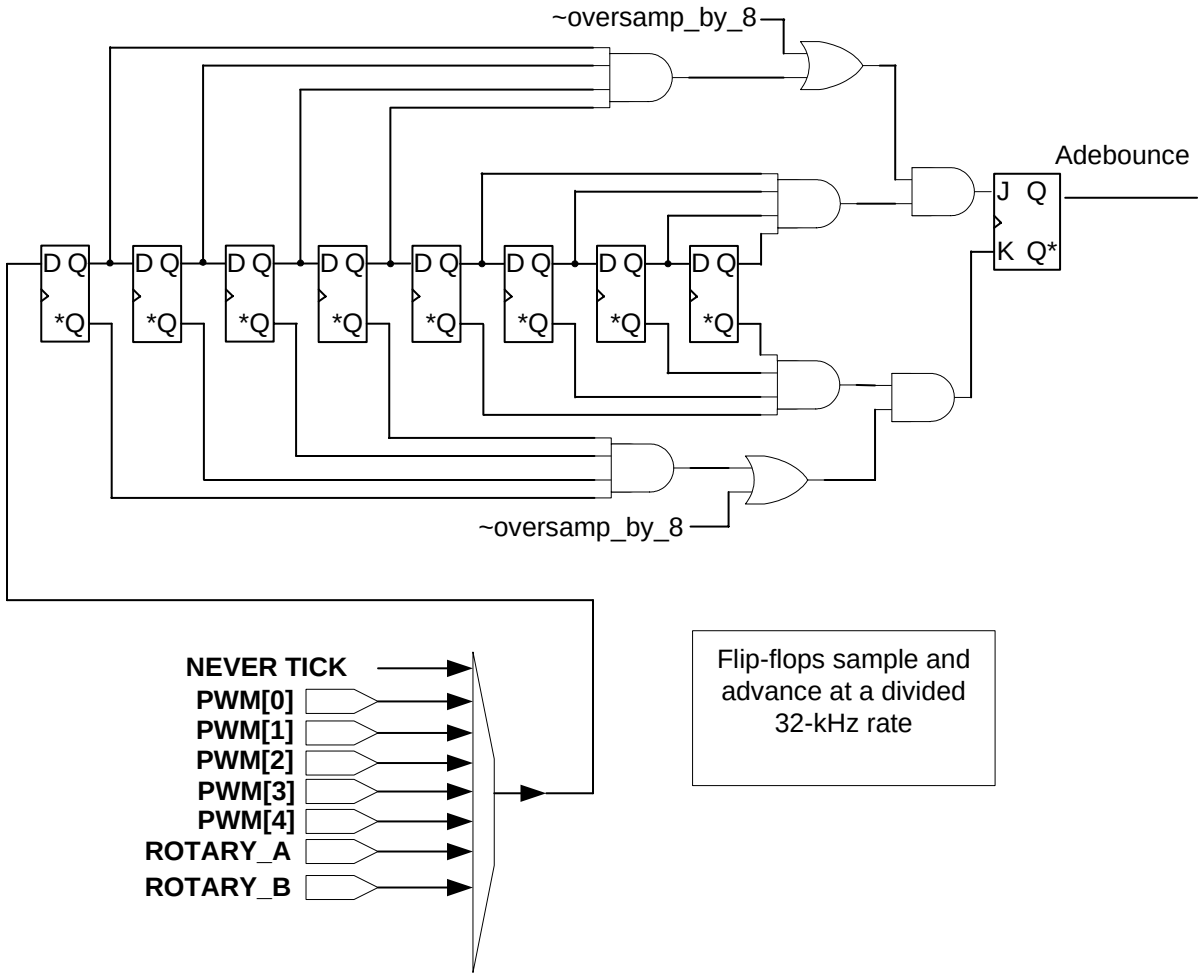


Figure 85. Rotary Decoding Mode—Debouncing Rotary A and B Inputs

A rotary decoder transition-following state machine is provided to detect the direction of rotation and the time at which to increment or decrement the 16-bit signed counter in HW_TIMROT_ROTCOUNT. The updown counter can be treated as either a relative count or an absolute count, depending on the state of the HW_TIMROT_ROTCTRL_RELATIVE bit. When set to the relative mode, each read of the counter has the side effect of resetting it. The edge detectors respond to both edges of each input to determine the self-timed transition inputs to the state machine (see Figure 86).

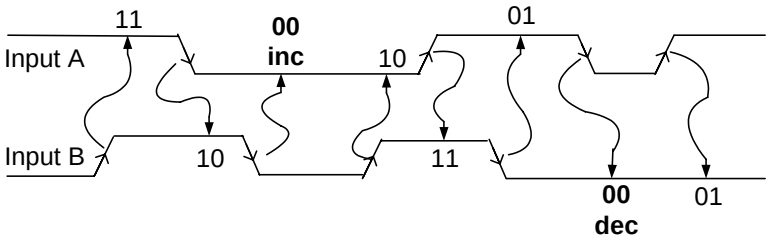


Figure 86. Rotary Decoding Mode—Input Transitions

Figure 86 shows that each detected edge causes a transition in the decoder state machine. Not all transitions are legal (see Table 656). For example, there is no legal way to transition directly from state 11 to 00 using normal inputs. In the cases where this occurs, the state machine goes to an alternate set of states and follows the input sequence until a valid sequence leading to state 00 is detected. No increment or decrement action is taken from the alternate state sequence.

Table 656. Rotary Decoder State Machine Transitions

CURRENT STATE	"INPUT" BA=00	"INPUT" BA=01	"INPUT" BA=10	"INPUT" BA=11
00	00	01	10	error
01	00, dec	01	error	11
10	00, inc	error	10	11
11	error	01	10	11

18.3.1. Testing the Rotary Decoder

To test the rotary decoder, select PWM1 and PWM2 as inputs to ROTARYA and ROTARYB. Since PWM1 and PWM2 can be started with known phase offsets and duty cycles, a continuous increment or decrement stream can be generated. Since PWM1 and PWM2 can be used as GPIO devices, the final part of the test is to generate and test a sequence of clockwise and counter-clockwise rotations to cover the entire state machine transitions, including the error conditions.

18.3.2. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

18.4. Programmable Registers

The following registers describe the programming interface for the timers and the rotary decoder.

18.4.1. Rotary Decoder Control Register Description

The Rotary Decoder Control Register specifies the reset state and the source selection for the rotary decoder. In addition, it specifies the polarity of any external input source that is used. This register also contains some general block controls including soft reset, clock gate, and present bits.

HW_TIMROT_ROTCTRL 0x80068000
HW_TIMROT_ROTCTRL_SET 0x80068004
HW_TIMROT_ROTCTRL_CLR 0x80068008
HW_TIMROT_ROTCTRL_TOG 0x8006800C

Table 657. HW_TIMROT_ROTCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	ROTARY_PRESENT	TIM3_PRESENT	TIM2_PRESENT	TIM1_PRESENT	TIM0_PRESENT	STATE			DIVIDER						RSRVD3			RELATIVE	OVERSAMPLE		POLARITY_B	POLARITY_A	RSRVD2	SELECT_B				RSRVD1	SELECT_A	

Table 658. HW_TIMROT_ROTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	This bit must be set to zero to enable operation of any timer or the rotary decoder. When set to one, it forces a block-level reset and gates off the clocks to the block.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29	**ROTARY_PRESENT**	RO	0x1	0= Rotary decoder is not present in this product. 1= Rotary decoder is present is in this product.
28	**TIM3_PRESENT**	RO	0x1	0= Timer 3 is not present in this product. 1= TIMER3 is present is in this product.
27	**TIM2_PRESENT**	RO	0x1	0= Timer 2 is not present in this product. 1= TIMER2 is present is in this product.
26	**TIM1_PRESENT**	RO	0x1	0= Timer 1 is not present in this product. 1= TIMER1 is present is in this product.
25	**TIM0_PRESENT**	RO	0x1	0= Timer 0 is not present in this product. 1= TIMER0 is present is in this product.
24:22	**STATE**	RO	0x0	Read-only view of the rotary decoder transition detecting state machine.

Table 658. HW_TIMROT_ROTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:16	**DIVIDER**	RW	0x0	This bit field determines the divisor used to divide the 32-kHz on chip clock rate for oversampling (debouncing) the rotary A and B inputs. Note that the divider value is actually the (value of this field+1).
15:13	**RSRVD3**	RO	0x0	Always write zeroes to this bit field.
12	**RELATIVE**	RW	0x0	Set this bit to one to cause the rotary decoders updown counter to be reset to zero whenever it is read.
11:10	**OVERSAMPLE**	RW	0x0	This bit field determines the oversample rate to use in debouncing Rotary A and B inputs. 8X = 0x0 8x Oversample: 8 successive ones or zeroes to transition. 4X = 0x1 4x Oversample: 4 successive ones or zeroes to transition. 2X = 0x2 2x Oversample: 2 successive ones or zeroes to transition. 1X = 0x3 1x Oversample: Transition on each first input change.
9	**POLARITY_B**	RW	0x0	Set this bit to one to invert the input to the edge detector.
8	**POLARITY_A**	RW	0x0	Set this bit to one to invert the input to the edge detector.
7	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
6:4	**SELECT_B**	RW	0x0	Selects the source for the timer "tick" that increments the free-running counter that measures the A2B and B2A overlap counts. NEVER_TICK = 0x0 SelectB: Never tick. PWM0 = 0x1 SelectB: Input from PWM0. PWM1 = 0x2 SelectB: Input from PWM1. PWM2 = 0x3 SelectB: Input from PWM2. PWM3 = 0x4 SelectB: Input from PWM3. PWM4 = 0x5 SelectB: Input from PWM4. ROTARYA = 0x6 SelectB: Input from Rotary A. ROTARYB = 0x7 SelectB: Input from Rotary B.
3	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
2:0	**SELECT_A**	RW	0x0	Selects the source for the timer "tick" that increments the free-running counter that measures the A2B and B2A overlap counts. NEVER_TICK = 0x0 SelectA: Never tick. PWM0 = 0x1 SelectA: Input from PWM0. PWM1 = 0x2 SelectA: Input from PWM1. PWM2 = 0x3 SelectA: Input from PWM2. PWM3 = 0x4 SelectA: Input from PWM3. PWM4 = 0x5 SelectA: Input from PWM4. ROTARYA = 0x6 SelectA: Input from Rotary A. ROTARYB = 0x7 SelectA: Input from Rotary B.

DESCRIPTION:

This register contains control parameters to specify the rotary decoder setup. It also contains some general block controls including soft reset, clock gate, and present bits.

EXAMPLE:

```
Empty Example.
```

18.4.2. Rotary Decoder Up/Down Counter Register Description

The Rotary Decoder Up/Down Counter Register contains the timer counter value that counts up or down as the rotary encoder is rotated.

HW_TIMROT_ROTCOUNT 0x80068010

Table 659. HW_TIMROT_ROTCOUNT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																UPDOWN															

Table 660. HW_TIMROT_ROTCOUNT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
15:0	**UPDOWN**	RO	0x00	At each edge of the Rotary A input, the Rotary B value is sampled, similarly at each edge of the Rotary B input, the Rotary A input is sampled. These values drive a rotary decoder state machine that determines when this counter is incremented or decremetned. When set in the RELATIVE mode, reads from this register clear this register as a side effect. Counter values in this register are signed 16-bit values.

DESCRIPTION:

This register contains the read-only current count for the rotary decoder.

EXAMPLE:

```
Empty Example.
```

18.4.3. Timer 0 Control and Status Register Description

The Timer 0 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 0.

HW_TIMROT_TIMCTRL0 0x80068020
HW_TIMROT_TIMCTRL0_SET 0x80068024
HW_TIMROT_TIMCTRL0_CLR 0x80068028
HW_TIMROT_TIMCTRL0_TOG 0x8006802C

Table 661. HW_TIMROT_TIMCTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2																IRQ	IRQ_EN	RSRVD1					POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			

Table 662. HW_TIMROT_TIMCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
15	**IRQ**	RW	0x0	This bit is set to one when Timer 0 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14	**IRQ_EN**	RW	0x0	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13:9	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
8	**POLARITY**	RW	0x0	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7	**UPDATE**	RW	0x0	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6	**RELOAD**	RW	0x0	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.
5:4	**PRESCALE**	RW	0x0	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. DIV_BY_1 = 0x0 PreScale: Divide the APBX clock by 1. DIV_BY_2 = 0x1 PreScale: Divide the APBX clock by 2. DIV_BY_4 = 0x2 PreScale: Divide the APBX clock by 4. DIV_BY_8 = 0x3 PreScale: Divide the APBX clock by 8.
3:0	**SELECT**	RW	0x0	Selects the source for the timer "tick" that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. NEVER_TICK = 0x0 Never tick. PWM0 = 0x1 Input from PWM0. PWM1 = 0x2 Input from PWM1. PWM2 = 0x3 Input from PWM2. PWM3 = 0x4 Input from PWM3. PWM4 = 0x5 Input from PWM4. ROTARYA = 0x6 Input from Rotary A. ROTARYB = 0x7 Input from Rotary B. 32KHZ_XTAL = 0x8 Input from 32-kHz crystal. 8KHZ_XTAL = 0x9 Input from 8-kHz (divided from 32-kHz crystal). 4KHZ_XTAL = 0xA Input from 4-kHz (divided from 32-kHz crystal). 1KHZ_XTAL = 0xB Input from 1-kHz (divided from 32-kHz crystal). TICK_ALWAYS = 0xC Always tick.

DESCRIPTION:

This control register specifies control parameters, as well as interrupt status and the enable for Timer 0.

EXAMPLE:

```
Empty Example.
```

18.4.4. Timer 0 Count Register Description

The Timer 0 Count Register contains the timer counter values for Timer 0.

HW_TIMROT_TIMCOUNT0 0x80068030

Table 663. HW_TIMROT_TIMCOUNT0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RUNNING_COUNT																FIXED_COUNT															

Table 664. HW_TIMROT_TIMCOUNT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RUNNING_COUNT**	RO	0x00	This bit field shows the current state of the running count as it decrements.
15:0	**FIXED_COUNT**	RW	0x00	Software loads the fixed count bit field with the value to count down. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count.

DESCRIPTION:

This timer count register contains the programable and readback counter values for Timer 0.

EXAMPLE:

```
Empty Example.
```

18.4.5. Timer 1 Control and Status Register Description

The Timer 1 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 1.

HW_TIMROT_TIMCTRL1 0x80068040
HW_TIMROT_TIMCTRL1_SET 0x80068044
HW_TIMROT_TIMCTRL1_CLR 0x80068048
HW_TIMROT_TIMCTRL1_TOG 0x8006804C

Table 665. HW_TIMROT_TIMCTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2																IRQ	IRQ_EN	RSRVD1					POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			

Table 666. HW_TIMROT_TIMCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
15	**IRQ**	RW	0x0	This bit is set to one when Timer 1 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14	**IRQ_EN**	RW	0x0	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13:9	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
8	**POLARITY**	RW	0x0	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7	**UPDATE**	RW	0x0	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6	**RELOAD**	RW	0x0	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.

Table 666. HW_TIMROT_TIMCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
5:4	**PRESCALE**	RW	0x0	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. DIV_BY_1 = 0x0 PreScale: Divide the APBX clock by 1. DIV_BY_2 = 0x1 PreScale: Divide the APBX clock by 2. DIV_BY_4 = 0x2 PreScale: Divide the APBX clock by 4. DIV_BY_8 = 0x3 PreScale: Divide the APBX clock by 8.
3:0	**SELECT**	RW	0x0	Selects the source for the timer "tick" that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. NEVER_TICK = 0x0 Never tick. PWM0 = 0x1 Input from PWM0. PWM1 = 0x2 Input from PWM1. PWM2 = 0x3 Input from PWM2. PWM3 = 0x4 Input from PWM3. PWM4 = 0x5 Input from PWM4. ROTARYA = 0x6 Input from Rotary A. ROTARYB = 0x7 Input from Rotary B. 32KHZ_XTAL = 0x8 Input from 32-kHz crystal. 8KHZ_XTAL = 0x9 Input from 8 kHz (divided from 32-kHz crystal). 4KHZ_XTAL = 0xA Input from 4 kHz (divided from 32-kHz crystal). 1KHZ_XTAL = 0xB Input from 1 kHz (divided from 32-kHz crystal). TICK_ALWAYS = 0xC Always tick.

DESCRIPTION:

This control register specifies control parameters, as well as interrupt status and the enable for Timer 1.

EXAMPLE:

```
Empty Example.
```

18.4.6. *Timer 1 Count Register Description*

The Timer 1 Count Register contains the timer counter values for Timer 1.

HW_TIMROT_TIMCOUNT1 0x80068050

Table 667. HW_TIMROT_TIMCOUNT1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RUNNING_COUNT																FIXED_COUNT															

Table 668. HW_TIMROT_TIMCOUNT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RUNNING_COUNT**	RO	0x00	This bit field shows the current state of the running count as it decrements.
15:0	**FIXED_COUNT**	RW	0x00	Software loads the fixed count bit field with the value to count down. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count.

DESCRIPTION:

This timer count register contains the programable and readback counter values for Timer 1.

EXAMPLE:

```
Empty Example.
```

18.4.7. Timer 2 Control and Status Register Description

The Timer 2 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 2.

HW_TIMROT_TIMCTRL2 0x80068060
HW_TIMROT_TIMCTRL2_SET 0x80068064
HW_TIMROT_TIMCTRL2_CLR 0x80068068
HW_TIMROT_TIMCTRL2_TOG 0x8006806C

Table 669. HW_TIMROT_TIMCTRL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2																IRQ	IRQ_EN	RSRVD1					POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			

Table 670. HW_TIMROT_TIMCTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
15	**IRQ**	RW	0x0	This bit is set to one when Timer 2 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14	**IRQ_EN**	RW	0x0	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13:9	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.

Table 670. HW_TIMROT_TIMCTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**POLARITY**	RW	0x0	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7	**UPDATE**	RW	0x0	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6	**RELOAD**	RW	0x0	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.
5:4	**PRESCALE**	RW	0x0	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. DIV_BY_1 = 0x0 PreScale: Divide the APBX clock by 1. DIV_BY_2 = 0x1 PreScale: Divide the APBX clock by 2. DIV_BY_4 = 0x2 PreScale: Divide the APBX clock by 4. DIV_BY_8 = 0x3 PreScale: Divide the APBX clock by 8.
3:0	**SELECT**	RW	0x0	Selects the source for the timer "tick" that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. NEVER_TICK = 0x0 Never tick. PWM0 = 0x1 Input from PWM0. PWM1 = 0x2 Input from PWM1. PWM2 = 0x3 Input from PWM2. PWM3 = 0x4 Input from PWM3. PWM4 = 0x5 Input from PWM4. ROTARYA = 0x6 Input from Rotary A. ROTARYB = 0x7 Input from Rotary B. 32KHZ_XTAL = 0x8 Input from 32-kHz crystal. 8KHZ_XTAL = 0x9 Input from 8 kHz (divided from 32-kHz crystal). 4KHZ_XTAL = 0xA Input from 4 kHz (divided from 32-kHz crystal). 1KHZ_XTAL = 0xB Input from 1 kHz (divided from 32-kHz crystal). TICK_ALWAYS = 0xC Always tick.

DESCRIPTION:

This control register specifies control parameters as well as interrupt status and the enable for Timer 2.

EXAMPLE:

```
Empty Example.
```

18.4.8. Timer 2 Count Register Description

The Timer 2 Count Register contains the timer counter values for Timer 2.

HW_TIMROT_TIMCOUNT2 0x80068070

Table 671. HW_TIMROT_TIMCOUNT2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RUNNING_COUNT																FIXED_COUNT															

Table 672. HW_TIMROT_TIMCOUNT2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**RUNNING_COUNT**	RO	0x00	This bit field shows the current state of the running count as it decrements.
15:0	**FIXED_COUNT**	RW	0x00	Software loads the fixed count bit field with the value to count down. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count.

DESCRIPTION:

This timer count register contains the programable and readback counter values for Timer 2.

EXAMPLE:

```
Empty Example.
```

18.4.9. Timer 3 Control and Status Register Description

The Timer 3 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 3.

HW_TIMROT_TIMCTRL3 0x80068080
HW_TIMROT_TIMCTRL3_SET 0x80068084
HW_TIMROT_TIMCTRL3_CLR 0x80068088
HW_TIMROT_TIMCTRL3_TOG 0x8006808C

Table 673. HW_TIMROT_TIMCTRL3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2												TEST_SIGNAL				IRQ	IRQ_EN	RSRVD1			DUTY_VALID	DUTY_CYCLE	POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			

Table 674. HW_TIMROT_TIMCTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
19:16	**TEST_SIGNAL**	RW	0x0	Selects the source of the signal to be measured in duty cycle mode. NEVER_TICK = 0x0 Never tick. Freeze the count. PWM0 = 0x1 Input from PWM0. PWM1 = 0x2 Input from PWM1. PWM2 = 0x3 Input from PWM2. PWM3 = 0x4 Input from PWM3. PWM4 = 0x5 Input from PWM4. ROTARYA = 0x6 Input from Rotary A. ROTARYB = 0x7 Input from Rotary B. 32KHZ_XTAL = 0x8 Input from 32-kHz crystal. 8KHZ_XTAL = 0x9 Input from 8 kHz (divided from 32-kHz crystal). 4KHZ_XTAL = 0xA Input from 4 kHz (divided from 32-kHz crystal). 1KHZ_XTAL = 0xB Input from 1 kHz (divided from 32-kHz crystal). TICK_ALWAYS = 0xC Always tick.
15	**IRQ**	RW	0x0	This bit is set to one when Timer 3 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14	**IRQ_EN**	RW	0x0	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13:11	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
10	**DUTY_VALID**	RO	0x0	This bit is set and cleared by the hardware. It is set only when in duty cycle measuring mode and the HW_TIMROT_TIMCOUNT3 has valid duty cycle data to be read. This register will be cleared if not in duty cycle mode or on writes to this register. In the case that it is written while in duty cycle mode, this bit will clear but will again be set at the appropriate time for reading the count register.
9	**DUTY_CYCLE**	RW	0x0	Set this bit to one to cause the timer to operate in duty cycle measuring mode.
8	**POLARITY**	RW	0x0	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7	**UPDATE**	RW	0x0	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.

Table 674. HW_TIMROT_TIMCTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	**RELOAD**	RW	0x0	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.
5:4	**PRESCALE**	RW	0x0	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. DIV_BY_1 = 0x0 PreScale: Divide the APBX clock by 1. DIV_BY_2 = 0x1 PreScale: Divide the APBX clock by 2. DIV_BY_4 = 0x2 PreScale: Divide the APBX clock by 4. DIV_BY_8 = 0x3 PreScale: Divide the APBX clock by 8.
3:0	**SELECT**	RW	0x0	Selects the source for the timer "tick" that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. In duty cycle mode it increments the counter used to calculate the high and low cycle counts. NEVER_TICK = 0x0 Never tick. Freeze the count. PWM0 = 0x1 Input from PWM0. PWM1 = 0x2 Input from PWM1. PWM2 = 0x3 Input from PWM2. PWM3 = 0x4 Input from PWM3. PWM4 = 0x5 Input from PWM4. ROTARYA = 0x6 Input from Rotary A. ROTARYB = 0x7 Input from Rotary B. 32KHZ_XTAL = 0x8 Input from 32-kHz crystal. 8KHZ_XTAL = 0x9 Input from 8 kHz (divided from 32-kHz crystal). 4KHZ_XTAL = 0xA Input from 4 kHz (divided from 32-kHz crystal). 1KHZ_XTAL = 0xB Input from 1 kHz (divided from 32-kHz crystal). TICK_ALWAYS = 0xC Always tick.

DESCRIPTION:

This control register specifies control parameters, as well as interrupt status and the enable for Timer 3.

EXAMPLE:

```
Empty Example.
```

18.4.10. Timer 3 Count Register Description

The Timer 3 Count Register contains the timer counter values for Timer 3. NOTE: This timer can be put in a special duty cycle mode that will measure the duty cycle of an input test signal.

HW_TIMROT_TIMCOUNT3 0x80068090

Table 675. HW_TIMROT_TIMCOUNT3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
LOW_RUNNING_COUNT																HIGH_FIXED_COUNT															

Table 676. HW_TIMROT_TIMCOUNT3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**LOW_RUNNING_COUNT**	RO	0x00	In duty cycle mode, this bit field is loaded from the running counter when it has just finished measuring the low portion of the duty cycle. In normal timer mode, it shows the running count as a read-only value.
15:0	**HIGH_FIXED_COUNT**	RW	0x00	Software loads the fixed count bit field with the value to count down. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count. In duty cycle mode, this bit field is loaded from the running counter when it has finished measuring the high portion of the duty cycle.

DESCRIPTION:

This timer count register contains the programable and readback counter values for Timer 3. The definitions of the fields change depending whether the timer is in normal or duty cycle mode.

EXAMPLE:

```
Empty Example.
```

TIMROT XML Revision: 1.39

19. REAL-TIME CLOCK, ALARM, WATCHDOG, AND PERSISTENT BITS

This chapter describes the real-time clock, alarm clock, watchdog reset, persistent bits, and millisecond counter included on the STMP36xx. Programmable registers are described in Section 19.7.

19.1. Overview

The real-time clock (RTC), alarm, watchdog reset, and persistent bits share a common source of one-millisecond and one-second time pulses and utilize persistent storage when the chip is in its powered-down state. Figure 87 illustrates this block.

NOTE: The term *power-down*, as used here, refers to a state in which the DC-DC converter and various parts of the crystal power domain are still powered up, but the rest of the chip is powered down. If the battery is removed, then the persistent bits, the alarm value, and the second counter value will be lost. The *crystal power domain* powers both the 32-kHz and 24-MHz crystals.

Upon battery insertion, the crystals (32-kHz and 24-MHz) are in a quiescent state. Whether and when either or both of these crystals are activated is under software control through the *RTC persistent bits*, as described later in this chapter. Moreover, whether either or both of the crystals remain active during a power-down state is similarly controlled by software.

The one-second time base is derived either from the 24.0-MHz crystal oscillator or the 32.768-kHz crystal oscillator, as controlled by the value of the corresponding bit in Persistent Register 0. The time base thus generated is used to increment the value of the persistent seconds count register. Like the values of the other persistent registers, the value of the persistent seconds count register is not lost across a power down state. Whether this register continues to count seconds through a power down state or simply retains its value is under control of software.

Contrary to the one-second time base, no record or count is made of the one-millisecond time base in the crystal power domain. The one-millisecond time base is always derived from the 24.0-MHz crystal oscillator and is not available when the chip is powered down.

The real-time clock seconds counter, alarm functions, and persistent bit storage are kept in the crystal oscillator clock and power domain. Shadow versions of these values are maintained in the CPU's power and APBX clock domain when the chip is in a power-up state. When the chip transitions from power-off to power-on, the master values are copied to shadow values by the copy controller. Whenever software writes to a shadow register, then the copy controller copies the new value into the master register in the crystal oscillator power domain.

Some of the persistent bits are used to control features that can continue to operate after power-down, such as the second counter and the alarm function. Other persistent bits are available to store application state information over power-downs. 64-bits are used to hold the SRAM repair configuration. This value is computed immediately after battery insertion (cold start) and stored for use in each subsequent warm start.

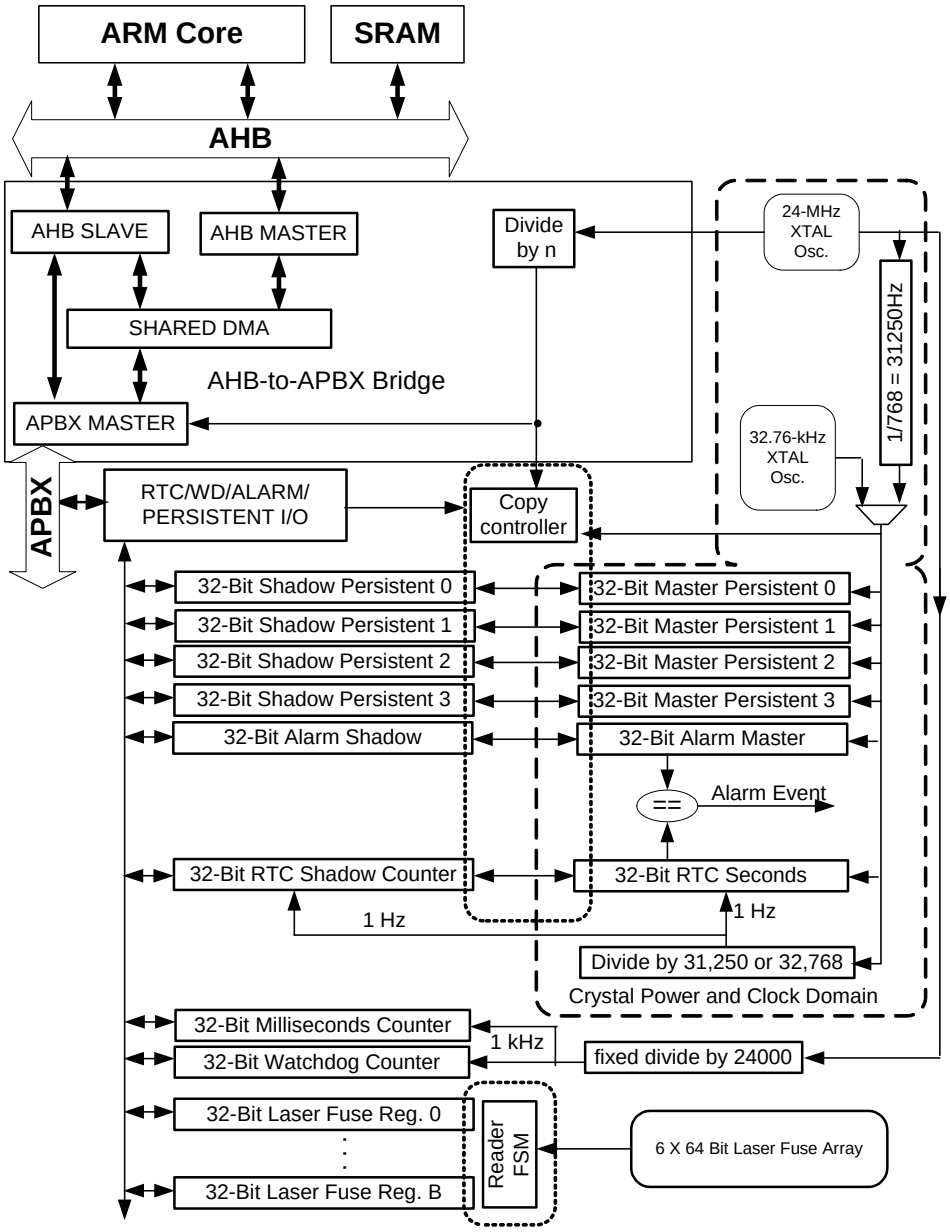


Figure 87. RTC, Watchdog, Alarm, and Persistent Bits Block Diagram

Immediately after reset, it can take several hundred clocks for the copy controller to complete the copy process from the analog domain to the digital domain. Software cannot rely on the contents of the seconds counter, alarm, or persistent bits until this copy is complete. Therefore, software must wait until all bits of interest in the HW_RTC_STAT_STALE_REGS field have been reset to zero by the copy controller before reading the initial state of these values (see Figure 88). Note that HW_RTC_PERSISTENT2 and HW_RTC_PERSISTENT3 are the first ones read by the copy controller, so that the SRAM configuration data is available first.

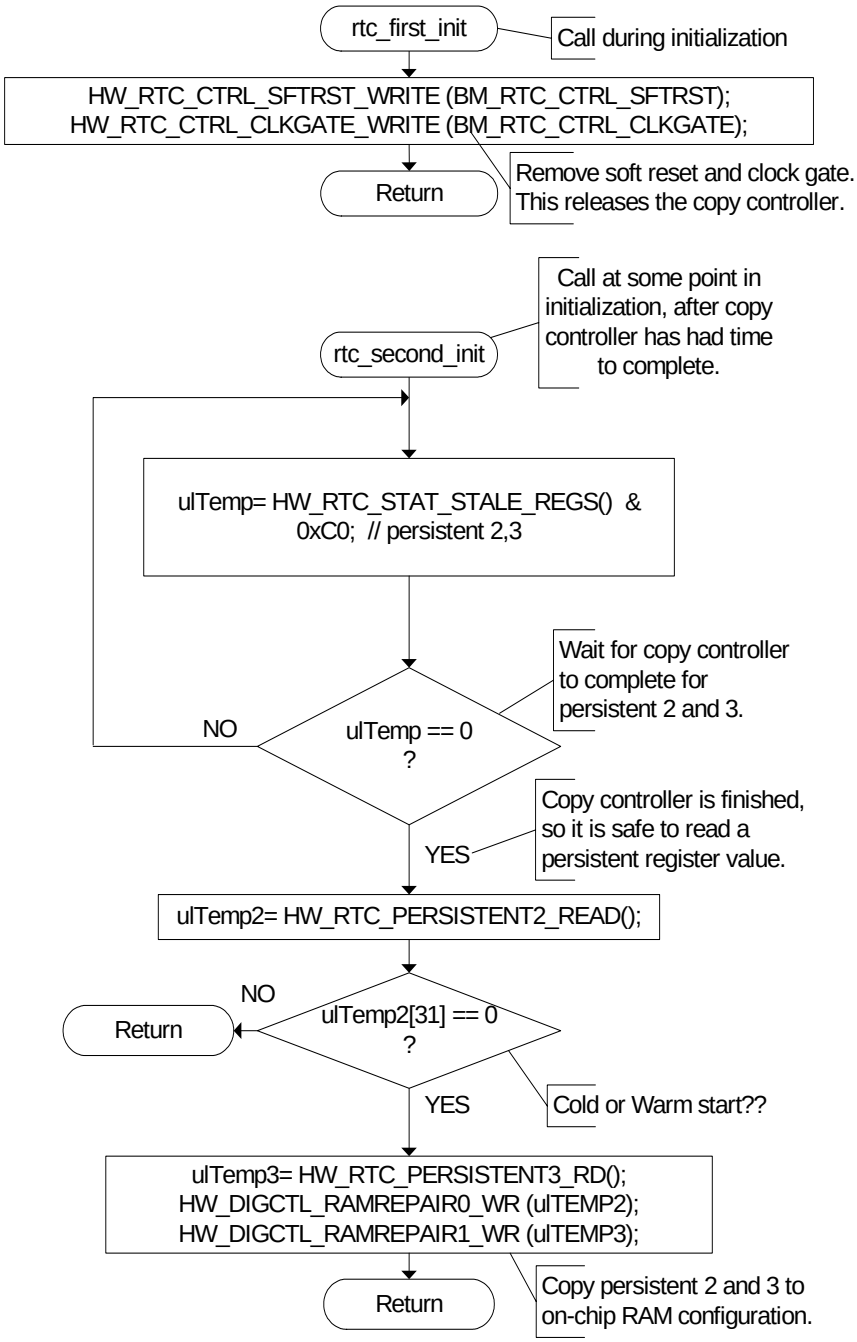


Figure 88. RTC Initialization Sequence

The RTC functions that are implemented in the crystal power domain are referred to as the RTC analog functions. These functions operate at 32.768 kHz, generated by the 32.768-kHz crystal oscillator when the clock source bit is set to one (HW_RTC_PERSISTENT0_CLOCK_SOURCE). When the clock source bit is set to zero, these functions operate on a clock domain derived from the 24.0-MHz crystal

oscillator divided by 768 to yield 31.250 kHz. Switching between these two clock domains is handled by a glitch-free clock mux. The 1-Hz time base is derived by dividing either 32.768 kHz by 32768 or by dividing 31.250 kHz by 31,250, controlled by the CLOCK_SOURCE bit. Note that the clock mux is only glitch-free when switching from the 24.0-MHz crystal to the 32.768-kHz crystal.

The automatic write-back that occurs for each register as the copy controller services writes to the shadow registers can lead to some very long timing loops if efficient write procedures are not used. Writing all six shadow registers can take up to 4 milliseconds to complete. A single word write can be transferred to the analog side of the RTC within 40 microseconds. Do not attempt to write to more than one shadow register immediately before power down.

Registers are copied as pairs to the RTC analog section. While Persistent registers 2 and 3 are marked as holding on-chip RAM configuration information, software is free to reorder the location where this data is retained. There are no hardwired uses for any of the bits of Persistent registers 1, 2, and 3. In addition there are no hardwired uses for the upper portion of Persistent Register 0, i.e., HW_RTC_PERSISTENT0[31:16]. The lower half of Persistent Register 0 has specific hardwired uses for each bit.

Figure 89 illustrates the timing of the analog/digital interface. Registers are read in pairs from the RTC analog section, starting with Persistent registers 2 and 3 (SRAM configuration information is needed very soon after power up), followed by the Alarm Register and Persistent Register 1, and finally the seconds counter and Persistent Register 0.

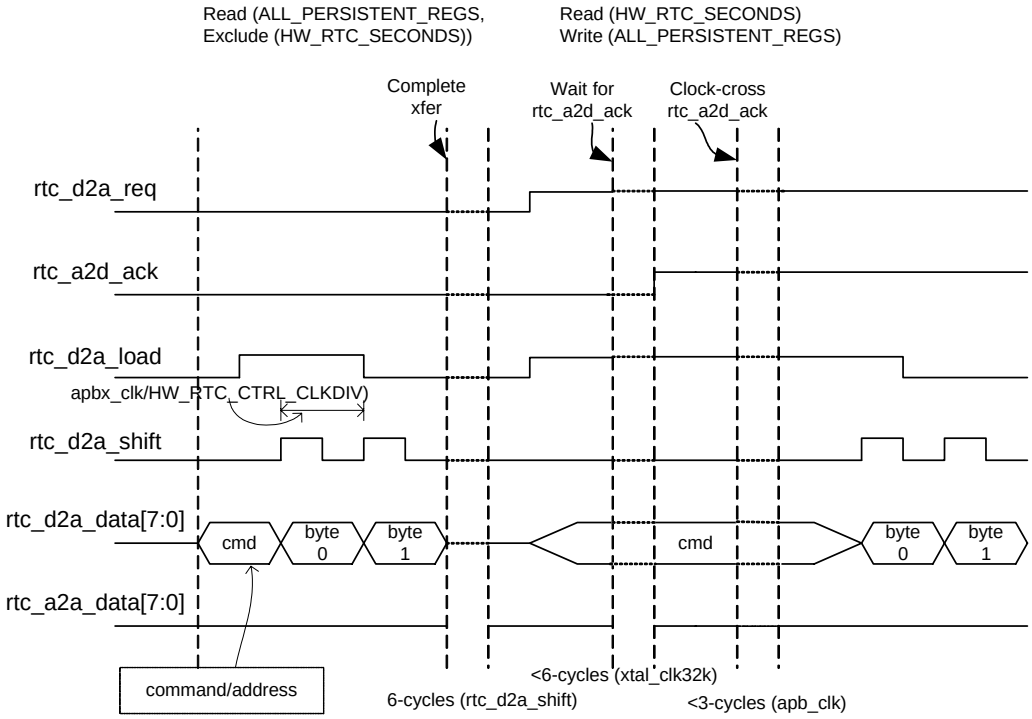


Figure 89. Analog/Digital Interface Timing

NOTE: Copying registers in pairs to and from the analog side is basically an implementation detail that has a minor effect on the time required to transfer a register write to the analog side.

Before a new value is written to a shadow register by the CPU, software must first confirm that the corresponding bit of HW_RTC_STAT_NEWREGS is a zero, as shown in Figure 90. This ensures that a value previously written to the register has been completely handled by the copy state machine. Failure to obey this constraint could cause a newer updated value to be lost.

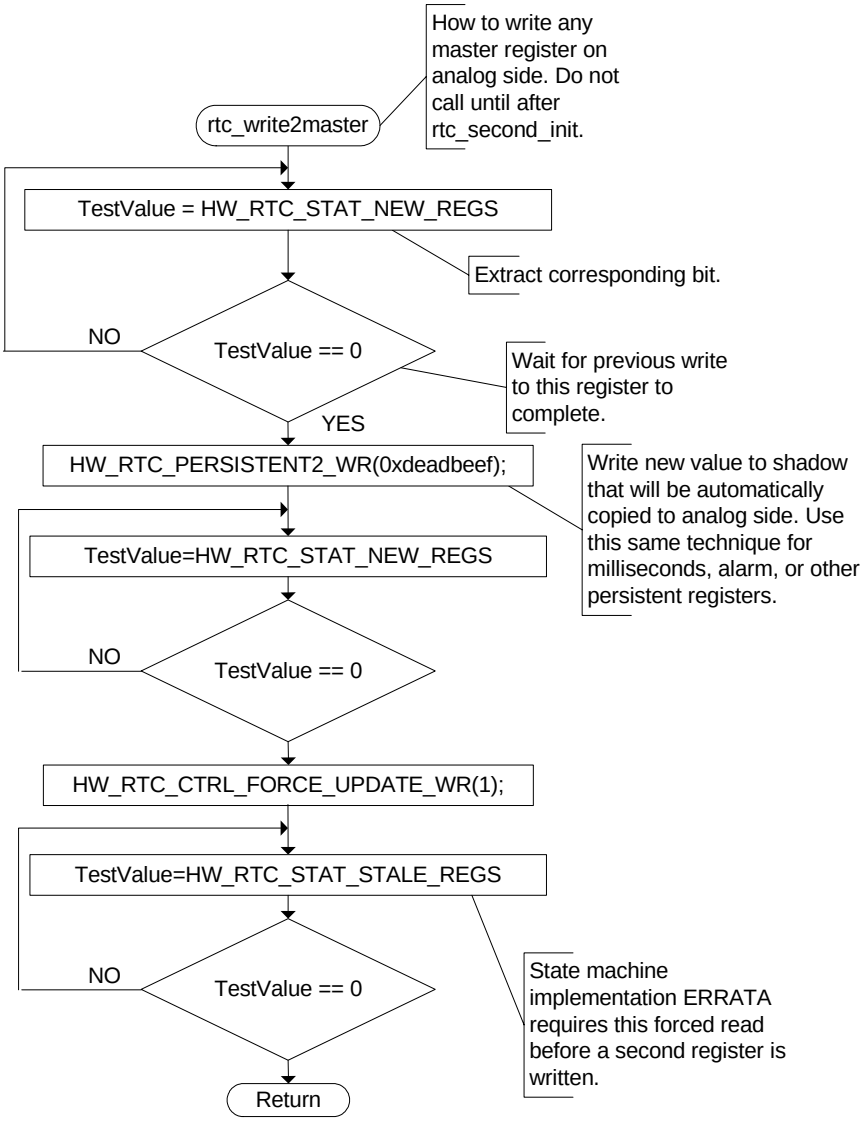


Figure 90. RTC Writing to a Master Register from CPU

19.2. Real-Time Clock

The real-time clock is a CPU-accessible, continuously-running 32-bit counter that increments every second and that can be derived from either the 24-MHz or the 32-MHz clock, as determined by a writable bit value in the RTC control register.

A 32-bit second counter has enough resolution to count up to 136 years with one-second increments. The RTC can continue to count time as long as a voltage is applied to the BATT pin, irrespective of whether the rest of the chip is powered up. The normal digital reset has no effect on the master RTC registers located in the crystal power and clock domain. A special first-power-on reset establishes the default value of the master RTC registers.

For consistency across applications, it is recommended that the second timer should be referenced to January 1, 1980 at a 32-bit value of zero (same epoch reference as PC) in applications that use it as a time-of-day clock. If the real-time clock function is not present on a specific chip, as indicated in the control and status register (HW_RTC_STAT_RTC_PRESENT), then no real-time epoch is maintained over power-down cycles.

19.2.1. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

19.3. Millisecond Resolution Timing Facility

A millisecond counter facility is provided based on a 1-kHz signal derived from either the 24-MHz clock. The count value is neither maintained nor incremented during power-down cycles. At each power-up, this register is set to its reset state. On each tick of the 1-kHz source, the milliseconds counter increments. With a 32-bit counter, a kernel can run up to 4,294,967,294 milliseconds or 49.7 days before it must deal with a counter wrap.

WARNING: When the 32.768-kHz crystal oscillator is selected as the source for the seconds counter, an anomaly is created between the time intervals of the millisecond counter and the seconds counter. That is, the manufacturing tolerance of the two crystals are such that 1000 millisecond counter increments are not exactly one second as measured by the real-time clock seconds counter.

19.4. Alarm Clock

The alarm clock function allows an application to specify a future instant at which the chip should be awakened, i.e., if powered down, it can be powered up and the CPU can be interrupted. The alarm clock setting is a CPU-accessible, 32-bit value that is continuously matched against the 32-bit real-time clock seconds counter. When the two values are equal, an alarm event is triggered. Persistent bits indicate whether an alarm event should power up the chip from its powered-down state. In addition to or instead of powering up the chip, the alarm event can also cause a CPU interrupt.

NOTE: If the alarm is set to power up the chip in the event of an alarm and such an event occurs, then the only record of the wake-up cause is located in the analog side. At power-up, the analog side registers are copied to the digital shadow registers and the alarm-wake bit is visible in the digital shadow register. If an alarm event

occurs that is not associated with a power up condition, the wake-up bit is only valid on the analog side. In this case, diagnostic software should force a copy from the analog side back to the digital shadow register before reading the ALARM_WAKE status bit.

19.5. Watchdog Reset Register

The watchdog reset is a CPU-configurable device. It is programmed by software to generate a chip wide reset after HW_RTC_WATCHDOG milliseconds. The module generates this reset if software does not rewrite this register before this time elapses. The watchdog timer decrements once for every tick of the 1-kHz clock supplied from the RTC analog section (see Figure 87). The reset generated by the watchdog timer has no effect on the values retained in the master registers of the real-time clock seconds counter, alarm, or persistent registers.

The watchdog timer is initially disabled and set to count 4,294,967,295 milliseconds before generating a watchdog reset.

The watchdog timer does not run when the chip is in its powered-down state. Therefore, there is no master/shadow register pairing for the watchdog timer. The watchdog timer must be “present” on an actual chip to perform this function (see the HW_RTC_STAT_WATCHDOG_PRESENT bit description).

19.6. Laser Fuse Bits

The STMP36xx contains 384 laser programmable fuse bits. These bits are programmed at the end of wafer processing and cannot be changed once the parts are packaged. Separate documentation describes the usage and mapping of laser fuse bits to functions. The laser fuse bits can be read from registers contained in this block, if unlocked. Refer to the register descriptions for information about unlocking and reading the laser fuse registers. Provided that neither HW_LASERFUSE8_BITS[31] nor HW_LASERFUSE8_BITS[30] is set to one, then the laser fuse registers can be written by software. Once either one of these bits is set, the laser fuse registers become strictly read-only.

Implementation Note: This is different from the way the STMP35xx behaves.

19.7. Programmable Registers

This section describes the programmable registers of the real-time clock, including the watchdog register, alarm register, laser fuse registers, and persistent registers.

19.7.1. Real-Time Clock Control Register Description

HW_RTC_CTRL is the control register for the real-time clock, alarm, and watchdog timer.

```
HW_RTC_CTRL       0x8005C000
HW_RTC_CTRL_SET 0x8005C004
HW_RTC_CTRL_CLR 0x8005C008
HW_RTC_CTRL_TOG 0x8005C00C
```

Table 677. HW_RTC_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSVD2		CLKDIV				RSVD1																	SUPPRESS_COPY2ANALOG	FORCE_UPDATE	WATCHDOGEN	ONEMSEC_IRQ	ALARM_IRQ	ONEMSEC_IRQ_EN	ALARM_IRQ_EN

Table 678. HW_RTC_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	1= Hold real-time clock digital side in soft reset state. This bit has no effect on the RTC analog section.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block. This bit has no effect on the RTC analog section.
29:28	**RSVD2**	RO	0x0	Reserved, write only zeroes.
27:24	**CLKDIV**	RW	0x2	Sets the APBX clock divisor used to generate the RTC analog/digital interface clock. 0,1: Interface clock is disabled. 2-15: Interface clock divisor.
23:7	**RSVD1**	RO	0x0	Reserved, write only zeroes.
6	**SUPPRESS_COPY2ANALOG**	RW	0x0	This bit is used for diagnostic purposes. 1= Suppress the automatic copy that normally occurs to the analog side, whenever a shadow register is written. 0= Normal operation. Use SCT writes to set clear or toggle. NORMAL = 0x0 Data written to shadow registers is automatically copied to the analog side. NO_COPY = 0x1 Suppress the automatic copying of write data to the analog side.
5	**FORCE_UPDATE**	RW	0x0	This bit is used for diagnostic purposes. 1= Force Analog Side Update. 0 = Normal Operation. Use SCT writes to set clear or toggle. As long as this bit is set, the copy controller will attempt to copy all six registers from the analog side to the digital side. Software should set this bit, then reset it as soon as practical. Then software should poll the HW_RTC_STAT_STALE bit field until it goes to zero. NORMAL = 0x0 Stale data on the anlog side is copied to the shadows as appropriate. FORCE_COPY = 0x1 Force automatic copying of write data from the analog side to the shadow registers.
4	**WATCHDOGEN**	RW	0x0	1= Enable Watchdog Timer to force chip wide resets. Use SCT writes to set clear or toggle.

Table 678. HW_RTC_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**ONEMSEC_IRQ**	RW	0x0	1= One-Millisecond Interrupt Request Status. Use SCT writes to clear this interrupt status bit.
2	**ALARM_IRQ**	RW	0x0	1= Alarm Interrupt Status. Use SCT writes to clear this interrupt status bit.
1	**ONEMSEC_IRQ_EN**	RW	0x0	1= Enable One-Millisecond Interrupt. Use SCT writes to set clear or toggle.
0	**ALARM_IRQ_EN**	RW	0x0	1= Enable Alarm Interrupt. Use SCT writes to set clear or toggle.

DESCRIPTION:

The contents of this register control the operation of the RTC portions implemented as an APBX peripheral running in the APBX clock domain. These functions operate only when the chip is in its full power-up state.

EXAMPLE:

```
HW_RTC_CTRL_CLR(BM_RTC_CTRL_SFTRST); // remove the soft reset condition
HW_RTC_CTRL_CLR(BM_RTC_CTRL_CLKGATE); // enable clocks within the RTC
while(HW_RTC_STAT.STALE_REGS !=0)
{
printf(" something is stale in one of the digital side registers

// the copy controller will copy analog registers to digtial registers as required,
// turning off staleregs bits as it goes about its business.
}
if(HW_RTC_STAT.WATCHDOG_PRESENT != 0) // then you can use the watchdog timer on this chip
```

19.7.2. Real-Time Clock Status Register Description

HW_RTC_STAT is the status register for the real-time clock, alarm, and watchdog timer.

HW_RTC_STAT 0x8005C010
HW_RTC_STAT_SET 0x8005C014
HW_RTC_STAT_CLR 0x8005C018
HW_RTC_STAT_TOG 0x8005C01C

Table 679. HW_RTC_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RTC_PRESENT	ALARM_PRESENT	WATCHDOG_PRESENT	XTAL32768_PRESENT	RSVD3						STALE_REGS						RSVD2		NEW_REGS						RSVD1						FUSE_UNLOCK	FUSE_DONE

Table 680. HW_RTC_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RTC_PRESENT**	RO	0x1	This read-only bit reads back a one if the RTC is present in the device.
30	**ALARM_PRESENT**	RO	0x1	This read-only bit reads back a one if the Alarm function is present in the device.
29	**WATCHDOG_PRESENT**	RO	0x1	This read-only bit reads back a one if the Watchdog Timer function is present in the device.
28	**XTAL32768_PRESENT**	RO	0x1	This read-only bit reads back a one if the 32.768-kHz crystal oscillator function is present in the device.
27:22	**RSVD3**	RO	0x0	Reserved, write only zeroes.
21:16	**STALE_REGS**	RO	0x3F	These read-only bits are set to one whenever the corresponding shadow register contents are older than the analog side contents. These bits are set by reset and cleared by the copy controller. They are also set by writing a one to the FORCE_UPDATE bit.
15:14	**RSVD2**	RO	0x0	Reserved, write zeroes only.
13:8	**NEW_REGS**	RO	0x00	These read-only bits are set to one whenever the corresponding shadow register contents are newer than the analog side contents. These bits are set by writing to the corresponding register and cleared by the copy controller.
7:2	**RSVD1**	RO	0x0	Reserved, write zeroes only.
1	**FUSE_UNLOCK**	RO	0x0	This read-only bit reads back a one if the laser fuse registers are unlocked.
0	**FUSE_DONE**	RO	0x1	Reflects the state of the laser fuse reader. 1=Laser fuse reader has finished.

DESCRIPTION:

The contents of this register control the operation of the portions of the RTC that are implemented as an APBX peripheral running in the APBX clock domain. These functions operate only when the chip is in its full power-up state.

EXAMPLE:

```
  HW_RTC_CTRL_CLR(BM_RTC_CTRL_SFTRST); // remove the soft reset condition
  HW_RTC_CTRL_CLR(BM_RTC_CTRL_CLKGATE); // enable clocks within the RTC
  HW_RTC_CTRL_CLR(BM_RTC_CTRL_ALARM_IRQ); // reset the alarm interrupt by clearing its sta-
tus bit
  while(HW_RTC_STAT.STALE_REGS !=0)
  {
  printf(" something is stale in one of the digital side registers

  // the copy controller will copy analog registers to digtial registers as required,
  // turning off staleregs bits as it goes about its business.
  }
```

19.7.3. Real-Time Clock Milliseconds Counter Description

The Real-Time Clock Milliseconds Counter Register provides a reliable elapsed time reference to the kernel with millisecond resolution.

HW_RTC_MILLISECONDS 0x8005C020
HW_RTC_MILLISECONDS_SET 0x8005C024
HW_RTC_MILLISECONDS_CLR 0x8005C028

HW_RTC_MILLISECONDS_TOG 0x8005C02C

Table 681. HW_RTC_MILLISECONDS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 682. HW_RTC_MILLISECONDS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x00000000	32-bit milliseconds counter.

DESCRIPTION:

HW_RTC_MSECONDS provides access to the 32-bit milliseconds counter. This counter is not a shadow register, i.e., the contents of this register are not preserved over power-down states. This counter increments once per millisecond from a clock source derived from the 24.0-MHz crystal clock. This 1-kHz source does not vary as the APBX clock frequency is changed.

The millisecond counter wraps at 4,294,967,294 milliseconds or 49.7 days.

EXAMPLE:

```
HW_RTC_MILLISECONDS_WR(0); // write an initial starting value to the milliseconds counter
Count = HW_RTC_MILLISECONDS_RD(); // read the current value of the milliseconds counter.
```

19.7.4. Real-Time Clock Seconds Counter Register Description

The Real-Time Clock Seconds Counter Register is used to maintain real time for applications, even across certain chip power-down states.

HW_RTC_SECONDS 0x8005C030
HW_RTC_SECONDS_SET 0x8005C034
HW_RTC_SECONDS_CLR 0x8005C038
HW_RTC_SECONDS_TOG 0x8005C03C

Table 683. HW_RTC_SECONDS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 684. HW_RTC_SECONDS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0x00000000	Increments once per second.

DESCRIPTION:

HW_RTC_SECONDS provides access to the 32-bit real-time seconds counter.

Both the shadow register on the digital side and the analog side register update every second. When the chip enters the power-down state, the shadow register is powered down and loses its state value. When the chip powers up, the analog side register contents are automatically copied to the shadow register. The reset value of

0x22222222 for the digital side register is only visible until the copy controller overwrites with the value from the analog side. The analog side register resets to zero upon power-on reset (POR), i.e., when the battery is first inserted or whenever a battery-less part is plugged into USB power or into a wall transformer.

EXAMPLE:

```
HW_RTC_SECONDS_WR(0); // write an initial value to the digital side. This value will
// be automatically copied to the analog side
rt_clock = HW_RTC_SECONDS_RD(); // read the 32 seconds counter value
```

19.7.5. Real-Time Clock Alarm Register Description

The 32-bit alarm value is matched against the 32-bit seconds counter to detect an alarm condition.

HW_RTC_ALARM 0x8005C040
HW_RTC_ALARM_SET 0x8005C044
HW_RTC_ALARM_CLR 0x8005C048
HW_RTC_ALARM_TOG 0x8005C04C

Table 685. HW_RTC_ALARM

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
VALUE																															

Table 686. HW_RTC_ALARM Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**VALUE**	RW	0x00000000	Seconds match-value used to trigger assertion of the RTC alarm.

DESCRIPTION:

The 32-bit alarm value can be used to awaken the chip from a power-down state or simply to cause an interrupt at a specific time.

When the chip enters the power-down state, the shadow register is powered down and loses its state value. When the chip powers up, the analog side register contents are automatically copied to the shadow register. The reset value of 0x33333333 for the digital side register is only visible until the copy controller overwrites with the value from the analog side. The analog side register resets to zero upon power-on reset (POR), i.e., when the battery is first inserted or whenever a battery-less part is plugged into a power USB or into a wall transformer.

EXAMPLE:

```
HW_RTC_ALARM_WR(60); // generate rtc alarm after 60 seconds
```

19.7.6. Watchdog Timer Register Description

The 32-bit watchdog timer can be used to reset the chip if enabled and not adequately serviced.

HW_RTC_WATCHDOG 0x8005C050
HW_RTC_WATCHDOG_SET 0x8005C054
HW_RTC_WATCHDOG_CLR 0x8005C058
HW_RTC_WATCHDOG_TOG 0x8005C05C

Table 687. HW_RTC_WATCHDOG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
COUNT																															

Table 688. HW_RTC_WATCHDOG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**COUNT**	RW	0xffffffff	If the watchdog timer decrements to zero and the watchdog timer reset is enabled, then the chip will be reset. The watchdog timer decrements once per millisecond, when enabled.

DESCRIPTION:

The 32-bit watchdog timer will reset the chip upon decrementing to zero, if this function is enabled and present on the chip. The 1-kHz source is derived from the 24-MHz crystal oscillator and does not vary when the APBX clock is changed.

EXAMPLE:

```
HW_RTC_WATCHDOG_WR(10000); // reload the watchdog and keep it from resetting the chip
```

19.7.7. Persistent State Register 0 Description

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states. Bits in this register are used by the ROM and the SDK. Refer to the SDK documentation for more specific information about the bits in this register and how they are allocated.

HW_RTC_PERSISTENT0 0x8005C060
HW_RTC_PERSISTENT0_SET 0x8005C064
HW_RTC_PERSISTENT0_CLR 0x8005C068
HW_RTC_PERSISTENT0_TOG 0x8005C06C

Table 689. HW_RTC_PERSISTENT0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
GENERAL																DCDC_CTRL										XTAL32_PDOWN	XTAL24_PDOWN	ALARM_WAKE_EN	ALARM_EN	ALARM_WAKE	CLOCKSOURCE

Table 690. HW_RTC_PERSISTENT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**GENERAL**	RW	0x0000	Firmware use, defined as follows: SDRAM_BOOT = 0x8000 Boot from SDRAM. ENUMERATE_500MA_TWICE = 0x4000 Enumerate at 500mA twice before dropping back to 100mA. USB_BOOT_PLAYER_MODE = 0x2000 Boot to player when connected to USB. SKIP_CHECKDISK = 0x1000 Run Checkdisk flag. USB_LOW_POWER_MODE = 0x0800 USB Hi/Lo Current select. OTG_HNP_BIT = 0x0400 HNP has been required if set to one. OTG_ATL_ROLE_BIT = 0x0200 USB role. SDRAM_CS_HI = 0x0100 MSB of two bit field recording which chip select (0-3) the SDRAM is connected to. SDRAM_CS_LO = 0x0080 LSB of two bit field recording which chip select (0-3) the SDRAM is connected to. SDRAM_NDX_3 = 0x0040 SDRAM configuration table index bit 3 SDRAM_NDX_2 = 0x0020 SDRAM configuration table index bit 2 SDRAM_NDX_1 = 0x0010 SDRAM configuration table index bit 1 SDRAM_NDX_0 = 0x0008 SDRAM configuration table index bit 0 ETM_ENABLE = 0x0004 ETM enable bit (0 = disabled, 1 = enabled)
15:6	**DCDC_CTRL**	RW	0x1	These bits are proprietary to SigmaTel. Customers should contact SigmaTel before changing them from their default value. SD_PRESENT = 0x200 Set to one to disable startup using internal oscillator. This bit should be set when using 24MHz as the source for the rtc. Setting this bit from 0 to 1 will also powerdown the 3600 after 500ns , so this should only be set immediately before powering down the chip via the PWD bit in HW_POWER_RESET. LOWBAT_3P0 = 0x100 Set to one to change Lithium-Ion low-battery threshold to 3.0 V. Set to zero for 2.7-V threshold. SELFBIAS_PWRUP = 0x080 Set to one to enable the self bias circuit to remain powered up when the device is powered down. This bit must also be set to allow 24-MHz crystal to be used as an RTC. AUTO_RESTART = 0x040 Set to one to enable the chip to automatically power up approximately 180 ms after powering down. DETECT_LOWBAT = 0x020 Set to one to enable 24-MHz crystal, in an RTC application, to turn off when the battery falls below threshold. The threshold is determined by LOWBAT_3P0. DROP_BIAS1 = 0x010 Set to one to decrease 24-MHz crystal bias current. DROP_BIAS2 = 0x008 Set to one to decrease 24-MHz crystal bias current an additional amount to take it a 50 percent reduction. SPARE = 0x004 Not Connected to any specific hardware function. DISABLE_XTALSTOP = 0x002 Set to one to disable the circuit that resets the chip if 24-MHz frequency falls below 2 MHz. The circuit defaults to enabled and will power down the device if the 24-MHz stop oscillating for any reason. SPARE2 = 0x001 Not Connected to any specific hardware function.
5	**XTAL32_PDOWN**	RW	0x1	Set to one to power down the 32.768-kHz crystal oscillator and its power domain, including the real time clock and persistent bits (default). Set to zero to enable the crystal oscillator and its power domain to remain on while the rest of the chip is in the power-down state.
4	**XTAL24_PDOWN**	RW	0x1	Set to one to power down the 24.0-MHz crystal oscillator, including the real-time clock and persistent bits (default). Set to zero to enable the crystal oscillator and its power domain to remain on while the rest of the chip is in the power down state. When setting this bit to zero, it is also necessary to set SELFBIAS_PWRUP, and SD_PRESENT.

Table 690. HW_RTC_PERSISTENT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**ALARM_WAKE_EN**	RW	0x0	Set this bit to one to wake up the chip upon the arrival of an alarm event. ALARM_EN must be set to one to enable the detection of an alarm event. When the alarm is not present in the device (as indicated by the fuse bits) the copy of shadow0 to persistent0 will not allow bits[2:3] to be written and persistent bits[2:3] will always read back 0, regardless of the values in the shadow register.
2	**ALARM_EN**	RW	0x0	Set this bit to one to enable the detection of an alarm event. This bit must be turned on before an alarm event can awaken a powered-down device, or before it can generate an alarm interrupt to a powered-up CPU. When the alarm is not present in the device (as indicated by the fuse bits) the copy of shadow0 to persistent0 will not allow bits[2:3] to be written and persistent bits[2:3] will always read back 0, regardless of the values in the shadow register.
1	**ALARM_WAKE**	RW	0x0	This bit is set to one to upon the arrival of an alarm event that powers up the chip. ALARM_EN must be set to one to enable the detection of an alarm event. This bit is reset by writing a zero directly to the shadow register, which causes the copy controller to move it across to the analog domain.
0	**CLOCKSOURCE**	RW	0x0	Set to one to select the 32-kHz crystal oscillator as the source for the 32-kHz clock domain used by the RTC analog domain circuits. Set to zero to select the 24-MHz crystal oscillator as the source for generating the 32-kHz clock domain used by the RTC analog domain circuits.

DESCRIPTION:

The register initalizes to a known reset pattern. The copy controller overwrites the digital reset values very soon after power on, but not in zero time.

EXAMPLE:

```
  HW_RTC_PERSISTENT0_SET(BM_RTC_PERSISTENT0_ALARM_WAKE_EN); // wake up the chip if the alarm
event occurs
  HW_RTC_PERSISTENT0_SET(BM_RTC_PERSISTENT0_CLOCKSOURCE); // select the 32KHz oscillator as
the source for the RTC analog clock
```

19.7.8. Persistent State Register 1 Description

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states. Bits in this register are used by the ROM and the SDK, and some are reserved for customers. Refer to the SDK documentation for more specific information about the bits in this register and how they are allocated.

HW_RTC_PERSISTENT1 0x8005C070
HW_RTC_PERSISTENT1_SET 0x8005C074
HW_RTC_PERSISTENT1_CLR 0x8005C078
HW_RTC_PERSISTENT1_TOG 0x8005C07C

Table 691. HW_RTC_PERSISTENT1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
GENERAL																															

Table 692. HW_RTC_PERSISTENT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**GENERAL**	RW	0x00000000	General-use persistent bits.

DESCRIPTION:

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power on, but not in zero time.

EXAMPLE:

```
HW_RTC_PERSISTENT1_WR(0x12345678); // this write will ultimately push data to the analog side via the copy controller
```

19.7.9. Persistent State (On-Chip RAM Configuration) Register 2 Description

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states. Bits in this register are used by the SDK. Refer to the SDK documentation for more specific information.

HW_RTC_PERSISTENT2 0x8005C080
HW_RTC_PERSISTENT2_SET 0x8005C084
HW_RTC_PERSISTENT2_CLR 0x8005C088
HW_RTC_PERSISTENT2_TOG 0x8005C08C

Table 693. HW_RTC_PERSISTENT2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SRAM_LO																															

Table 694. HW_RTC_PERSISTENT2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**SRAM_LO**	RW	0x00000000	See the SDK documentation.

DESCRIPTION:

After the POST runs and determines the necessary startup conditions, software copies the setup information here. At each subsequent power-up, these values are copied to the SRAM configuration.

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power-on, but not in zero time.

EXAMPLE:

```
HW_RTC_PERSISTENT2_WR(0x12345678); // this write will ultimately push data to the analog side via the copy controller
```

19.7.10. Persistent State (On-Chip RAM Configuration) Register 3 Description

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states. Bits in this register are used by the SDK. Refer to the SDK documentation for more specific information.

HW_RTC_PERSISTENT3 0x8005C090
HW_RTC_PERSISTENT3_SET 0x8005C094
HW_RTC_PERSISTENT3_CLR 0x8005C098
HW_RTC_PERSISTENT3_TOG 0x8005C09C

Table 695. HW_RTC_PERSISTENT3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SRAM_HI																															

Table 696. HW_RTC_PERSISTENT3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**SRAM_HI**	RW	0x00000000	See the SDK documentation.

DESCRIPTION:

After the POST runs and determines the necessary startup conditions, software copies the setup information here. At each subsequent power up, these values are copied to the SRAM configuration.

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power on, but not in zero time.

EXAMPLE:

```
  HW_RTC_PERSISTENT3_WR(0x12345678); // this write will ultimately push data to the analog
side via the copy controller
```

19.7.11. Real-Time Clock Debug Register Description

This 32-bit register provides debug read access to various internal states for diagnostic purposes.

HW_RTC_DEBUG 0x8005C0A0
HW_RTC_DEBUG_SET 0x8005C0A4
HW_RTC_DEBUG_CLR 0x8005C0A8
HW_RTC_DEBUG_TOG 0x8005C0AC

Table 697. HW_RTC_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																														WATCHDOG_RESET_MASK	WATCHDOG_RESET

Table 698. HW_RTC_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**RSVD0**	RO	0x0	Debug read-only view of various state machine bits.
1	**WATCHDOG_RESET_MASK**	RW	0x0	When set, masks the reset generation by the watchdog timer for testing purposes.
0	**WATCHDOG_RESET**	RO	0x0	Reflects the state of the watchdog reset. Used for testing purposes so that the watchdog can be tested without resetting part. When set, watchdog reset is asserted.

DESCRIPTION:

Read-only view into the internals of the digital side of the RTC for diagnostic purposes.

EXAMPLE:

```
DebugValue = HW_RTC_DEBUG_RD(); // read debug register value
```

19.7.12. RTC Unlock Register Description

When the RTC Unlock Register is written with a specific key, then the laser fuse registers become readable. They may be writable, provided the laser fuse contents have not locked out write operations.

HW_RTC_UNLOCK 0x8005C200
HW_RTC_UNLOCK_SET 0x8005C204
HW_RTC_UNLOCK_CLR 0x8005C208
HW_RTC_UNLOCK_TOG 0x8005C20C

Table 699. HW_RTC_UNLOCK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
KEY																															

Table 700. HW_RTC_UNLOCK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**KEY**	RW	0x0	Write 0xC6A83957 (BV_RTC_UNLOCK_KEY__VAL) to unlock access to the laser fuse registers. This register always reads back 0x0. Use the RTC_STAT FUSE_UNLOCK status bit to determine whether laserfuses are currently locked or not. VAL = 0xC6A83957 Key value needed to unlock laser fuse registers.

DESCRIPTION:

When access to the laser fuse registers is unlocked, they read back the actual values. When access is denied, all 12 laser fuse registers read back all zeroes. In addition to this lock field, there are two LOCK bits within the laser fuse registers (RTC_LASERFUSE9 bits 31:30). If either one of these two bits is set to one, then further writes to the laser fuse registers are denied under any condition. The RTC_UNLOCK register always reads back 0x0, use the RTC_STAT FUSE_UNLOCK status bit to determine the current lock status.

The RTC Unlock Register has no persistence over any power-down state.

EXAMPLE:

```
HW_RTC_UNLOCK_WR(BV_RTC_UNLOCK_KEY__VAL); // unlock laser fuse access
HW_RTC_UNLOCK_WR(0); // lock them back up so they can't be read or written
```

19.7.13. HW Laser Fuse Register 0 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE0 0x8005C300
HW_RTC_LASERFUSE0_SET 0x8005C304
HW_RTC_LASERFUSE0_CLR 0x8005C308
HW_RTC_LASERFUSE0_TOG 0x8005C30C

Table 701. HW_RTC_LASERFUSE0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 702. HW_RTC_LASERFUSE0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R1F31 through R1F00.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(0); // read laser fuse register value
```

19.7.14. HW Laser Fuse Register 1 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE1 0x8005C310
HW_RTC_LASERFUSE1_SET 0x8005C314
HW_RTC_LASERFUSE1_CLR 0x8005C318
HW_RTC_LASERFUSE1_TOG 0x8005C31C

Table 703. HW_RTC_LASERFUSE1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 704. HW_RTC_LASERFUSE1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R1F63 through R1F32.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(1); // read laser fuse register value
```

19.7.15. HW Laser Fuse Register 2 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE2 0x8005C320
HW_RTC_LASERFUSE2_SET 0x8005C324
HW_RTC_LASERFUSE2_CLR 0x8005C328
HW_RTC_LASERFUSE2_TOG 0x8005C32C

Table 705. HW_RTC_LASERFUSE2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 706. HW_RTC_LASERFUSE2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R2F31 through R2F00.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(2); // read laser fuse register value
```

19.7.16. HW Laser Fuse Register 3 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE3 0x8005C330
HW_RTC_LASERFUSE3_SET 0x8005C334
HW_RTC_LASERFUSE3_CLR 0x8005C338
HW_RTC_LASERFUSE3_TOG 0x8005C33C

Table 707. HW_RTC_LASERFUSE3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 708. HW_RTC_LASERFUSE3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R2F63 through R2F32.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(3); // read laser fuse register value
```

19.7.17. HW Laser Fuse Register 4 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE4 0x8005C340
HW_RTC_LASERFUSE4_SET 0x8005C344
HW_RTC_LASERFUSE4_CLR 0x8005C348
HW_RTC_LASERFUSE4_TOG 0x8005C34C

Table 709. HW_RTC_LASERFUSE4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 710. HW_RTC_LASERFUSE4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R3F31 through R3F00.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(4); // read laser fuse register value
```

19.7.18. HW Laser Fuse Register 5 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE5 0x8005C350
HW_RTC_LASERFUSE5_SET 0x8005C354
HW_RTC_LASERFUSE5_CLR 0x8005C358
HW_RTC_LASERFUSE5_TOG 0x8005C35C

Table 711. HW_RTC_LASERFUSE5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 712. HW_RTC_LASERFUSE5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R3F63 through R3F32.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(5); // read laser fuse register value
```

19.7.19. HW Laser Fuse Register 6 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE6 0x8005C360
HW_RTC_LASERFUSE6_SET 0x8005C364
HW_RTC_LASERFUSE6_CLR 0x8005C368
HW_RTC_LASERFUSE6_TOG 0x8005C36C

Table 713. HW_RTC_LASERFUSE6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 714. HW_RTC_LASERFUSE6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R4F31 through R4F00.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(6); // read laser fuse register value
```

19.7.20. HW Laser Fuse Register 7 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE7 0x8005C370
HW_RTC_LASERFUSE7_SET 0x8005C374
HW_RTC_LASERFUSE7_CLR 0x8005C378
HW_RTC_LASERFUSE7_TOG 0x8005C37C

Table 715. HW_RTC_LASERFUSE7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 716. HW_RTC_LASERFUSE7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R4F63 through R4F32. ALT_FORCE_POST = 0x80000000 OR with FORCE_POST, below. ALT_FORCE_REPAIR = 0x40000000 OR with FORCE_REPAIR, below. ALT_DISABLE_RECOVERY_MODE = 0x20000000 OR with DISABLE_RECOVERY, below. ALT_ICACHE_CTRL = 0x10000000 OR with ICACHE_CTRL, below. ALT_ILS_MARGIN_BITS = 0x00070000 OR with ILS_MARGIN_BITS, below. FORCE_POST = 0x00008000 OR with ALT_FORCE_POST, above. FORCE_REPAIR = 0x00004000 OR with ALT_FORCE_REPAIR, above. DISABLE_RECOVERY_MODE = 0x00002000 OR with ALT_DISABLE_RECOVERY, above. ICACHE_CTRL = 0x00001000 OR with ALT_ICACHE_CTRL, above. ILS_MARGIN_BITS = 0x00000007 OR with ALT_ILS_MARGIN_BITS, above.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(7); // read laser fuse register value
```

19.7.21. HW Laser Fuse Register 8 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE8 0x8005C380
HW_RTC_LASERFUSE8_SET 0x8005C384
HW_RTC_LASERFUSE8_CLR 0x8005C388
HW_RTC_LASERFUSE8_TOG 0x8005C38C

Table 717. HW_RTC_LASERFUSE8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 718. HW_RTC_LASERFUSE8 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R5F31 through R5F00.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(8); // read laser fuse register value
```

19.7.22. HW Laser Fuse Register 9 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE9 0x8005C390
HW_RTC_LASERFUSE9_SET 0x8005C394
HW_RTC_LASERFUSE9_CLR 0x8005C398
HW_RTC_LASERFUSE9_TOG 0x8005C39C

Table 719. HW_RTC_LASERFUSE9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 720. HW_RTC_LASERFUSE9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R5F63 through R5F32. ALT_SKIP_REPAIR = 0x00040000 OR with SKIP_REPAIR, below. SKIP_REPAIR = 0x00000400 OR with ALT_SKIP_REPAIR, above.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes. Bits [31:24] contain the alternate device identifier extension that appears in the chip revision register.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(9); // read laser fuse register value
```

19.7.23. HW Laser Fuse Register 10 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE10 0x8005C3A0
HW_RTC_LASERFUSE10_SET 0x8005C3A4
HW_RTC_LASERFUSE10_CLR 0x8005C3A8
HW_RTC_LASERFUSE10_TOG 0x8005C3AC

Table 721. HW_RTC_LASERFUSE10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 722. HW_RTC_LASERFUSE10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R6F31 through R6F00.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked, then they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(10); // read laser fuse register value
```

19.7.24. HW Laser Fuse Register 11 Description

This 32-bit laser fuse register provides software access to a portion of the 384-bit laser fuse.

HW_RTC_LASERFUSE11 0x8005C3B0
HW_RTC_LASERFUSE11_SET 0x8005C3B4
HW_RTC_LASERFUSE11_CLR 0x8005C3B8

HW_RTC_LASERFUSE11_TOG 0x8005C3BC

Table 723. HW_RTC_LASERFUSE11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BITS																															

Table 724. HW_RTC_LASERFUSE11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**BITS**	RW	0x0	Laser fuse bits R6F63 through R6F32.

DESCRIPTION:

These bits are loaded from the laser fuse blocks at first power on. If the laser fuse bits are not locked then, they can be written by the CPU. Setting either HW_LASERFUSE8_BITS[31] or HW_LASERFUSE8_BITS[30] to one locks all subsequent writes. Bits [31:24] contain the device identifier extension which appears in the chip revision register.

EXAMPLE:

```
FuseValue = HW_RTC_LASERFUSEn_RD(11); // read laser fuse register value
```

RTC XML Revision: 1.82

20. PULSE-WIDTH MODULATOR (PWM) CONTROLLER

This chapter describes the pulse-width modulator (PWM) controller included on the STMP36xx and how to use it. Programmable registers are described in Section 20.5.

20.1. Overview

The STMP36xx contains five PWM output controllers that can be used in place of GPIO pins. Applications include LED brightness control and high voltage generators for electroluminescent lamp (E.L.) display back lights. Independent output control of each phase allows zero, one, or high-Z to be independently selected for the active and inactive phases. Individual outputs can be run in lock step with guaranteed non-overlapping portions for differential drive applications.

Figure 91 shows the block diagram of the PWM controller. The controller does not use DMA. Initial values of Period, Active, and Inactive widths are set for each desired channel. The outputs are selected by phase and then the desired PWM channels are simultaneously enabled. This effectively launches the PWM outputs to autonomously drive their loads without further intervention.

In backlit high-voltage applications, a feed-forward control can be periodically used to change the count parameters based on LRADC evaluation of the battery state. Feedback control can be provided by assigning one LRADC channel to monitor the integrating capacitor voltage. Care must be taken to protect the LRADC from catastrophic over-voltage in this case. For most Electroluminescent (EL) backlight applications, open loop control with precision PWM timers based on a stable crystal oscillator is sufficient.

20.2. Operation

Each PWM channel has two control registers that are used to specify the channel output: HW_PWM_ACTIVEx and HW_PWM_PERIODx.

When programming a channel, it is important to remember that there is an order dependence for register writes.

- The HW_PWM_ACTIVEx register must be written first, followed by HW_PWM_PERIODx.
- If the order is reversed, the parameters written to the HW_PWM_ACTIVEx register will not take effect in the hardware.

The hardware waits for a HW_PWM_PERIODx register write to update the hardware with the values in both registers. This register write order dependence allows smooth on-the-fly reprogramming of the channel. Also, when the user reprograms the channel in this manner, the new register values will not take effect until the beginning of a new output period. This eliminates the potential for output glitches that could occur if the registers were updated while the channel was enabled and in the middle of a cycle.

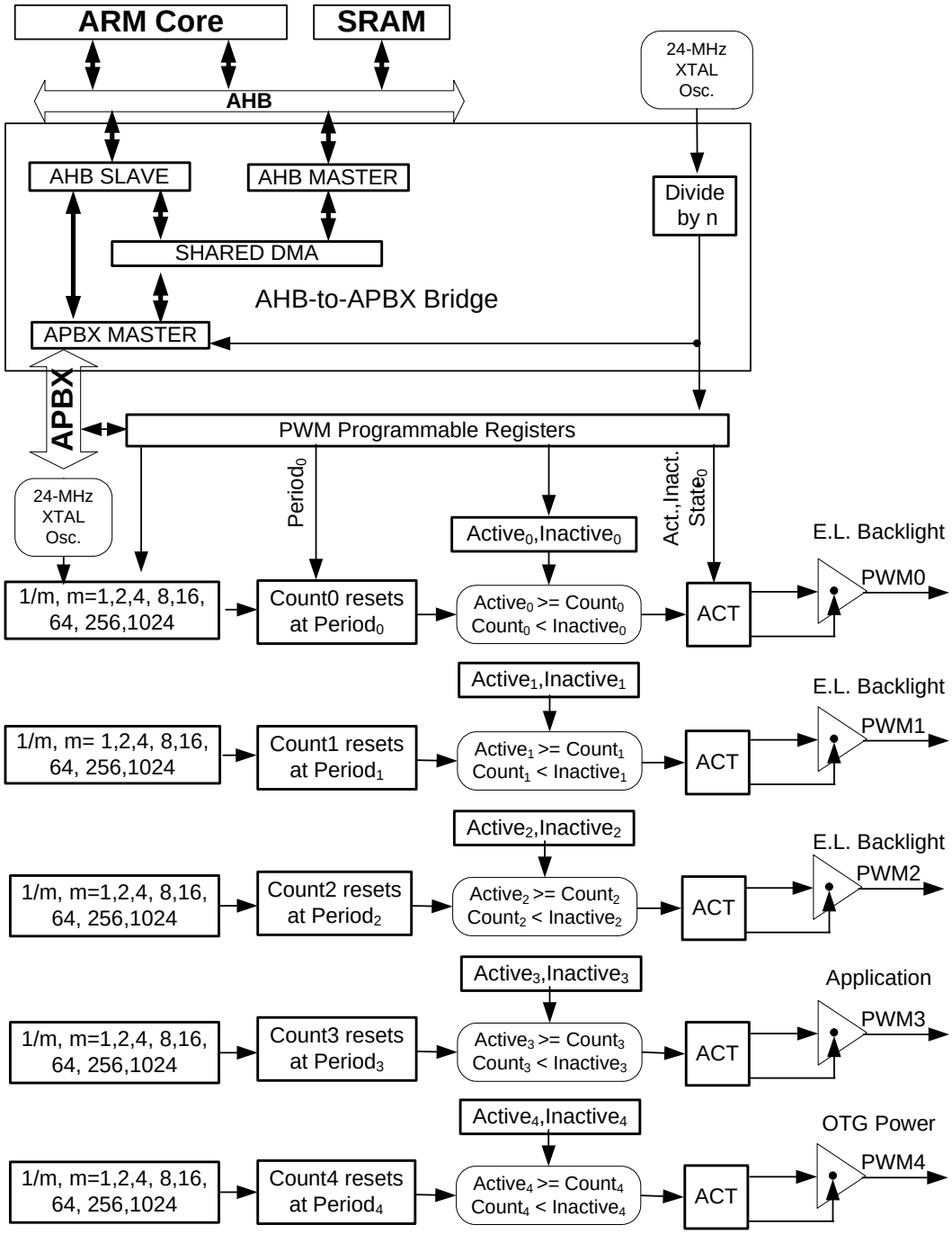


Figure 91. Pulse-Width Modulation Controller (PWM) Block Diagram

Each channel has a dedicated internal 16-bit counter that increments once for each divided clock period presented from the clock divider.

- The internal counter resets when it reaches the value stored in the channel control registers, e.g., HW_PWM_PERIOD0_PERIOD.
- The Active flip-flop is set to one when the internal counter reaches the value stored in HW_PWM_ACTIVE0_ACTIVE.
- It remains high until the internal counter exceeds the value stored in HW_PWM_ACTIVE0_INACTIVE.

These two values define the starting and ending points for the logically *"active"* portion of the waveform. As shown in Figure 92, the actual state on the output for each phase, e.g., active or inactive, is completely controlled by the active and inactive state values in the channel control registers.

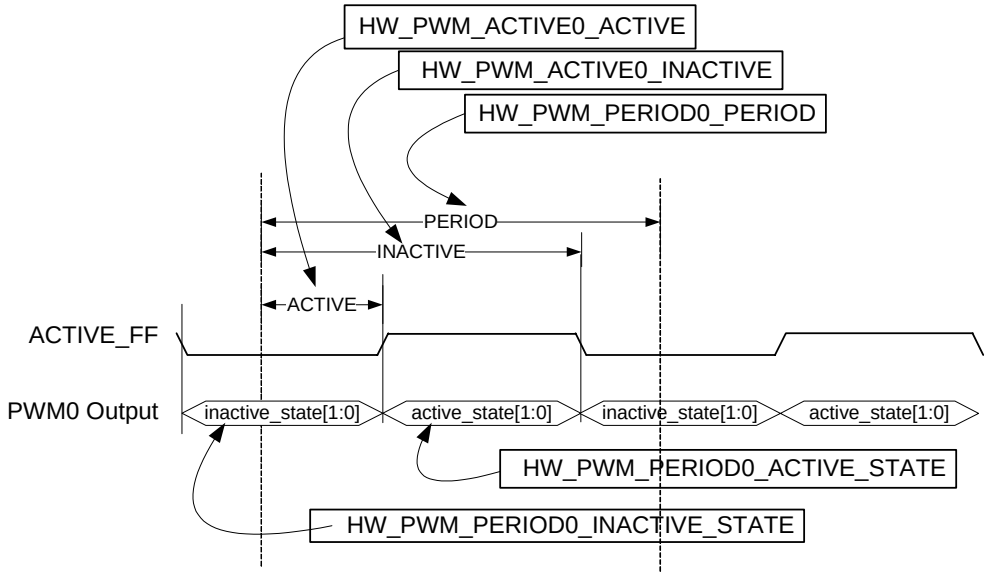
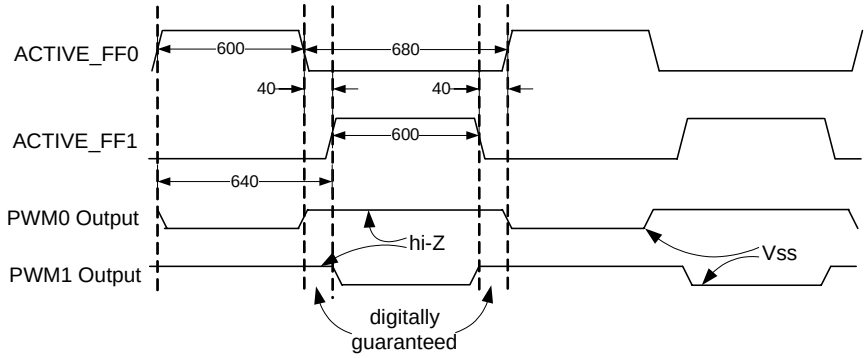


Figure 92. PWM Output Example

The actual values obtainable on the output are shown in Figure 93. Notice that one possible state is to turn off the output driver to provide a high-Z output. This is useful for external circuits that drive E.L. backlights and for direct drive of LEDs.

By setting up two channels in lock step and by setting their low and high states to opposite values, one can generate a differential signal pair that alternates between pulling to Vss and floating to high-Z. By creating an appropriate offset in the settings of the two channels with the same period and the same enables, one can generate differential drive pulses with digitally guaranteed non-overlapping intervals suitable for controlling high-voltage switches.

In Figure 93, a differential pair is established using channel zero and channel one. The period is set for 1280 divided clocks for both channels. All active phases are set for 600 divided clocks. There is a 40 divided clock guaranteed off-time between each active phase. Since this is based on a crystal oscillator, it is a very stable non-overlapping period. The total period is also a very stable crystal-oscillator-based time interval. In this example, the active phases are pulled to Vss (ground), while the inactive phases are allowed to float to a high-Z state.



HW_PWM_PERIOD0_PERIOD = 1280	HW_PWM_PERIOD1= 1280
HW_PWM_ACTIVE0_ACTIVE = 0	HW_PWM_ACTIVE1 = 640
HW_PWM_ACTIVE0_INACTIVE = 600	HW_PWM_INACTIVE1 = 1240
HW_PWM_PERIOD0_ACTIVE_STATE = 10	HW_PWM_ACTIVE1_STATE = 10
HW_PWM0BR_INACTIVE_STATE = 00	HW_PWM_INACTIVE1_STATE = 00

Figure 93. PWM Differential Output Pair Example

Figure 94 shows the generation of the PWM channel 3 output. This channel controls the output pin when PWM control is selected in PINCTRL block and HW_PWM_CTRL_PWM3_ENABLE is set to one. The output pin can be set to a zero, a one, or left to float in the high-impedance state. These choices can be made independently for either the active or inactive phase of the output.

20.3. Multi-Chip Attachment Mode

The multi-chip attachment mode (MATT) allows a 24-MHz crystal clock that is an input to the STMP36xx to be routed to the PWM output pins. In this case, the normal PWM programming parameters (e.g., PERIOD, ACTIVE, etc.) are ignored. This mode allows for supplying and controlling the crystal clock for external application interfaces, as shown in Figure 94.

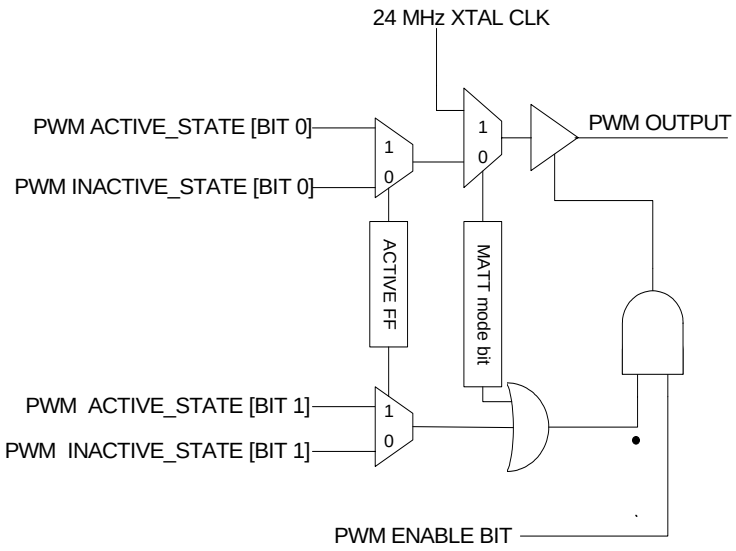


Figure 94. PWM Output Driver

20.4. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

20.5. Programmable Registers

The following registers are available for CPU programmer access and control of the PWM controller.

20.5.1. PWM Control and Status Register 0 Description

The PWM Control and Status Register 0 specifies the reset state, availability, and the enables for the five PWM units.

HW_PWM_CTRL 0x80064000
HW_PWM_CTRL_SET 0x80064004
HW_PWM_CTRL_CLR 0x80064008
HW_PWM_CTRL_TOG 0x8006400C

Table 725. HW_PWM_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	PWM4_PRESENT	PWM3_PRESENT	PWM2_PRESENT	PWM1_PRESENT	PWM0_PRESENT	RSRVD1																				PWM4_ENABLE	PWM3_ENABLE	PWM2_ENABLE	PWM1_ENABLE	PWM0_ENABLE

Table 726. HW_PWM_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it forces a block-wide reset.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29	**PWM4_PRESENT**	RO	0x1	0= PWM4 is not present in this product.
28	**PWM3_PRESENT**	RO	0x1	0= PWM3 is not present in this product.
27	**PWM2_PRESENT**	RO	0x1	0= PWM2 is not present in this product.
26	**PWM1_PRESENT**	RO	0x1	0= PWM1 is not present in this product.
25	**PWM0_PRESENT**	RO	0x1	0= PWM0 is not present in this product.
24:5	**RSRVD1**	RO	0x0	Always write zeroes to this bit field.
4	**PWM4_ENABLE**	RW	0x0	Enables PWM channel 4 to begin cycling when set to one. To enable PWM4 onto the output pin, the PINCTL registers must programmed accordingly.
3	**PWM3_ENABLE**	RW	0x0	Enables PWM channel 3 to begin cycling when set to one. To enable PWM3 onto the output pin, the PINCTL registers must programmed accordingly.
2	**PWM2_ENABLE**	RW	0x0	Enables PWM channel 2 to begin cycling when set to one. To enable PWM2 onto the output pin, the PINCTL registers must programmed accordingly.
1	**PWM1_ENABLE**	RW	0x0	Enables PWM channel 1 to begin cycling when set to one. To enable PWM1 onto the output pin, the PINCTL registers must programmed accordingly.
0	**PWM0_ENABLE**	RW	0x0	Enables PWM channel 0 to begin cycling when set to one. To enable PWM0 onto the output pin, the PINCTL registers must programmed accordingly.

DESCRIPTION:

The PWM Control and Status Register 0 specifies the reset state, availability, and the enables for the five PWM elements.

EXAMPLE:

```
Empty Example.
```

20.5.2. PWM Channel 0 Active Register Description

The PWM Channel 0 Active Register specifies the active time and inactive time for channel 0.

HW_PWM_ACTIVE0 0x80064010
HW_PWM_ACTIVE0_SET 0x80064014
HW_PWM_ACTIVE0_CLR 0x80064018
HW_PWM_ACTIVE0_TOG 0x8006401C

Table 727. HW_PWM_ACTIVE0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INACTIVE																ACTIVE															

Table 728. HW_PWM_ACTIVE0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**INACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15:0	**ACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

DESCRIPTION:

The PWM Channel 0 Active Register specifies the active time and inactive time for channel 0.

EXAMPLE:

```
Empty Example.
```

20.5.3. PWM Channel 0 Period Register Description

The PWM Channel 0 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

HW_PWM_PERIOD0 0x80064020
HW_PWM_PERIOD0_SET 0x80064024
HW_PWM_PERIOD0_CLR 0x80064028
HW_PWM_PERIOD0_TOG 0x8006402C

Table 729. HW_PWM_PERIOD0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2								MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE		PERIOD															

Table 730. HW_PWM_PERIOD0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
23	**MATT**	RW	0x0	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables the 24-MHz crystal clock on the PWM0 output pin for inter-chip signaling.
22:20	**CDIV**	RW	0x0	Clock divider ratio to apply to the crystal clock frequency (24.0 MHz) that times the PWM output signal. DIV_1 = 0x0 Divide by 1. DIV_2 = 0x1 Divide by 2. DIV_4 = 0x2 Divide by 4. DIV_8 = 0x3 Divide by 8. DIV_16 = 0x4 Divide by 16. DIV_64 = 0x5 Divide by 64. DIV_256 = 0x6 Divide by 256. DIV_1024 = 0x7 Divide by 1024.
19:18	**INACTIVE_STATE**	RW	0x0	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Inactive state sets PWM output to high-impendance. 0 = 0x2 Inactive state sets PWM output to 0 (low). 1 = 0x3 Inactive state sets PWM output to 1 (high).
17:16	**ACTIVE_STATE**	RW	0x0	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Active state sets PWM output to high-impendance. 0 = 0x2 Active state sets PWM output to 0 (low). 1 = 0x3 Active state sets PWM output to 1 (high).
15:0	**PERIOD**	RW	0x0	Number of divided XTAL clock cycles in the entire period of the PWM waveform, minus one. For example, to obtain six clock cycles in the actual period, then set this field to five.

DESCRIPTION:

The PWM Channel 0 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

EXAMPLE:

```
Empty Example.
```

20.5.4. PWM Channel 1 Active Register Description

The PWM Channel 1 Active Register specifies the active time and inactive time for channel 1.

HW_PWM_ACTIVE1 0x80064030
HW_PWM_ACTIVE1_SET 0x80064034
HW_PWM_ACTIVE1_CLR 0x80064038
HW_PWM_ACTIVE1_TOG 0x8006403C

Table 731. HW_PWM_ACTIVE1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INACTIVE																ACTIVE															

Table 732. HW_PWM_ACTIVE1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**INACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15:0	**ACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

DESCRIPTION:

The PWM Channel 1 Active Register specifies the active time and inactive time for channel 1.

EXAMPLE:

```
Empty Example.
```

20.5.5. PWM Channel 1 Period Register Description

The PWM Channel 1 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

HW_PWM_PERIOD1 0x80064040
HW_PWM_PERIOD1_SET 0x80064044
HW_PWM_PERIOD1_CLR 0x80064048
HW_PWM_PERIOD1_TOG 0x8006404C

Table 733. HW_PWM_PERIOD1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2								MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE		PERIOD															

Table 734. HW_PWM_PERIOD1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
23	**MATT**	RW	0x0	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables the 24-MHz crystal clock on the PWM1 output pin for inter-chip signaling.
22:20	**CDIV**	RW	0x0	Clock divider ratio to apply to the crystal clock frequency (24.0 MHz) that times the PWM output signal. DIV_1 = 0x0 Divide by 1. DIV_2 = 0x1 Divide by 2. DIV_4 = 0x2 Divide by 4. DIV_8 = 0x3 Divide by 8. DIV_16 = 0x4 Divide by 16. DIV_64 = 0x5 Divide by 64. DIV_256 = 0x6 Divide by 256. DIV_1024 = 0x7 Divide by 1024.
19:18	**INACTIVE_STATE**	RW	0x0	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Inactive state sets PWM output to high-impendance. 0 = 0x2 Inactive state sets PWM output to 0 (low). 1 = 0x3 Inactive state sets PWM output to 1 (high).
17:16	**ACTIVE_STATE**	RW	0x0	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Active state sets PWM output to high-impendance. 0 = 0x2 Active state sets PWM output to 0 (low). 1 = 0x3 Active state sets PWM output to 1 (high).
15:0	**PERIOD**	RW	0x0	Number of divided XTAL clock cycles in the entire period of the PWM waveform, minus one. For example, to obtain six clock cycles in the actual period, then set this field to five.

DESCRIPTION:

The PWM Channel 1 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

EXAMPLE:

```
Empty Example.
```

20.5.6. PWM Channel 2 Active Register Description

The PWM Channel 2 Active Register specifies the active time and inactive time for channel 2.

HW_PWM_ACTIVE2 0x80064050
HW_PWM_ACTIVE2_SET 0x80064054
HW_PWM_ACTIVE2_CLR 0x80064058
HW_PWM_ACTIVE2_TOG 0x8006405C

Table 735. HW_PWM_ACTIVE2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INACTIVE																ACTIVE															

Table 736. HW_PWM_ACTIVE2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**INACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15:0	**ACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

DESCRIPTION:

The PWM Channel 2 Active Register specifies the active time and inactive time for channel 2.

EXAMPLE:

```
Empty Example.
```

20.5.7. PWM Channel 2 Period Register Description

The PWM Channel 2 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

HW_PWM_PERIOD2 0x80064060
HW_PWM_PERIOD2_SET 0x80064064
HW_PWM_PERIOD2_CLR 0x80064068
HW_PWM_PERIOD2_TOG 0x8006406C

Table 737. HW_PWM_PERIOD2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2								MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE		PERIOD															

Table 738. HW_PWM_PERIOD2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
23	**MATT**	RW	0x0	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables the 24-MHz crystal clock on the PWM2 output pin for inter-chip signaling.
22:20	**CDIV**	RW	0x0	Clock divider ratio to apply to the crystal clock frequency (24.0 MHz) that times the PWM output signal. DIV_1 = 0x0 Divide by 1. DIV_2 = 0x1 Divide by 2. DIV_4 = 0x2 Divide by 4. DIV_8 = 0x3 Divide by 8. DIV_16 = 0x4 Divide by 16. DIV_64 = 0x5 Divide by 64. DIV_256 = 0x6 Divide by 256. DIV_1024 = 0x7 Divide by 1024.
19:18	**INACTIVE_STATE**	RW	0x0	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Inactive state sets PWM output to high-impendance. 0 = 0x2 Inactive state sets PWM output to 0 (low). 1 = 0x3 Inactive state sets PWM output to 1 (high).
17:16	**ACTIVE_STATE**	RW	0x0	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Active state sets PWM output to high-impendance. 0 = 0x2 Active state sets PWM output to 0 (low). 1 = 0x3 Active state sets PWM output to 1 (high).
15:0	**PERIOD**	RW	0x0	Number of divided XTAL clock cycles in the entire period of the PWM waveform, minus one. For example, to obtain six clock cycles in the actual period, then set this field to five.

DESCRIPTION:

The PWM Channel 2 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

EXAMPLE:

```
Empty Example.
```

20.5.8. PWM Channel 3 Active Register Description

The PWM Channel 3 Active Register specifies the active time and inactive time for channel 3.

HW_PWM_ACTIVE3 0x80064070
HW_PWM_ACTIVE3_SET 0x80064074
HW_PWM_ACTIVE3_CLR 0x80064078
HW_PWM_ACTIVE3_TOG 0x8006407C

Table 739. HW_PWM_ACTIVE3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INACTIVE																ACTIVE															

Table 740. HW_PWM_ACTIVE3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**INACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15:0	**ACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

DESCRIPTION:

The PWM Channel 3 Active Register specifies the active time and inactive time for channel 3.

EXAMPLE:

```
Empty Example.
```

20.5.9. PWM Channel 3 Period Register Description

The PWM Channel 3 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

HW_PWM_PERIOD3 0x80064080
HW_PWM_PERIOD3_SET 0x80064084
HW_PWM_PERIOD3_CLR 0x80064088
HW_PWM_PERIOD3_TOG 0x8006408C

Table 741. HW_PWM_PERIOD3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2								MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE		PERIOD															

Table 742. HW_PWM_PERIOD3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
23	**MATT**	RW	0x0	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables the 24-MHz crystal clock on the PWM3 output pin for inter-chip signaling.
22:20	**CDIV**	RW	0x0	Clock divider ratio to apply to the crystal clock frequency (24.0 MHz) that times the PWM output signal. DIV_1 = 0x0 Divide by 1. DIV_2 = 0x1 Divide by 2. DIV_4 = 0x2 Divide by 4. DIV_8 = 0x3 Divide by 8. DIV_16 = 0x4 Divide by 16. DIV_64 = 0x5 Divide by 64. DIV_256 = 0x6 Divide by 256. DIV_1024 = 0x7 Divide by 1024.
19:18	**INACTIVE_STATE**	RW	0x0	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Inactive state sets PWM output to high-impendance. 0 = 0x2 Inactive state sets PWM output to 0 (low). 1 = 0x3 Inactive state sets PWM output to 1 (high).
17:16	**ACTIVE_STATE**	RW	0x0	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Active state sets PWM output to high-impendance. 0 = 0x2 Active state sets PWM output to 0 (low). 1 = 0x3 Active state sets PWM output to 1 (high).
15:0	**PERIOD**	RW	0x0	Number of divided XTAL clock cycles in the entire period of the PWM waveform, minus one. For example, to obtain six clock cycles in the actual period, then set this field to five.

DESCRIPTION:

The PWM Channel 3 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

EXAMPLE:

```
Empty Example.
```

20.5.10. PWM Channel 4 Active Register Description

The PWM Channel 4 Active Register specifies the active time and inactive time for channel 4.

HW_PWM_ACTIVE4 0x80064090
HW_PWM_ACTIVE4_SET 0x80064094
HW_PWM_ACTIVE4_CLR 0x80064098
HW_PWM_ACTIVE4_TOG 0x8006409C

Table 743. HW_PWM_ACTIVE4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INACTIVE																ACTIVE															

Table 744. HW_PWM_ACTIVE4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**INACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15:0	**ACTIVE**	RW	0x0	Number of divided XTAL clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

DESCRIPTION:

The PWM Channel 4 Active Register specifies the active time and inactive time for channel 4.

EXAMPLE:

```
Empty Example.
```

20.5.11. PWM Channel 4 Period Register Description

The PWM Channel 4 Period Register specifies the multi-chip attachment mode, clock divider value, active high/low values, and period.

HW_PWM_PERIOD4 0x800640A0
HW_PWM_PERIOD4_SET 0x800640A4
HW_PWM_PERIOD4_CLR 0x800640A8
HW_PWM_PERIOD4_TOG 0x800640AC

Table 745. HW_PWM_PERIOD4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2								MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE		PERIOD															

Table 746. HW_PWM_PERIOD4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD2**	RO	0x0	Always write zeroes to this bit field.
23	**MATT**	RW	0x0	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables the 24-MHz crystal clock on the PWM4 output pin for inter-chip signaling.
22:20	**CDIV**	RW	0x0	Clock divider ratio to apply to the crystal clock frequency (24.0 MHz) that times the PWM output signal. DIV_1 = 0x0 Divide by 1. DIV_2 = 0x1 Divide by 2. DIV_4 = 0x2 Divide by 4. DIV_8 = 0x3 Divide by 8. DIV_16 = 0x4 Divide by 16. DIV_64 = 0x5 Divide by 64. DIV_256 = 0x6 Divide by 256. DIV_1024 = 0x7 Divide by 1024.
19:18	**INACTIVE_STATE**	RW	0x0	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Inactive state sets PWM output to high-impendance. 0 = 0x2 Inactive state sets PWM output to 0 (low). 1 = 0x3 Inactive state sets PWM output to 1 (high).
17:16	**ACTIVE_STATE**	RW	0x0	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-Z. HI_Z = 0x0 Active state sets PWM output to high-impendance. 0 = 0x2 Active state sets PWM output to 0 (low). 1 = 0x3 Active state sets PWM output to 1 (high).
15:0	**PERIOD**	RW	0x0	Number of divided XTAL clock cycles in the entire period of the PWM waveform, minus one. For example, to obtain six clock cycles in the actual period, then set this field to five.

DESCRIPTION:

The PWM Channel 4 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

EXAMPLE:

```
Empty Example.
```

PWM XML Revision: 1.24

21. I²C INTERFACE

This chapter describes the I²C interface implemented on the STMP36xx. It includes sections on the external pins, interrupt sources, I²C bus protocol, and programming examples. Programmable registers are included in Section 21.7.

21.1. Overview

The I²C is a standard two-wire serial interface used to connect the chip with peripherals or host controllers. This interface provides a standard speed (up to 100 kbps), and a fast speed (up to 400 kbps) I²C connection to multiple devices with the chip acting in either I²C master or I²C slave mode. Typical applications for the I²C bus include: EEPROM, LED/LCD, FM tuner, cell phone baseband chip connection, etc.

The I²C port supports multi-master configurations.

As implemented on the STMP36xx, the I²C block includes the following functions:

- The I²C block can be configured as either a master or slave device. In master mode, it generates the clock (I2C_SCL) and initiates transactions on the data line (I2C_SDA).
- The I²C block packs/unpacks data into 8-, 16-, 24-, or 32-bit words for DMA transactions. Data on the I²C bus is always byte-oriented. Short transmission (up to three bytes plus address) can be easily triggered using only PIO operations, i.e., no DMA setup required.
- The I²C block has programmable device addresses for master transactions. It also has a programmable 7-bit address that defaults to 0x43 = 7'b1000011 for slave transactions. As seen in the 8-bit device address byte, this address corresponds to 0x86 where the least significant bit (LSB) is the R/W bit.
- Master transactions are composed of one or more DMA commands chained together. The first byte conveys the slave address and read/write bit for the first command. If the entire transaction is an I²C write command, then it can be sent by a single DMA command. If the command is an I²C read transaction, then at least two DMA commands are required to handle it.
- When the slave interface is enabled, it immediately goes into address search mode and searches for a start event. It then looks for a match on its programmable device address. As soon as the address byte is matched, it is acknowledged on the I²C bus and then the SCL clock is held low until released by software. The address phase initiates a CPU interrupt if a slave address match is detected. Software then reads the address LSB to determine whether to use a read or write DMA command to complete the slave transaction.

Figure 95 shows a block diagram of the I²C interface implemented on the STMP36xx.

21.2. I²C Interface External Pins

I2C_SDAQ: I²C Serial Data—This pin carries all address and data bits.

I2C_SCL: I²C Serial Clock—This pin carries the clock used to time the address and data.

Pullup resistors are required on both of the I²C lines as all of the I²C drivers are open drain (pulldown only). Typically, external 2kΩ resistors are used to pull the signals up to VDDIO for normal and fast speeds.

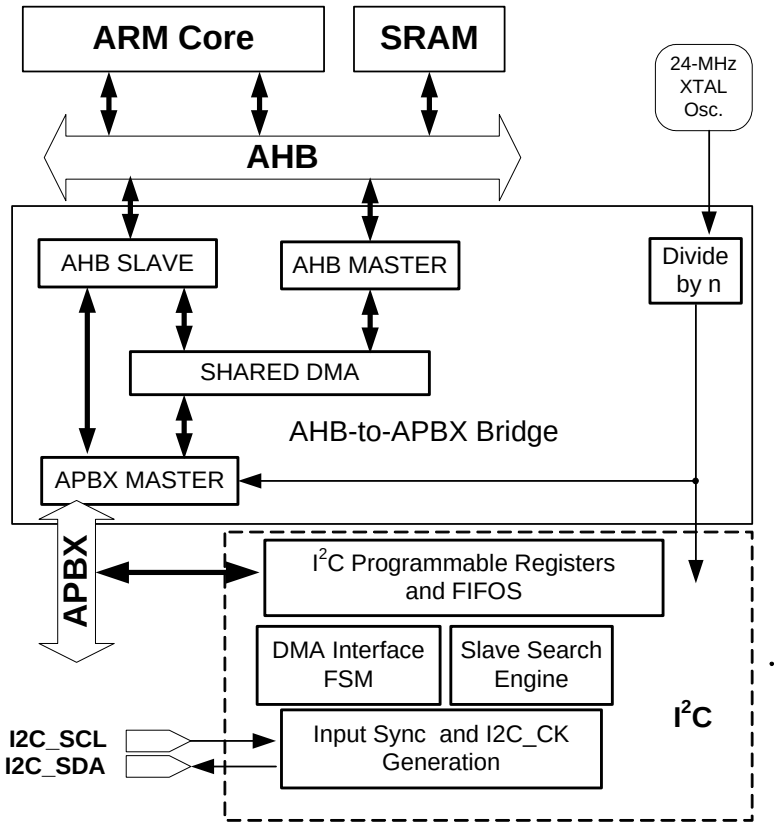


Figure 95. I²C Interface Block Diagram

21.3. I²C Interrupt Sources

The I²C port can be used in either interrupt-driven or polled modes. An interrupt can be generated by the completion of a DMA command in the APBX DMA. DMA interrupts are the reporting mechanism for I²C transactions that terminate normally. Abnormal terminations or partial completions are signaled by interrupts generated within the I²C controller.

If I²C interrupts are enabled, a level-sensitive interrupt will be signaled to the processor upon one of the events listed in Table 747.

Table 747. I²C Slave and Master Interrupt Conditions

SOURCE	BIT NAME	DESCRIPTION
Slave Address	HW_I2C_CTRL1_SLAVE_IRQ	This interrupt is generated when an address match occurs. It indicates that the CPU should read the captured RW bit from the I²C address byte to determine the type of DMA to use for the data transfer phase.
Slave Stop	HW_I2C_CTRL1_SLAVE_STOP_IRQ	This interrupt is generated when a stop condition is detected after a slave address has been matched.
Oversize Xfer	HW_I2C_CTRL1_OVERSIZE_XFER_TERM_IRQ	The DMA and I²C controller are initialized for an expected transfer size. If the data phase is not terminated within this transfer size then oversize transfer processing goes into effect and the CPU is alerted via this interrupt.
Early Termination	HW_I2C_CTRL1_EARLY_TERM_IRQ	The DMA and I²C controller are initialized for an expected transfer size. If the data phase is terminated before this transfer size then early termination processing goes into effect and the CPU is alerted via this interrupt.
Master Loss	HW_I2C_CTRL1_MASTER_LOSS_IRQ	A master begins transmission on an idle I²C bus and monitors the data line. If it ever attempts to send a one on the line and notes that a zero has been sent instead, then it notes that it has lost mastership of the I²C bus. It terminates its transfer and reports the condition to the CPU via this interrupt. This detection only happens on master transmit operations.
No Slave Ack	HW_I2C_CTRL1_NO_SLAVE_ACK_IRQ	When a start condition is transmitted in master mode, the next byte contains an address for a targeted slave. If the targeted slave does not acknowledge the address byte, then this interrupt is set, no further I²C protocol is processed, and the I²C bus returns to the idle state.
Data Engine Complete	HW_I2C_CTRL1_DATA_ENGINE_CMPLT_IRQ	This bit is set whenever the DMA interface state machine completes a transaction and resets its run bit. This is useful for PIO mode transmit transactions that are not mediated by the DMA and therefore cannot use the DMA command completion interrupt. This bit is still set for master completions when the DMA is used, but can be ignored in that case.
Bus Free	HW_I2C_CTRL_BUS_FREE_IRQ	When bus mastership is lost during the I²C arbitration phase, the bus becomes busy running services for another master. This interrupt is set whenever a stop command is detected so the master transaction can attempt a retry.

The interrupt lines are tied directly to the bits of Control Register 1. Clearing these bits through software removes the interrupt request.

21.4. I²C Bus Protocol

The I²C interface operates as shown in Figure 96 and Figure 97.

- A START condition is defined as a high-to-low transition on the data line while the I2C_SCL line is held high.
- After this has been transmitted by the master, the bus is considered busy.
- The next byte of data transmitted after the start condition contains the address of the slave in the first seven bits, and the eighth bit tells whether the Master is receiving data from the slave or transmitting data to the slave.
- When an address is sent, each device in the system compares the first seven bits after a start condition with its address.
- If they match, the device considers itself addressed by the master.

In slave mode, the default I²C write address is 86h, and its default read address is 87h. The slave address is programmable.

Data transfer with acknowledge is obligatory.

- The transmitter must release the I2C_SDA line during the acknowledge pulse.
- The receiver must then pull the data line low, so that it remains stable low during the high period of the acknowledge clock pulse.
- A receiver that has been addressed is obliged to generate an acknowledge after each byte of data has been received.
- A slave device can terminate a transfer by withholding its acknowledgement.

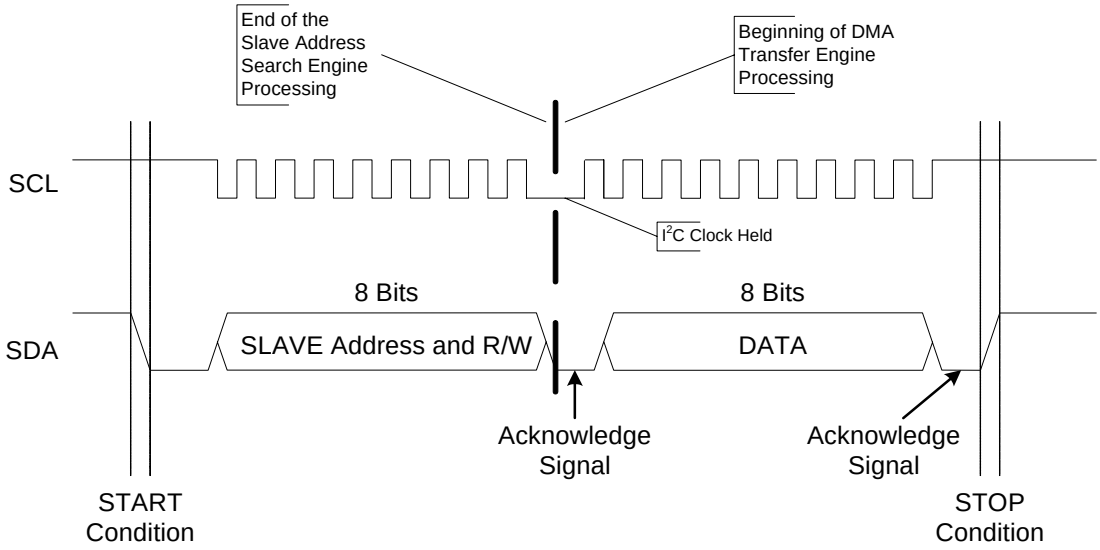


Figure 96. I²C Data and Clock Timing

The clock is generated by the master, according to parameters set in the HW_I2C_TIMING register. This register also provides programmable timing for capturing received data, as well as for changing the transmitted data bit.

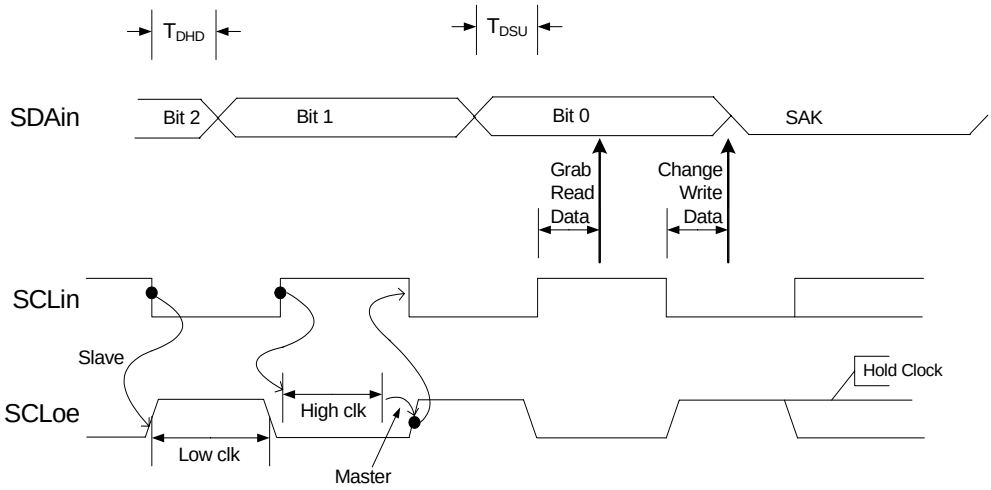


Figure 97. I²C Data and Clock Timing Generation

21.4.1. Simple Device Transactions

The simplest transfer of interest on an I²C bus is writing a single data byte from a master to a slave, for example, writing a single byte to an FM tuner. In this transaction, a start condition is transmitted, followed by the device address byte, followed by a single byte of write data. This sequence always ends with a stop condition.

Table 748. I²C Transfer When the Interface is Transmitting as a Master

ST	SAD+W	SAK	DATA	SAK	SP

Table 749 defines the symbols used in describing I²C transactions. For example, in the single byte write operation, ST is a start condition, and SP is a stop condition. The data transfer occurs between these two bus events. It starts with a slave address plus write byte (SAD+W) addressing the targeted slave. A slave-generated acknowledge bit (SAK) tells the master that a slave has recognized the address and will accept the transfer. The master sends the data byte (DATA), and the slave acknowledges it with an SAK.

Table 749. I²C Slave and Master Mode Address Definitions

BIT	DESCRIPTION
ST	Start Condition
SR	Repeated Start Condition
SAD	Slave Address
SAK	Slave Acknowledge
SUB	Sub-Address, e.g., for EEPROMs
DATA	Data
SP	Stop Condition
MAK	Master Acknowledge
NMAK	No Master Acknowledge

To receive one data byte from a slave device such as an FM tuner, the following bus transaction takes place.

Table 750. I²C Transfer "FM Tuner" Read of One Byte

ST	SAD+R	SAK	DATA	MAK	SP

In this transaction:

- The master first generates a start condition, ST.
- It then sends the seven-bit slave address for the FM tuner plus a read bit (SAD+R).
- The slave in the FM tuner responds with a slave acknowledge bit (SAK).
- The master then generates I²C clocks for a data byte to be transferred (DATA).
- The slave provides data to the I²C data bus during the DATA byte transfer.
- Next, the master generates a master acknowledge to the slave (MAK), indicating its acceptance of the data byte.
- Finally, the master generates a stop condition (SP), terminating the transaction and freeing the I²C bus for other masters to use.

The following example shows a multiple byte read from an FM tuner or other slave device:

Table 751. I²C Transfer "FM Tuner" Read of Three Bytes

ST	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

21.4.2. *Typical EEPROM Transactions*

I²C EEPROMs typically have a specific transaction sequence for reading and writing data bytes to and from the EEPROM array. Table 752 through Table 755 show the first two bytes of data as a sub-address for purposes of illustration. The sub-address is used to address the memory space inside the device. Table 749 defines each element of the transactions shown. When writing a single byte of data to the EEPROM, one must first transfer two bytes of sub-address as follows:

Table 752. I²C Transfer When Master is Writing One Byte of Data to a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	SP

The sub-address only needs to be specified once for a multibyte transfer, as shown here. Note that the sub-address must be sent for each start condition that initiates a transaction.

Table 753. I²C Transfer When Master is Writing Multiple Bytes to a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	DATA	SAK	SP

One must also provide the sub-address before reading bytes from the EEPROM. The sub-address is transmitted from the master to the slave before it can receive data bytes. The two transfers are joined into a single bus transaction though the use of a repeated start condition (SR). Normally, a stop condition precedes a start condition. However, when a start condition is preceded by another start condition, it is known as a repeated start (SR). Note that the two-byte sub address is transferred using an SAD+W address, while the data is received using a SAD+R address.

Table 754. I²C Transfer When Master is Receiving One Byte of Data from a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	NMAK	SP

Table 755. I²C Transfer When Master is Receiving Multiple Bytes of Data from a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

21.4.3. Master Mode Protocol

In master mode, the I²C interface generates the clock and initiates all transfers.

21.4.3.1. Clock Generation

The I²C clock is generated from the APBX clock, as described in the register description.

- If another device pulls the clock low before the I²C block has counted the high period, then the I²C block immediately pulls the clock low as well and starts counting its low period.
- Once the low period has been counted, the I²C block releases the clock line high, but must then check to see if another device stills holds the line low, in which case it enters a high wait state.

In this way, the I2C_SCL clock is generated, with its low period determined by the device with the longest clock low period and its high period determined by the one with the shortest clock high period.

21.4.3.2. Master Mode Operation

The finite state machine for master mode operation is shown in Figure 98 through Figure 101. Figure 98 shows the generation of the optional start condition. Figure 99 shows the receive states, Figure 100 shows the transmit states. Figure 101 shows the generation of the optional stop state.

Table 756 through Table 759 show examples of Master Mode I²C transactions. Table 749 defines each sub-address shown. The following read-after-write transactions are performed using the restart technique.

Table 756. I²C Transfer When the Interface as Master is Transmitting One Byte of Data

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	SP

Table 757. I²C Transfer When the Interface as Master is Receiving >1 Byte of Data from Slave

ST	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

Table 758. I²C Transfer when Master is Receiving 1 Byte of Data from Slave Internal Subaddress

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	NMAK	SP

Table 759. I²C Transfer When Master is Receiving >1 byte of Data from Slave Internal Subaddress

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

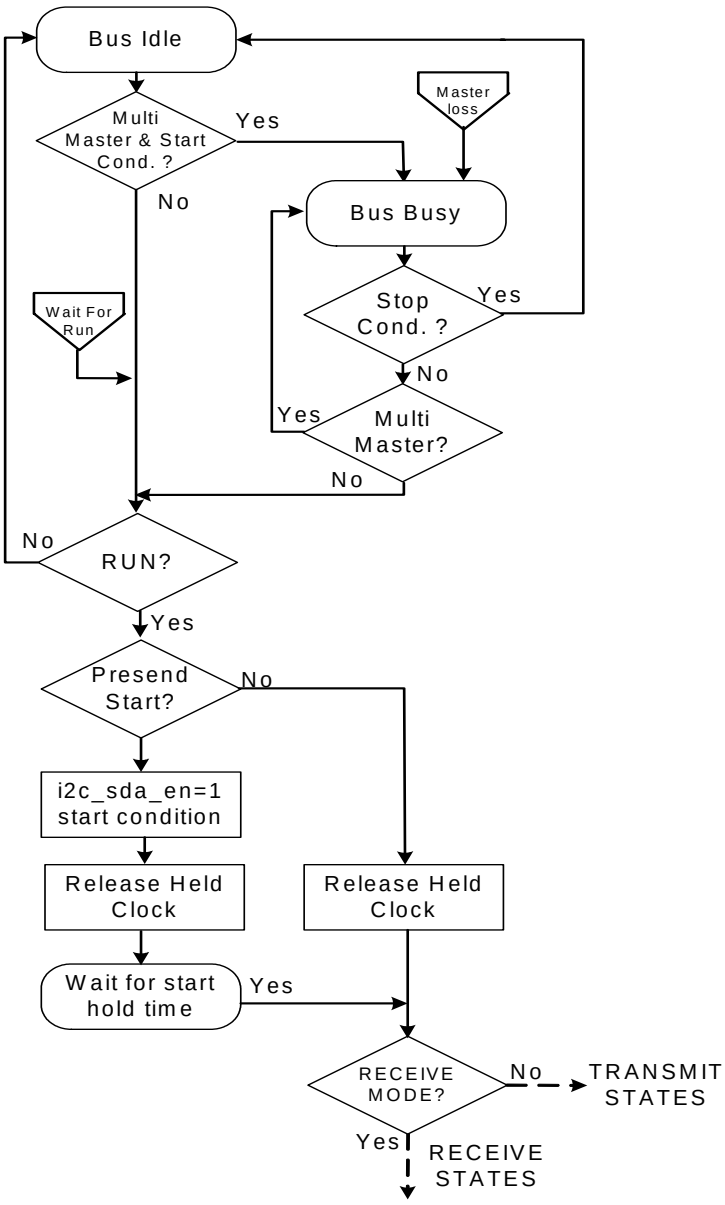


Figure 98. I²C Master Mode Flow Chart—Initial States

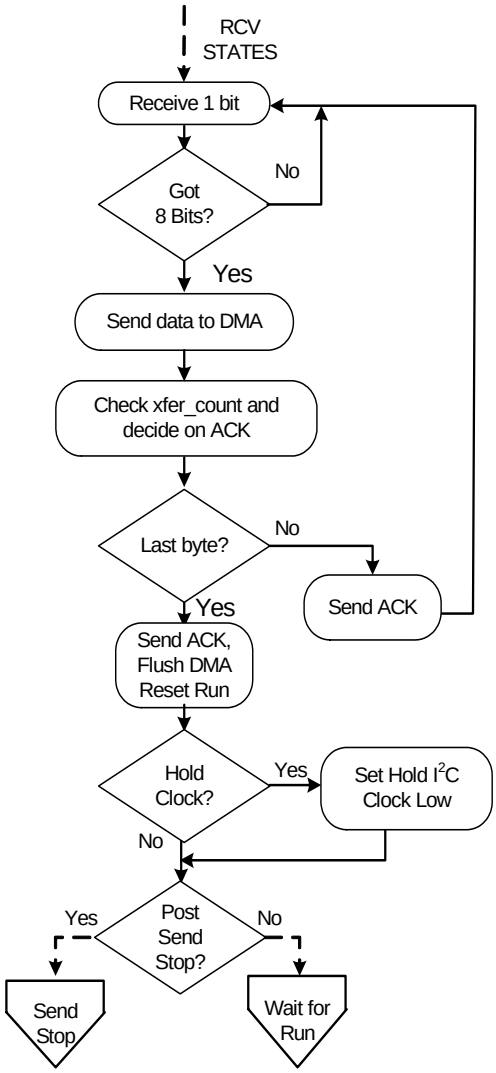


Figure 99. I²C Master Mode Flow Chart—Receive States

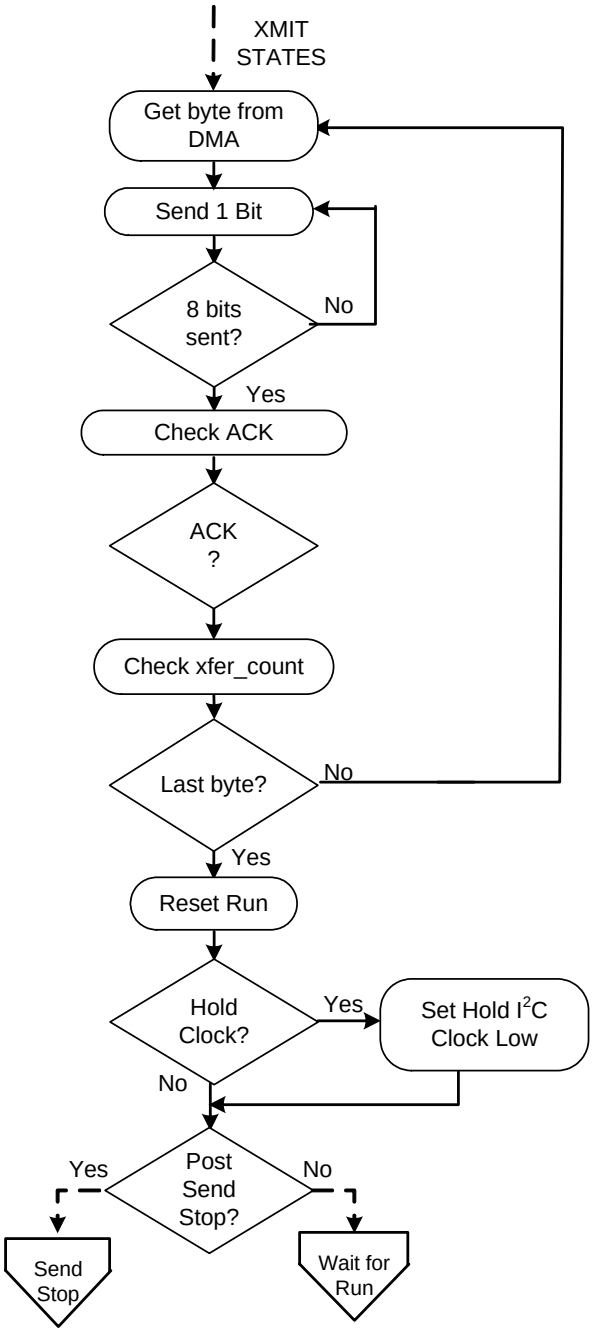


Figure 100. I²C Master Mode Flow Chart—Transmit States

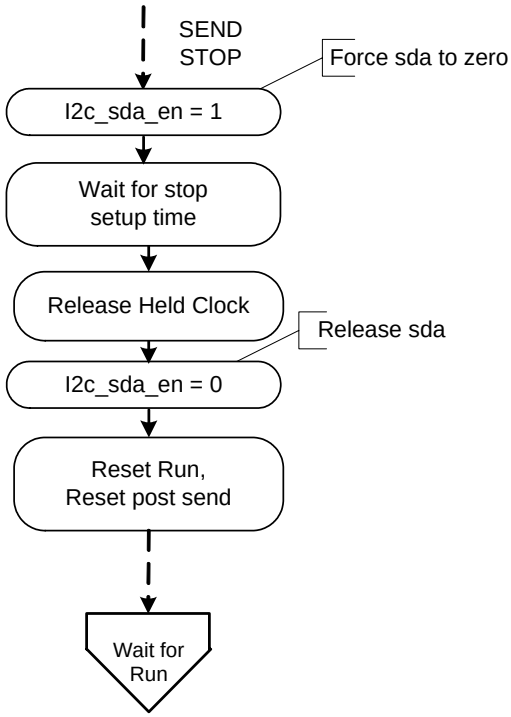


Figure 101. I²C Master Mode Flow Chart—Send Stop States

21.4.4. Slave Mode Protocol

The I²C slave protocol is handled by a combination of I²C functional block hardware, the DMA, and some supporting software to intervene in the transaction.

The flow chart for slave mode is shown in Figure 102.

- At device start-up, all the registers are reset so that the state is known from that time onward.
- Once the I²C slave search engine is enabled, the slave waits to detect a start condition on the I2C_SCL and I2C_SDA lines.
- Once this is detected, the slave reads in eight bits and checks against its programmed device address (which defaults to 0x86 == 7'b1000011) to see if a master device is trying to start a transfer with STMP36xx operating as a slave.
- If it is the programmed address, an acknowledge is sent; otherwise the slave does not acknowledge and returns to state IDLE.
- Once the slave search engine detects an address, it holds the clock line and interrupts the CPU.
- Next the software checks the RW bit.
- If it is a write operation, then the software programs the DMA channel for a DMA_WRITE (to on-chip RAM or off-chip SDRAM).
- The slave search engine leaves the programmable state set up for the DMA transfer engine to send the address acknowledge for the address byte as soon as the clock is released.

- It then accepts eight-bit bytes and pushes them into the DMA data register, acknowledging each data byte as it is received, until the transfer count reaches zero.
- The DMA engine stops with the clock held and the hardware ready to acknowledge the last byte when the clock is released. Software decides whether the last byte is acknowledged or not.
- If the master is requesting a read operation, then the STMP36xx slave must start sending data on the I2C_SDA bus immediately after acknowledging the slave address and RW bit.
- After each byte, the acknowledge from the master must be checked. When the master has received the last byte, it does not send an acknowledge, and the slave terminates while setting the Early Termination interrupt request. This notifies software that the DMA will not be interrupting for the termination and that software should deal with a shorter than expected packet of data.
- If the transfer count reaches zero and the master has not sent an MNAK or stop condition, then the slave DMA transfer controller terminates the transfer while setting the Oversize Transfer interrupt request. This notifies software to set up for an additional buffer of data to transmit to the master.

Data is transmitted in byte format. Each data transfer has to contain eight bits. The number of bytes transferred per transfer is unlimited. Data is transferred with the most significant bit (MSB) first. If a receiver cannot receive another complete byte of data until it has performed some other function, it can hold the I2C_SCL clock line low to force the transmitter into a wait state. Data transfer can only continue when the receiver is ready for another byte and releases the clock line.

If a slave receiver does not acknowledge the slave address (e.g., it is unable to receive because it is performing some real-time function), the data line must be left high by the slave. The master can then abort the transfer.

A low-to-high transition on the I2C_SDA line while the I2C_SCL line is high is defined as a Stop condition. Each data transfer must be terminated by the generation of a Stop condition. A write transfer from a master can be terminated by the master by sending a Stop condition instead of an additional data byte. The STMP36xx slave DMA transfer engine reports this to software as an Early Termination interrupt request.

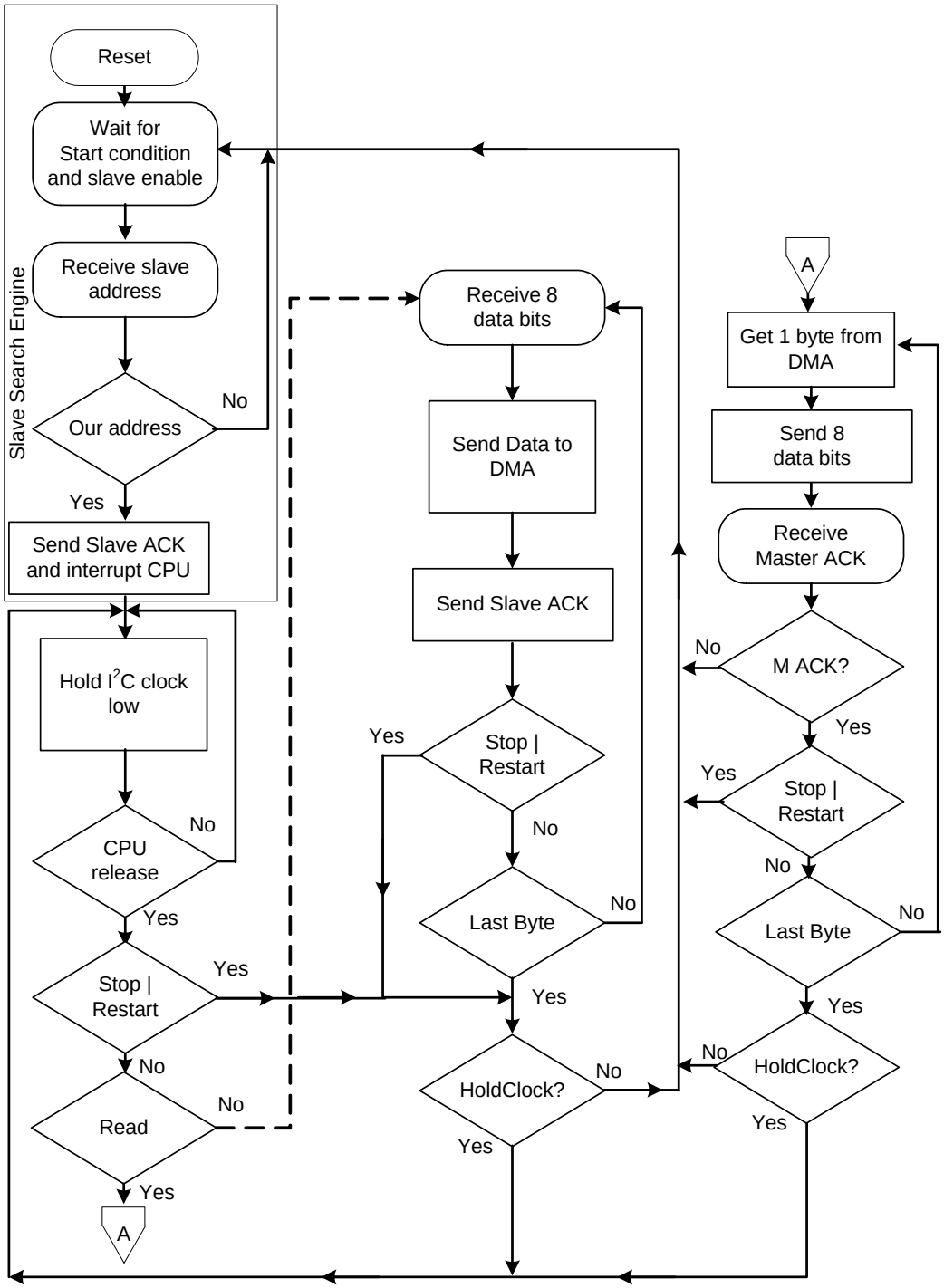


Figure 102. I²C Slave Mode Flow Chart

21.5. Programming Examples

21.5.1. Five Byte Master Write Using DMA

The example in Figure 103 shows sending five bytes from an STMP36xx operating as an I^{2}C master to another device acting as an I^{2}C slave.

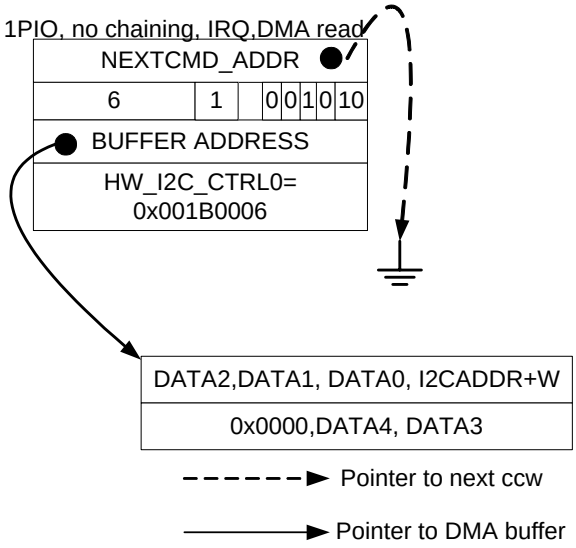


Figure 103. I^{2}C Writing Five Bytes

The DMA command is initialized to send six bytes to the I^{2}C controller and one word of PIO information to the HW_I2C_CTRL0 register.

Table 760. I^{2}C Transfer When the Master Transmits 5 Bytes of Data to the Slave

ST	SAD+W	SAK	DATA	SAK	DATA	MAK	D	A	D	A	D	A	D	A	DATA	NMAK	SP

The following C code is used to send a five-byte transmission:

```
// SEND: start, 0x56, 0x01,0x02,0x03,0x04,0x05,stop
//-------------------------------------------------------
#define I2C_CHANNEL_NUM 3
// dma buffer of 6 bytes (i2c address + 5 data bytes)
static reg32_t I2C_DATA_BUFFER[2]=
{
   0x03020156,  //slave address 56+W
   0x00000504   // last two data bytes
};
// DMA command chain
const static reg32_t  I2C_DMA_CMD[4] =
{
    (reg32_t)  I2C_DMA_CMD2,
    (BF_APBX_CHn_CMD_XFER_COUNT(6) |
     BF_APBX_CHn_CMD_CMDWORDS(1)   |
     BF_APBX_CHn_CMD_WAIT4ENDCMD(1)|
     BF_APBX_CHn_CMD_CHAIN(0)      |
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)),
    (reg32_t) &eeprom_command_buffer[0],
  BF_I2C_CTRL0_POST_SEND_STOP(BV_I2C_CTRL0_POST_SEND_STOP__SEND_STOP)|
  BF_I2C_CTRL0_PRE_SEND_START(BV_I2C_CTRL0_PRE_SEND_START__SEND_START) |
  BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE__MASTER)           |
  BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION__TRANSMIT)             |
```

```
  BF_I2C_CTRL0_XFER_COUNT(6)
};



void SendFiveBytes(){
 // Reset the APBX dma channels associated with I2C.
 reset_mask = BF_APBX_CTRL0_RESET_CHANNEL((1 << I2C_CHANNEL_NUM));
 HW_APBX_CTRL0_SET(reset_mask);

 // Poll for reset to clear the channel.
 for (retries = 0; retries < RESET_TIMEOUT; retries++)
      if ((reset_mask & HW_APBX_CTRL0_RD()) == 0)
             break;
 if( retries == RESET_TIMEOUT) exit(1);

 // Setup dma channel configuration.
 BF_WRn(APBX_CHn_NXTCMDAR,I2C_CHANNEL_NUM,
        CMD_ADDR,(reg32_t) I2C_DMA_CMD);
 BF_WR(APBX_CTRL1, CH3_CMDCMPLT_IRQ, 0); // clear interrupt

 // Start the dma channel by incrementing semaphore.
 BF_WRn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, INCREMENT_SEMA, 1);

 // Poll for the semaphore to decrement to zero on the DMA channel.
 for (retries = 0; retries < SEMAPHORE_TIMEOUT; retries++)
      if (0 == BF_RDn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, PHORE))
             break;

 // a frame with one byte of address and five bytes of data was just sent
}
```

21.5.2. Reading 256 bytes from an EEPROM

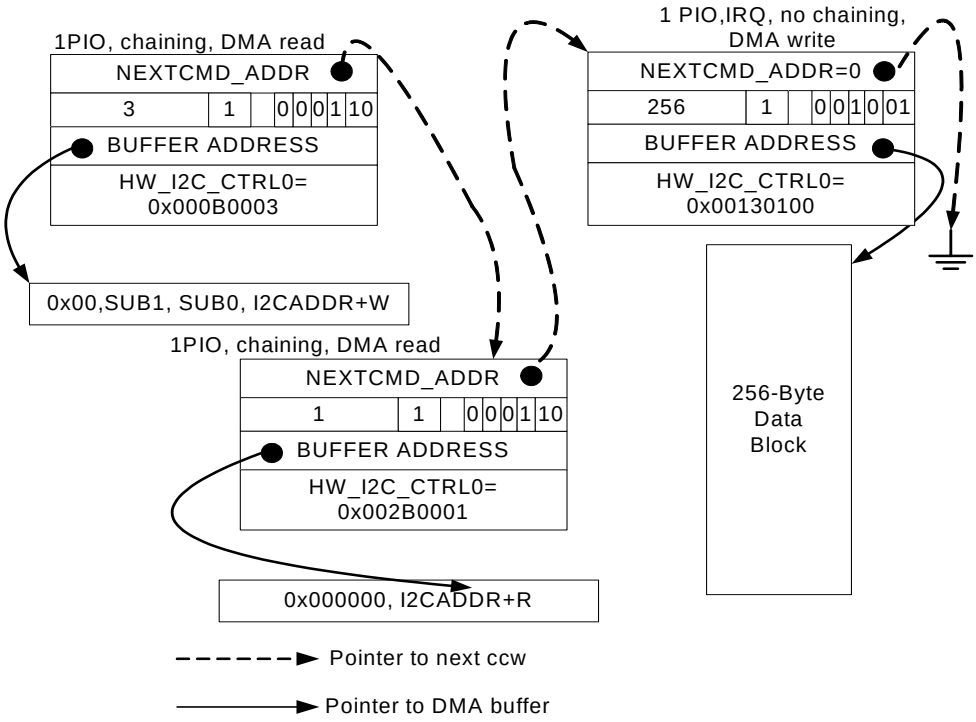


Figure 104. I²C Reading 256 Bytes from an EEPROM

```
//-----------------------------------------------------------------------
// dma buffers to hold i2c command string for slave addres+W plus sub0,
// sub 1 and the second command, a slave address+R
// eePROM write address == 0xA0,  read address == 0xA1
//-----------------------------------------------------------------------
unsigned char eeprom_command_buffer[4] = {0xA0,0x34,0x12,0xA1};

//-----------------------------------------------------------------------
// I2C DMA chain
//-----------------------------------------------------------------------
const static reg32_t  I2C_DMA_CMD3[4] =
{
    0x0,
    (BF_APBX_CHn_CMD_XFER_COUNT(256) |
     BF_APBX_CHn_CMD_SEMAPHORE(1)    |
     BF_APBX_CHn_CMD_CMDWORDS(1)     |
     BF_APBX_CHn_CMD_CHAIN(0)        |            // last command
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)),
    (reg32_t) &eeprom_command_buffer[3],
    BF_I2C_CTRL0_POST_SEND_STOP(BV_I2C_CTRL0_POST_SEND_STOP__SEND_STOP)|
    BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE__MASTER)         |
    BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION__RECEIVE)            |
    BF_I2C_CTRL0_XFER_COUNT(256)
};
const static reg32_t  I2C_DMA_CMD2[4] =
{
    (reg32_t)  I2C_DMA_CMD3,
    (BF_APBX_CHn_CMD_XFER_COUNT(1) |
     BF_APBX_CHn_CMD_SEMAPHORE(1)  |
     BF_APBX_CHn_CMD_CMDWORDS(1)   |
     BF_APBX_CHn_CMD_WAIT4ENDCMD(1)|
     BF_APBX_CHn_CMD_CHAIN(1)      |
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)),
    (reg32_t) &eeprom_command_buffer[3],
    BF_I2C_CTRL0_RETAIN_CLOCK(BV_I2C_CTRL0_RETAIN_CLOCK__HOLD_LOW)|

   BF_I2C_CTRL0_PRE_SEND_START(BV_I2C_CTRL0_PRE_SEND_START__SEND_START)|
   BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE__MASTER)          |
   BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION__TRANSMIT)            |
   BF_I2C_CTRL0_XFER_COUNT(1)
};
const static reg32_t  I2C_DMA_CMD1[4] =
{
    (reg32_t)  I2C_DMA_CMD2,
    (BF_APBX_CHn_CMD_XFER_COUNT(3) |
     BF_APBX_CHn_CMD_CMDWORDS(1)   |
     BF_APBX_CHn_CMD_WAIT4ENDCMD(1)|
     BF_APBX_CHn_CMD_CHAIN(1)      |
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)),
    (reg32_t) &eeprom_command_buffer[0],
  BF_I2C_CTRL0_PRE_SEND_START(BV_I2C_CTRL0_PRE_SEND_START__SEND_START) |
  BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE__MASTER)           |
  BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION__TRANSMIT)             |
  BF_I2C_CTRL0_XFER_COUNT(3)
};


//////////////////////////////////////////////////////
//Read256BytesFromEEPROM returns 1 for errors and 0 for OK
//////////////////////////////////////////////////////
int Read256BytesFromEEPROM(unsigned short usAddress){
 // insert eePROM addres param into dma command buffer
  I2C_CMD_BUFFER[1] = (unsigned char) (usAddress      &0x00ff);
  I2C_CMD_BUFFER[2] = (unsigned char) ((usAddress>>8) &0x00ff);

 // Reset the APBX dma channels associated with I2C.
 reset_mask = BF_APBX_CTRL0_RESET_CHANNEL((1 << I2C_CHANNEL_NUM));
 HW_APBX_CTRL0_SET(reset_mask);

 // Poll for reset to clear the channel.
 for (retries = 0; retries < RESET_TIMEOUT; retries++)
      if ((reset_mask & HW_APBX_CTRL0_RD()) == 0)
              break;
 if (retries == RESET_TIMEOUT)exit(1);
 // Setup dma channel configuration.
 BF_WRn(APBX_CHn_NXTCMDAR,I2C_CHANNEL_NUM,
        CMD_ADDR,(reg32_t) I2C_DMA_CMD_SUBADDR);
```

```
 BF_WR(APBX_CTRL1, CH3_CMDCMPLT_IRQ, 0);

 // Start the dma channel by incrementing semaphore.
 BF_WRn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, INCREMENT_SEMA, 1);

 // Poll for the semaphore to decrement to zero on the DMA channel.
 for (retries = 0; retries < SEMAPHORE_TIMEOUT; retries++)}
      if (0 == BF_RDn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, PHORE))
             break;
      if (1 == HW_I2C_CTRL1.MASTER_LOSS_IRQ)        return 1; // error
      if (1 == HW_I2C_CTRL1.OVERSIZE_XFER_TERM_IRQ) return 1; // error
      if (1 == HW_I2C_CTRL1.NO_SLAVE_ACK_IRQ)       return 1; // error
      if (1 == HW_I2C_CTRL1.EARLY_TERM_IRQ)         return 1; // error

 }
 ferreters == SEMAPHORE_TIMEOUT) exit(2);
 // the 256 bytes were read from the eePROM so return with no Error

 return 0;

}
```

21.6. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

21.7. Programmable Registers

The following registers describe the programming interface for the slave and master I^2C controller.

21.7.1. I2C Control Register 0 Description

The I2C Control Register specifies the reset state and the command and transfer size information for the I2C controller.

HW_I2C_CTRL0 0x80058000
HW_I2C_CTRL0_SET 0x80058004
HW_I2C_CTRL0_CLR 0x80058008
HW_I2C_CTRL0_TOG 0x8005800C

Table 761. HW_I2C_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RUN	RSVD1	PRE_ACK	ACKNOWLEDGE	SEND_NAK_ON_LAST	PIO_MODE	MULTI_MASTER	CLOCK_HELD	RETAIN_CLOCK	POST_SEND_STOP	PRE_SEND_START	SLAVE_ADDRESS_ENABLE	MASTER_MODE	DIRECTION	XFER_COUNT															

Table 762. HW_I2C_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. RUN = 0x0 Allow I2C to operate normally. RESET = 0x1 Hold I2C in reset.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block. RUN = 0x0 Allow I2C to operate normally. NO_CLKS = 0x1 Do not clock I2C gates in order to minimize power consumption.
29	**RUN**	RW	0x0	Set this bit to one to enable the I2C controller operation. This bit is automatically set by DMA commands that write to CTRL1 after the last PIO write of the DMA command. For soft DMA operation, software can set this bit to enable the controller. HALT = 0x0 No I2C command in progress. RUN = 0x1 Process a slave or master I2C command.
28	**RSVD1**	RO	0x0	Always set this bit field to zero.
27	**PRE_ACK**	RW	0x0	Reserved for SigmaTel use.
26	**ACKNOWLEDGE**	RW	0x0	Set this bit to one to cause a pending acknowledge bit (prior to DMA transfer) to be acknowledged. Set it to zero to NAK the pending acknowledge bit. This bit is set to one by the slave search engine if the criteria is met for acknowledging a slave address. Software can reset the bit to slave-not-acknowledge the address. This bit defines the state of the I2C_DATA line during the address acknowledge bit time. The slave search engine holds the clock at this point for a software decision. This bit has no effect when the presend start option is selected. SNAK = 0x0 Slave not acknowledge when the held clock is released. ACK = 0x1 Slave acknowledge when the held clock is released.
25	**SEND_NAK_ON_LAST**	RW	0x0	Set this bit to one to cause the DMA transfer engine to send a NAK on the last byte. ACK_IT = 0x0 Send an ACK on the last byte received. NAK_IT = 0x1 Send a NAK on the last byte received.
24	**PIO_MODE**	RW	0x0	Set this bit to one to enable PIO mode of operation for the I2C master. One can preload up to four bytes into HW_I2C_DATA register before setting the RUN bit. The state machine will not attempt to use the DMA for master transmit operation. The normal start and stop conditions can be sent and the clock can be held at the end of the transfer, if desired. NOTE: All receive operations must use the DMA mode, not the PIO mode.
23	**MULTI_MASTER**	RW	0x0	Set this bit to one to enable the master state machine to monitor the start conditions generated by other masters. The bus is assumed to be busy from the first start condition generated by another master until a stop condition is generated. SINGLE = 0x0 Assume we are the only master. MULTIPLE = 0x1 Enable multiple master bus busy monitoring from start detects.

Table 762. HW_I2C_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	**CLOCK_HELD**	RW	0x0	This bit is set to one by the I2C controller state machines. It holds the I2C clock line low until cleared. It must be cleared by firmware, either by CPU instructions or DMA PIO transactions. It is set high when a slave address is matched by the slave controller. It is also set high at the end of a master or slave transaction that had the RETAIN_CLOCK bit set high. Software should not set this bit to one. RELEASE = 0x0 Release the clock line. HELD_LOW = 0x1 The clock line is currently being held low.
21	**RETAIN_CLOCK**	RW	0x0	Set this bit to one to retain the clock at the end of this transaction. This has the effect of holding the clock low until the start of the next transaction. RELEASE = 0x0 Release the clock line after this data transfer. HOLD_LOW = 0x1 Hold the clock line low after this data transfer.
20	**POST_SEND_STOP**	RW	0x0	Set this bit to one to send a stop condition after transferring the data associated with this transaction. This bit is automatically cleared by the hardware after the operation has been performed. NO_STOP = 0x0 Do not send a stop condition before this transaction. SEND_STOP = 0x1 Send a stop condition before this transaction.
19	**PRE_SEND_START**	RW	0x0	Set this bit to one to send a start condition before transferring the data associated with this transaction. This bit is automatically cleared by the hardware after the operation has been performed. NO_START = 0x0 Do not send a start condition before this transaction. SEND_START = 0x1 Send a start condition before this transaction.
18	**SLAVE_ADDRESS_ENABLE**	RW	0x0	Set this bit to one to enable the slave address decoder. When an address match occurs, the I2C bus clock is frozen, by setting HW_I2C_CTRL0_CLOCK_HELD, and an interrupt is generated. DISABLED = 0x0 Disable the slave address decoder. ENABLED = 0x1 Enable the slave address decoder.
17	**MASTER_MODE**	RW	0x0	Set this bit to one to select master mode. Set it zero to select slave mode. SLAVE = 0x0 Operate in slave mode. MASTER = 0x1 Operate in master mode.
16	**DIRECTION**	RW	0x0	Set this bit to one to select an I2C transmit operation in either slave or master mode. XMIT = write in master mode, read in slave mode. Set this bit to zero to select an I2C receive operation in either slave or master mode. RECEIVE = 0x0 I2C receive operation for slave or master. TRANSMIT = 0x1 I2C transmit operation for slave or master.
15:0	**XFER_COUNT**	RW	0x0000	Number of bytes to transfer. This field decrements as bytes are transferred.

DESCRIPTION:

This register is either written by the DMA or the CPU depending on the state of an I2C transaction.

EXAMPLE:

```
// turn off soft reset and clock gating
HW_I2C_CTRL0_CLR(BM_I2C_CTRL0_SFTRST | BM_I2C_CTRL0_CLKGATE);
```

21.7.2. I2C Timing Register 0 Description

The timing for various phases of I2C controller commands are further defined by fields in the I2C Timing Register 0.

HW_I2C_TIMING0 0x80058010
HW_I2C_TIMING0_SET 0x80058014
HW_I2C_TIMING0_CLR 0x80058018
HW_I2C_TIMING0_TOG 0x8005801C

Table 763. HW_I2C_TIMING0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2						HIGH_COUNT										RSVD1						RCV_COUNT									

Table 764. HW_I2C_TIMING0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSVD2**	RO	0x0	Always set this bit field to zero.
25:16	**HIGH_COUNT**	RW	0x78	Load this bit field with the APBX clock count for the high period of the I2C clock.
15:10	**RSVD1**	RO	0x0	Always set this bit field to zero.
9:0	**RCV_COUNT**	RW	0x30	Load this bit field with the APBX clock count for capturing read data after the I2C clock goes high.

DESCRIPTION:

This register is primarily used for clock and timing generation.

EXAMPLE:

```
  HW_I2C_TIMING0_WR(0x00780030); // high time = 120 clocks, read bit at 48 for 95 kHz at 24
MHz
  HW_I2C_TIMING0_WR(0x000F0007); // high time = 15 clocks, read bit at 7 for 400 kHz at 24
MHz
```

21.7.3. I2C Timing Register 1 Description

The timing for various phases of I2C controller commands are further defined by fields in the I2C Timing Register 1.

HW_I2C_TIMING1 0x80058020
HW_I2C_TIMING1_SET 0x80058024
HW_I2C_TIMING1_CLR 0x80058028
HW_I2C_TIMING1_TOG 0x8005802C

Table 765. HW_I2C_TIMING1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2						LOW_COUNT										RSVD1						XMIT_COUNT									

Table 766. HW_I2C_TIMING1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSVD2**	RO	0x0	Always set this bit field to zero.
25:16	**LOW_COUNT**	RW	0x80	Load this bit field with the APBX clock count for the low period of the I2C clock.
15:10	**RSVD1**	RO	0x0	Always set this bit field to zero.
9:0	**XMIT_COUNT**	RW	0x30	Load this bit field with the APBX clock count for changing transmitted data after the I2C clock goes low. Set this value to produce valid I2C setup and hold times at the desired bit rate for the current APBX clock rate.

DESCRIPTION:

This register is primarily used for clock and timing generation.

EXAMPLE:

```
HW_I2C_TIMING1_WR(0x00800030); // low time at 128, write bit at 48 for 95 kHz at 24 MHz
HW_I2C_TIMING1_WR(0x001F000F); // low time at 31, write bit at 15 for 400 kHz at 24 MHz
```

21.7.4. I2C Timing Register 2 Description

The timing for various phases of I2C controller commands are further defined by fields in the I2C Timing Register 2.

HW_I2C_TIMING2 0x80058030
HW_I2C_TIMING2_SET 0x80058034
HW_I2C_TIMING2_CLR 0x80058038
HW_I2C_TIMING2_TOG 0x8005803C

Table 767. HW_I2C_TIMING2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2						BUS_FREE										RSVD1						LEADIN_COUNT									

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

Table 768. HW_I2C_TIMING2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSVD2**	RO	0x0	Always set this bit field to zero.
25:16	**BUS_FREE**	RW	0x30	Load this bit field with the APBX clock count for delaying the transition to the bus idle state after entering stop state in the clock generator.
15:10	**RSVD1**	RO	0x0	Always set this bit field to zero.
9:0	**LEADIN_COUNT**	RW	0x30	Load this bit field with the APBX clock count for delaying the rising edge of I2C_SCK after the kick.

DESCRIPTION:

This register is primarily used for clock and timing generation.

EXAMPLE:

```
HW_I2C_TIMING2_WR(0x0015000d); // bus free count of 21 lead in count of 13
```

21.7.5. I2C Control Register 1 Description

The I2C controller command is further defined by fields in this control extension register. The I2C Control Register 1 is where the I2C slave address is specified. Fast or normal mode is selected here.

HW_I2C_CTRL1 0x80058040
HW_I2C_CTRL1_SET 0x80058044
HW_I2C_CTRL1_CLR 0x80058048
HW_I2C_CTRL1_TOG 0x8005804C

Table 769. HW_I2C_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1							BCAST_SLAVE_EN	SLAVE_ADDRESS_BYTE								BUS_FREE_IRQ_EN	DATA_ENGINE_CMPLT_IRQ_EN	NO_SLAVE_ACK_IRQ_EN	OVERSIZE_XFER_TERM_IRQ_EN	EARLY_TERM_IRQ_EN	MASTER_LOSS_IRQ_EN	SLAVE_STOP_IRQ_EN	SLAVE_IRQ_EN	BUS_FREE_IRQ	DATA_ENGINE_CMPLT_IRQ	NO_SLAVE_ACK_IRQ	OVERSIZE_XFER_TERM_IRQ	EARLY_TERM_IRQ	MASTER_LOSS_IRQ	SLAVE_STOP_IRQ	SLAVE_IRQ

Table 770. HW_I2C_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSVD1**	RO	0x0	Always set this bit field to zero.
24	**BCAST_SLAVE_EN**	RW	0x0	Set this bit to one to enable the slave address search machine to look for both a match to the programmed slave address, as well as a match to the broadcast address of all zeroes. NO_BCAST = 0x0 Do not watch for broadcast address while matching programmed slave address. WATCH_BCAST = 0x1 Watch for the all zeroes broadcast address while matching programmed slave address.
23:16	**SLAVE_ADDRESS_BYTE**	RW	0x86	Slave Address Byte. Note that the slave address is only seven bits long. The slave address search state machine will respond to either a read or a write command issued to the seven-bit address. Set the LSB (bit 0) to one to match ALL 7 bit I2C addresses.
15	**BUS_FREE_IRQ_EN**	RW	0x0	Set this bit to one to enable bus free interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.
14	**DATA_ENGINE_CMPLT_IRQ_EN**	RW	0x0	Set this bit to one to enable data engine complete interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.
13	**NO_SLAVE_ACK_IRQ_EN**	RW	0x0	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.
12	**OVERSIZE_XFER_TERM_IRQ_EN**	RW	0x0	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.
11	**EARLY_TERM_IRQ_EN**	RW	0x0	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.
10	**MASTER_LOSS_IRQ_EN**	RW	0x0	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.
9	**SLAVE_STOP_IRQ_EN**	RW	0x0	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.

Table 770. HW_I2C_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**SLAVE_IRQ_EN**	RW	0x0	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. The corresponding HW_IRQ_STAT_SLAVE_IRQ interrupt bit is set by the slave search engine to indicate that it has stopped searching due to an address match or error. DISABLED = 0x0 No Interrupt Request Pending. ENABLED = 0x1 Interrupt Request Pending.
7	**BUS_FREE_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller because the bus has become free. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that the I2C bus, which was busy, has just become free. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
6	**DATA_ENGINE_CMPLT_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller because the data engine transfer has completed. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that the data engine has completed a DMA transfer in either master or slave mode. This notification is useful for PIO mode master write (transmit) or slave read (transmit) operations, i.e., data engine transmit operations. PIO receive operations are not supported. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
5	**NO_SLAVE_ACK_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller because the slave addressed by a master transfer did not respond with an acknowledge to its slave address. This bit is cleared by software by writing a one to its SCT clear address. NOTE: In master mode, the data engine checks the acknowledge of the first byte transmitted after a start condition is sent. If the slave does not acknowledge this specific byte, then this interrupt status bit is set. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
4	**OVERSIZE_XFER_TERM_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that a master DMA transfer did not complete by the end of the transfer size. This is indicated by the slave acknowledging the last byte of a write transfer instead of NAKing it. The master should then send additional bytes of data if desired. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

Table 770. HW_I2C_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**EARLY_TERM_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that a master write transfrom from the STMP36xx to a slave device was NAKed by the slave before the transfer was completed. In slave mode, it indicates that the master NAKed a byte transmitted by the slave causing early termination of the expected transfer. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
2	**MASTER_LOSS_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that a master read or write transaction lost an arbitration with another master. Master loss is indicated by the master attempting to transmit a one to the bus at the same time as another master writes a zero. The wired and bus produces a zero on the bus which is detected by the lossing master. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
1	**SLAVE_STOP_IRQ**	RW	0x0	This bit is set to indicate that an I2C Stop Condition was received by the slave address search engine after it had found a start command addressed to its slave address. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
0	**SLAVE_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This bit is set by the slave search engine to indicate that it has stopped searching due to an address match or error. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

DESCRIPTION:

This control register is primarily used for interrupt management. It also controls the special slave address matching mode. In addition, it controls the protocol speed, i.e., fast or 400-kHz versus normal or 100-kHz operation.

EXAMPLE:

```
HW_I2C_CTRL1_CLR(BM_I2C_CTRL1_SLAVE_IRQ); // clear the slave interrupt
```

21.7.6. I2C Status Register Description

The I2C controller reports status information in the I2C Status Register.

HW_I2C_STAT 0x80058050

Table 771. HW_I2C_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MASTER_PRESENT	SLAVE_PRESENT	ANY_ENABLED_IRQ	RSVD1					RCVD_SLAVE_ADDR								SLAVE_ADDR_EQ_ZERO	SLAVE_FOUND	SLAVE_SEARCHING	DATA_ENGINE_DMA_WAIT	BUS_BUSY	CLK_GEN_BUSY	DATA_ENGINE_BUSY	SLAVE_BUSY	BUS_FREE_IRQ_SUMMARY	DATA_ENGINE_CMPLT_IRQ_SUMMARY	NO_SLAVE_ACK_IRQ_SUMMARY	OVERSIZE_XFER_TERM_IRQ_SUMMARY	EARLY_TERM_IRQ_SUMMARY	MASTER_LOSS_IRQ_SUMMARY	SLAVE_STOP_IRQ_SUMMARY	SLAVE_IRQ_SUMMARY

Table 772. HW_I2C_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**MASTER_PRESENT**	RO	0x1	This read-only bit indicates that the I2C master function is present when it reads back a one. This I2C function is not available on a device that returns a zero for this bit field. UNAVAILABLE = 0x0 I2C is not present in this product. AVAILABLE = 0x1 I2C is present in this product.
30	**SLAVE_PRESENT**	RO	0x1	This read-only bit indicates that the I2C slave function is present when it reads back a one. This I2C function is not available on a device that returns a zero for this bit field. UNAVAILABLE = 0x0 I2C is not present in this product. AVAILABLE = 0x1 I2C is present in this product.
29	**ANY_ENABLED_IRQ**	RO	0x0	This read-only bit indicates that the I2C controller has at least one enable interrupt requesting service. It is the logic OR of all of the IRQ summary bits. NO_REQUESTS = 0x0 No enabled interrupts are requesting service. AT_LEAST_ONE_REQUEST = 0x1 At least one of the summary interrupt bits is set.
28:24	**RSVD1**	RO	0x0	Always set this bit field to zero.
23:16	**RCVD_SLAVE_ADDR**	RO	0x00	This read-only byte indicates that the state of the slave I2C address byte received, including the read/write bit received from an address byte that matched our slave address.

Table 772. HW_I2C_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15	**SLAVE_ADDR_EQ_ZERO**	RO	0x0	This read-only bit indicates that the I2C slave function was searching for a transaction that matches the current slave address. When set to one, it indicates that an address match was found for the exact adderss 0x00. ZERO_NOT_MATCHED = 0x0 I2C slave search did not match a zero. WAS_ZERO = 0x1 I2C has found an address match against address 0x00.
14	**SLAVE_FOUND**	RO	0x0	This read-only bit indicates that the I2C slave function was searching for a transaction that matches the current slave address. When set to one, it indicates that an address match was found and the I2C clock is frozen by the slave search. This bit is cleared by starting the appropriate slave DMA transfer or restarting a slave search. IDLE = 0x0 I2C slave search is idle. WAITING = 0x1 I2C has found an address match and is holding the I2C clock line low.
13	**SLAVE_SEARCHING**	RO	0x0	This read-only bit indicates that the I2C slave function is searching for a transaction that matches the current slave address. IDLE = 0x0 I2C slave search is idle. ACTIVE = 0x1 I2C is actively searching for an address match.
12	**DATA_ENGINE_DMA_WAIT**	RO	0x0	This read-only bit is set to one when the data engine is waitng for data from a DMA device. This bit can be used to transmit short I2C transactions without using a DMA channel. This generally works for up to three data bytes transmitted with one address byte. CONTINUE = 0x0 I2C master is not waiting on data from the DMA. WAITING = 0x1 I2C master is waiting on data from the DMA.
11	**BUS_BUSY**	RO	0x0	This read-only bit indicates that the I2C bus is busy with a transaction. It is set by a start condition and reset by a detected stop condition. IDLE = 0x0 I2C bus is idle, i.e., reset state or at least one stop condition detected. BUSY = 0x1 I2C bus is busy, i.e., at least one start condition has been detected.
10	**CLK_GEN_BUSY**	RO	0x0	This read-only bit indicates that the I2C clock generator is busy with a transaction. IDLE = 0x0 I2C clock generator is idle. BUSY = 0x1 I2C clock generator is busy performing a command.
9	**DATA_ENGINE_BUSY**	RO	0x0	This read-only bit indicates that the I2C data transfer engine is busy with a data transmit or recieve opertion. In addition, it can be busy, as a master, sending a start or stop condition. IDLE = 0x0 I2C Data Engine is idle. BUSY = 0x1 I2C is Data Engine busy performing a data transfer.

Table 772. HW_I2C_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**SLAVE_BUSY**	RO	0x0	This read-only bit indicates that the I2C slave address search engine is busy with a transaction. This bit will go high when an address search is started and will remain high until the slave search engine returns to its idle state. IDLE = 0x0 I2C slave search engine is idle. BUSY = 0x1 I2C slave search engine is busy searching for an address match.
7	**BUS_FREE_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
6	**DATA_ENGINE_CMPLT_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
5	**NO_SLAVE_ACK_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
4	**OVERSIZE_XFER_TERM_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
3	**EARLY_TERM_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
2	**MASTER_LOSS_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

Table 772. HW_I2C_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**SLAVE_STOP_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
0	**SLAVE_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

DESCRIPTION:

The status register provides read-only access to the function presence bits, as well as the busy indicators for the slave and master state machines.

EXAMPLE:

```
   while(HW_I2C_STAT.SLAVE_BUSY != BV_I2C_STAT_SLAVE_BUSY__IDLE_VAL);// then wait till it
finishes
```

21.7.7. I2C Controller DMA Read and Write Data Register Description

The I2C Controller DMA Read and Write Data Register is the target for both source and destination DMA transfers. This register is backed by an eight-deep FIFO.

HW_I2C_DATA 0x80058060

Table 773. HW_I2C_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 774. HW_I2C_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x00000000	The source DMA channel writes to this address. The destination DMA channel reads from this address.

DESCRIPTION:

DMA reads and writes are directed to this register.

EXAMPLE:

```
  The DMA data register is used by the DMA to read or write data from the I2C controller, as
mediated by the I2C controller`s DMA request signal.
```

21.7.8. I2C Device Debug Register 0 Description

The I2C Device Debug Register 0 provides a diagnostic view into the internal state machine and states of the I2C device.

HW_I2C_DEBUG0 0x80058070

HW_I2C_DEBUG0_SET 0x80058074
HW_I2C_DEBUG0_CLR 0x80058078
HW_I2C_DEBUG0_TOG 0x8005807C

Table 775. HW_I2C_DEBUG0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DMAREQ	DMAENDCMD	DMAKICK	TBD			DMA_STATE										START_TOGGLE	STOP_TOGGLE	GRAB_TOGGLE	CHANGE_TOGGLE	TESTMODE	SLAVE_HOLD_CLK	SLAVE_STATE									

Table 776. HW_I2C_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**DMAREQ**	RO	0x0	Read-only view of the toggle state of the DMA request signal.
30	**DMAENDCMD**	RO	0x0	Read-only view of the toggle state of the DMA End Command signal.
29	**DMAKICK**	RO	0x0	Read-only view of the toggle state of the DMA Kick signal.
28:26	**TBD**	RW	0x0	Reserved
25:16	**DMA_STATE**	RO	0x010	Current state of the DMA state machine.
15	**START_TOGGLE**	RO	0x0	Read-only view of the start detector. Toggles once for each detected start condition.
14	**STOP_TOGGLE**	RO	0x0	Read-only view of the stop detector. Toggles once for each detected stop condition.
13	**GRAB_TOGGLE**	RO	0x0	Read-only view of the grab receive data timing point. Toggles once for each read timing point, as delayed from rising clock.
12	**CHANGE_TOGGLE**	RO	0x0	Read-only view of the change transmit data timing point. Toggles once for each change transmit data timing point, as delayed from falling clock.
11	**TESTMODE**	RW	0x0	To be completed by designer.
10	**SLAVE_HOLD_CLK**	RO	0x0	Current State of the Slave Address Search FSM clock hold register.
9:0	**SLAVE_STATE**	RO	0x0000	Current State of the Slave Address Search FSM.

DESCRIPTION:

This register provides access to various internal states and controls that are used in diagnostic modes of operation.

EXAMPLE:

```
while(HW_I2C_DEBUG0.DMAREQ == old_dma_req_value); // wait for next dma request toggle
```

```
  old_dma_req_value = HW_I2C_DEBUG0.DMAREQ; // remember the new state of the dma request tog-
gle
```

21.7.9. I2C Device Debug Register 1 Description

The I2C Device Debug Register 1 provides a diagnostic view of the external bus and provides OE control for the clock and data.

HW_I2C_DEBUG1 0x80058080
HW_I2C_DEBUG1_SET 0x80058084
HW_I2C_DEBUG1_CLR 0x80058088
HW_I2C_DEBUG1_TOG 0x8005808C

Table 777. HW_I2C_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
I2C_CLK_IN	I2C_DATA_IN	RSVD4		DMA_BYTE_ENABLES				RSVD3	CLK_GEN_STATE							RSVD2					LST_MODE		LOCAL_SLAVE_TEST	RSVD1		FORCE_CLK_ON	FORCE_CLK_IDLE	FORCE_ARB_LOSS	FORCE_RCV_ACK	FORCE_I2C_DATA_OE	FORCE_I2C_CLK_OE

Table 778. HW_I2C_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**I2C_CLK_IN**	RO	0x1	A copy of the pad input signal for the I2C clock pad.
30	**I2C_DATA_IN**	RO	0x1	A copy of the pad input signal for the I2C data pad.
29:28	**RSVD4**	RO	0x0	Always set this bit field to zero.
27:24	**DMA_BYTE_ENABLES**	RO	0x0	A read-only view of the byte enables for HW_I2C_DATA register writes. These bits are used in the I2C DMA state machine to track the number of bytes written by the DMA. Individual bits are cleared as they are consummed.
23	**RSVD3**	RO	0x0	Always set this bit field to zero.
22:16	**CLK_GEN_STATE**	RO	0x0	A read-only view of the byte enables for HW_I2C_DATA register writes. These bits are used in the I2C DMA state machine to track the number of bytes written by the DMA. Individual bits are cleared as they are consummed.
15:11	**RSVD2**	RO	0x0	Always set this bit field to zero.
10:9	**LST_MODE**	RW	0x0	When in local slave test mode, this bit field defines the type of address generated for the slave. BCAST = 0x0 Broadcast, i.e., I2C address 0x00. MY_WRITE = 0x1 Send to my slave address with a RW bit equal 0. MY_READ = 0x2 Send to my slave address with a RW bit equal 1. NOT ME = 0x3 Send to an address that is not mine, i.e., bit four is complemented.

Table 778. HW_I2C_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**LOCAL_SLAVE_TEST**	RW	0x0	Writting a one to this bit places the slave in local test mode. One of three slave address can be sent in either read or write mode.
7:6	**RSVD1**	RO	0x0	Always set this bit field to zero.
5	**FORCE_CLK_ON**	RW	0x0	Writing a one to this bit will force the clock generator to send a continuous stream of clocks on the I2C bus.
4	**FORCE_CLK_IDLE**	RW	0x0	Writing a one to this bit will force the clock generator state machine to return to its idle state and stay there.
3	**FORCE_ARB_LOSS**	RW	0x0	Writing a one to this bit will force the appearance of an arbitration loss on the next one a master attempts to transmit.
2	**FORCE_RCV_ACK**	RW	0x0	Writing a one to this bit will force the appearance of a receive acknowledge to the byte level state machine at bit 9 of the transfer.
1	**FORCE_I2C_DATA_OE**	RW	0x0	Writting a one to this bit will force an output enable at the pad. The pad data line is tied to zero. Thus the I2C data line will either be hi-z or zero.
0	**FORCE_I2C_CLK_OE**	RW	0x0	Writing a one to this bit will force an output enable at the pad. The pad data line is tied to zero. Thus the I2C clock line will either be hi-z or zero.

DESCRIPTION:

This register provides access to the I2C clock and data pad cell state that are used in diagnostic modes of operation.

EXAMPLE:

```
while(HW_I2C_DEBUG1.I2C_CLK_IN == 0); // wait for I2C clock line to go high
```

I2C XML Revision: 1.55

22. APPLICATION UART

This chapter describes the Application UART included on the STMP36xx, how to operate it, and how to disable the FIFOs. Programmable registers are described in Section 22.4.

22.1. Overview

The Application UART:

- Performs serial-to-parallel conversion on data received from a peripheral device
- Performs parallel-to-serial conversion on data transmitted to the peripheral device
- Operates up to 1.5 Mb/s

The CPU or DMA controller reads and writes data and control/status information through the APBX interface. The transmit and receive paths are buffered with internal FIFO memories, enabling up to 16-bytes to be stored independently in both transmit and receive modes.

The Application UART includes a programmable baud rate generator that generates a common transmit and receive internal clock from the 24-MHz UART internal reference clock input UARTCLK, which is tied internally to XCLK.

It offers similar functionality to the industry-standard 16C550 UART device and supports baud rates of up to 1Mbits/s (in high-speed configuration). Figure 105 shows a block diagram of the Application UART. The Application UART operation and baud rate values are controlled by the line control register (HW_UARTAPP_LINECTRL).

The Application UART can generate a single combined interrupt, so that the output is asserted if any of the individual interrupts are asserted and unmasked. Interrupt sources include the receive (including timeout), transmit, modem status, and error conditions.

Two DMA channels are supported, one for transmit and one for receive.

If a framing, parity, or break error occurs during reception, the appropriate error bit is set and stored in the FIFO. If an overrun condition occurs, the overrun register bit is set immediately and FIFO data is prevented from being overwritten. You can program the FIFOs to be one-byte deep, providing a conventional double-buffered UART interface.

The modem status input signal Clear To Send (CTS) and output modem control line Request To Send (RTS) are supported. A programmable hardware flow control feature uses the nUARTCTS input and the nUARTRTS output to automatically control the serial data flow.

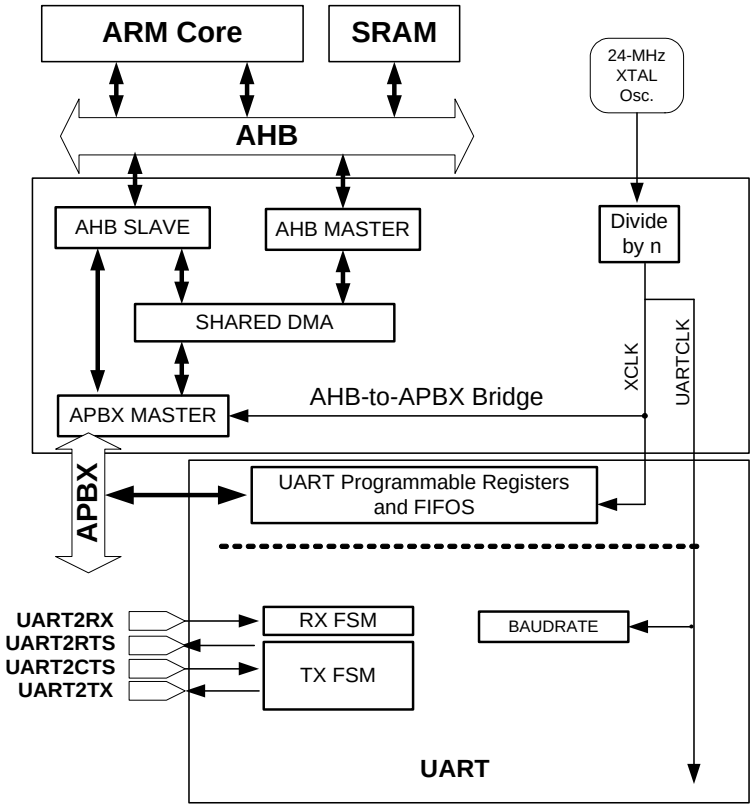


Figure 105. Application UART Block Diagram

22.2. Operation

Control data is written to the Application UART line control register. This register defines:

- Transmission parameters
- Word length
- Buffer mode
- Number of transmitted stop bits
- Parity mode
- Break generation
- Baud rate divisor

22.2.1. Fractional Baud Rate Divider

The baud rate divisor is calculated from the frequency of XCLK and the desired baud rate by using the following formula:

```
divisor = (XCLK * 4) / baud rate, rounded to the nearest integer
```

The divisor must be between 0x00000040 and 0x003FFFC0, inclusive. Program the lowest 6 bits of the divisor into BAUD_DIVFRAC, and the next 16 bits of the divisor into BAUD_DIVINT.

22.2.2. UART Character Frame

Figure 106 illustrates the UART character frame.

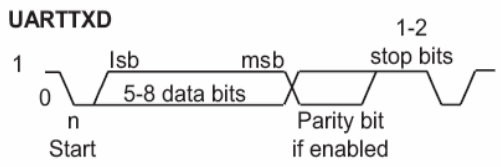


Figure 106. Application UART Character Frame

22.2.3. DMA Operation

The Application UART can generate a DMA request signal for interfacing with a Direct Memory Access (DMA) controller. Two DMA channels are supported, one for transmit and one for receive. Each channel has an associated 16-bit transfer counter for the number of bytes to transfer. Each DMA request is associated with one to four data bytes. For APBX DMA Channel 6, which is the UART RX channel, the first PIO word in the DMA command is CTRL1. However, for APBX DMA Channel 7, which is the UART TX, the first PIO word in a DMA command is CTRL1.

At the end of a receive DMA block transfer, the status register indicates any error conditions. If a timeout condition occurs in the middle of a receive DMA block transfer, then the UART sends dummy data to the DMA controller until the transfer counter is decremented to zero. A receive DMA can be setup to get the status of the previous receive DMA block transfer. The status indicates the amount of valid data bytes in the previous receive DMA block transfer.

22.2.4. Data Transmission or Reception

Data received or transmitted is stored in two 16-byte FIFOs, although the receive FIFO has an extra four bits per character for status information.

For transmission, data is written into the transmit FIFO. If the Application UART is enabled, it causes a data frame to start transmitting with the parameters indicated in UARTLCR_H. Data continues to be transmitted until there is no data left in the transmit FIFO.

The BUSY signal goes HIGH as soon as data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted HIGH while data is being transmitted. BUSY is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. BUSY can be asserted HIGH, even though the Application UART might no longer be enabled.

For each sample of data, three readings are taken and the majority value is kept. In the following paragraphs, the middle sampling point is defined, and one sample is taken on either side of it.

- When the receiver is idle (UARTRXD continuously 1, in the marking state) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by Baud16, begins running and data is sampled on the eighth cycle of that counter in normal UART mode to allow for the shorter logic 0 pulses (half way through a bit period).

- The start bit is valid if UARTRXD is still LOW on the eighth cycle of Baud16, otherwise a false start bit is detected and it is ignored. If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled.
- Lastly, a valid stop bit is confirmed if UARTRXD is HIGH, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word (see Table 779).

22.2.5. Error Bits

Three error bits are stored in bits [10:8] of the receive FIFO and are associated with a particular character. An additional error indicating an overrun error is stored in bit 11 of the receive FIFO.

22.2.6. Overrun Bit

The overrun bit is not associated with the character in the receive FIFO. The overrun error is set when the FIFO is full and the next character is completely received in the shift register. The data in the shift register is overwritten, but it is not written into the FIFO. When an empty location is available in the receive FIFO and another character is received, the state of the overrun bit is copied into the receive FIFO along with the received character. The overrun state is then cleared. Table 779 shows the bit functions of the receive FIFO.

Table 779. Receive FIFO Bit Functions

FIFO BIT	FUNCTION
11	Overrun indicator
10	Break error
9	Parity error
8	Framing error
7:0	Received data

22.2.7. Disabling the FIFOs

FIFOs can be disabled. In this case, the transmit and receive sides of the Application UART have one-byte holding registers (the bottom entry of the FIFOs). The overrun bit is set when a word has been received and the previous one was not yet read.

In this implementation, the FIFOs are not physically disabled, but the flags are manipulated to give the illusion of a one-byte register.

22.3. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

22.4. Programmable Registers

This section describes the Application UART's programable registers.

22.4.1. UART Receive DMA Control Register Description

The UART Receive DMA Control Register contains the dynamic information associated with the receive command.

HW_UARTAPP_CTRL0 0x8006C000
HW_UARTAPP_CTRL0_SET 0x8006C004
HW_UARTAPP_CTRL0_CLR 0x8006C008
HW_UARTAPP_CTRL0_TOG 0x8006C00C

Table 780. HW_UARTAPP_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSVD2	RUN	RSVD1			RX_SOURCE	RXTO_ENABLE	RXTIMEOUT							XFER_COUNT															

Table 781. HW_UARTAPP_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state.
30	**CLKGATE**	RW	0x1	Set this bit zero for normal operation. Setting this bit to one (default), gates all of the block level clocks off for miniminizing AC energy consumption.
29	**RSVD2**	RO	0x0	Reserved, read as zero, do not modify.
28	**RUN**	RW	0x0	Tell the UART to execute the RX DMA Command. The UART will clear this bit at the end of receive execution.
27:26	**RSVD1**	RO	0x0	Reserved, read as zero, do not modify.
25	**RX_SOURCE**	RW	0x0	Source of Receive Data. If this bit is set to 1, the status register will be the source of the DMA, otherwise RX data will be the source.
24	**RXTO_ENABLE**	RW	0x0	RXTIMEOUT Enable: If this bit is set to 0, the RX timeout will not affect receive DMA operation. If this bit is set to 1, a receive timeout will cause the receive DMA logic to terminate by filling the remaining DMA bytes with garbage data.

Table 781. HW_UARTAPP_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:16	**RXTIMEOUT**	RW	0x03	Receive Timeout Counter Value: number of 8-bit-time to wait before asserting timeout on the RX input. If the RXFIFO is not empty and the RX input is idle, then the watchdog counter will decrement each bit-time. Note 7-bit-time is added to the programmed value, so a value of zero will set the counter to 7-bit-time, a value of 0x1 gives 15-bit-time and so on. Also note that the counter is reloaded at the end of each frame, so if the frame is 10 bits long and the timeout counter value is zero, then timeout will occur (when FIFO is not empty) even if the RX input is not idle. The default value is 0x3 (31 bit-time).
15:0	**XFER_COUNT**	RW	0x00	Number of bytes to receive. This must be a multiple of 4.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

22.4.2. UART Transmit DMA Control Register Description

The UART Transmit DMA Control Register contains the dynamic information associated with the transmit command.

HW_UARTAPP_CTRL1 0x8006C010
HW_UARTAPP_CTRL1_SET 0x8006C014
HW_UARTAPP_CTRL1_CLR 0x8006C018
HW_UARTAPP_CTRL1_TOG 0x8006C01C

Table 782. HW_UARTAPP_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2			RUN	RSVD1												XFER_COUNT															

Table 783. HW_UARTAPP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD2**	RO	0x0	Reserved, read as zero, do not modify.
28	**RUN**	RW	0x0	Tell the UART to execute the TX DMA Command. The UART will clear this bit at the end of transmit execution.
27:16	**RSVD1**	RO	0x0	Reserved, read as zero, do not modify.
15:0	**XFER_COUNT**	RW	0x00	Number of bytes to transmit.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

22.4.3. UART Control Register Description

The UART Control Register contains configuation, including interrupt FIFO level select and the DMA control.

HW_UARTAPP_CTRL2 0x8006C020
HW_UARTAPP_CTRL2_SET 0x8006C024
HW_UARTAPP_CTRL2_CLR 0x8006C028
HW_UARTAPP_CTRL2_TOG 0x8006C02C

Table 784. HW_UARTAPP_CTRL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INVERT_RTS	INVERT_CTS	INVERT_TX	INVERT_RX	RSVD1	DMAONERR	TXDMAE	RXDMAE	RSVD2	RXIFLSEL			RSVD3	TXIFLSEL			CTSEN	RTSEN	OUT2	OUT1	RTS	DTR	RXE	TXE	LBE	RSVD4				SIRLP	SIREN	UARTEN

Table 785. HW_UARTAPP_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**INVERT_RTS**	RW	0x0	Invert RTS signal. If this bit is set to 1, the RTS output is inverted before transmitted.
30	**INVERT_CTS**	RW	0x0	Invert CTS signal. If this bit is set to 1, the CTS input is inverted before sampled.
29	**INVERT_TX**	RW	0x0	Invert TX signal. If this bit is set to 1, the TX output is inverted before transmitted.
28	**INVERT_RX**	RW	0x0	Invert RX signal. If this bit is set to 1, the RX input is inverted before sampled.
27	**RSVD1**	RO	0x0	Reserved, do not modify, read as zero.
26	**DMAONERR**	RW	0x0	DMA On Error. If this bit is set to 1, receive dma will terminate on error. (Cmd_end signal may not be asserted when this occurs.)
25	**TXDMAE**	RW	0x0	Transmit DMA Enable. Data Register can be loaded with up to 4 bytes per write. TXFIFO must be enabled in TXDMA mode.
24	**RXDMAE**	RW	0x0	Receive DMA Enable. Data Register can be contain up to 4 bytes per read. RXFIFO must be enabled in RXDMA mode.
23	**RSVD2**	RO	0x0	Reserved, do not modify, read as zero.

Table 785. HW_UARTAPP_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22:20	**RXIFLSEL**	RW	0x2	Receive Interrupt FIFO Level Select. The trigger points for the receive interrupt are as follows: NOT_EMPTY = 0x0 Trigger on FIFO not empty, i.e., at least 1 of 16 entries. ONE_QUARTER = 0x1 Trigger on FIFO full to at least 4 of 16 entries. ONE_HALF = 0x2 Trigger on FIFO full to at least 8 of 16 entries. THREE_QUARTERS = 0x3 Trigger on FIFO full to at least 12 of 16 entries. SEVEN_EIGHTHS = 0x4 Trigger on FIFO full to at least 14 of 16 entries. INVALID5 = 0x5 Reserved. INVALID6 = 0x6 Reserved. INVALID7 = 0x7 Reserved.
19	**RSVD3**	RO	0x0	Reserved, do not modify, read as zero.
18:16	**TXIFLSEL**	RW	0x2	Transmit Interrupt FIFO Level Select. The trigger points for the transmit interrupt are as follows: EMPTY = 0x0 Trigger on FIFO empty, i.e., no entries. ONE_QUARTER = 0x1 Trigger on FIFO less than 4 of 16 entries. ONE_HALF = 0x2 Trigger on FIFO less than 8 of 16 entries. THREE_QUARTERS = 0x3 Trigger on FIFO less than 12 of 16 entries. SEVEN_EIGHTHS = 0x4 Trigger on FIFO less than 14 of 16 entries. INVALID5 = 0x5 Reserved. INVALID6 = 0x6 Reserved. INVALID7 = 0x7 Reserved.
15	**CTSEN**	RW	0x0	CTS Hardware Flow Control Enable. If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the nUARTCTS signal is asserted.
14	**RTSEN**	RW	0x0	RTS Hardware Flow Control Enable. If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested when there is space in the receive FIFO for it to be received. The FIFO space is controlled by RXIFLSEL value.
13	**OUT2**	RW	0x0	This bit is the complement of the UART Out2 (nUARTOut2) modem status output. (Unsupported in STMP3600.) That is, when the bit is programmed to a 1, the output is 0. For DTE, this can be used as Ring Indicator (RI).
12	**OUT1**	RW	0x0	This bit is the complement of the UART Out1 (nUARTOut1) modem status output. (Unsupported in STMP3600.) That is, when the bit is programmed to a 1, the output is 0. For DTE, this can be used as Data Carrier Detect (DCD).
11	**RTS**	RW	0x0	Request To Send. Software can manually control the nUARTRTS pin via this bit when RTSEN = 0. This bit is the complement of the UART request to send (nUARTRTS) modem status output. That is, when the bit is programmed to a 1, the output is 0.
10	**DTR**	RW	0x0	Data Transmit Ready. This bit is the complement of the UART data transmit ready (nUARTDTR) modem status output. (Unsupported in STMP3600.) That is, when the bit is programmed to a 1, the output is 0.

Table 785. HW_UARTAPP_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9	**RXE**	RW	0x1	Receive Enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for either UART signals or SIR signals according to the setting of SIR Enable (SIREN, bit 1). When the UART is disabled in the middle of reception, it completes the current character before stopping.
8	**TXE**	RW	0x1	Transmit Enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for either UART signals or SIR signals according to the setting of SIR Enable (SIREN, bit 1). When the UART is disabled in the middle of transmission, it completes the current character before stopping.
7	**LBE**	RW	0x0	Loop Back Enable. If this bit is set to 1 and the SIR Enable bit is set to 1 and the test register TCR bit 2 (SIRTEST) is set to 1, then the nSIROUT path is inverted and fed through to the SIRIN path. The SIRTEST bit in the test register must be set to 1 to override the normal half-duplex SIR operation. This must be the requirement for accessing the test registers during normal operation, and SIRTEST must be cleared to 0 when loopback testing is finished. This feature reduces the amount of external coupling required during system test. If this bit is set to 1 and the SIRTEST bit is set to 0, the UARTTXD path is fed through to the UARTRXD path. In either SIR mode or normal mode, when this bit is set, the modem outputs are also fed through to the modem inputs.
6:3	**RSVD4**	RO	0x0	Reserved, do not modify, read as zero.
2	**SIRLP**	RW	0x0	IrDA SIR Low Power Mode. This bit selects the IrDA encoding mode. (Unsupported in STMP3600.) If this bit is cleared to 0, low-level bits are transmitted as an active high pulse with a width of 3/16th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances.

Table 785. HW_UARTAPP_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**SIREN**	RW	0x0	SIR Enable. If this bit is set to 1, the IrDA SIR ENDEC is enabled. (Unsupported in STMP3600.) This bit has no effect if the UART is not enabled by bit 0 being set to 1. When the IrDA SIR ENDEC is enabled, data is transmitted and received on nSIROUT and SIRIN. UARTTXD remains in the marking state (set to 1). Signal transitions on UARTRXD or modem status inputs have no effect. When the IrDA SIR ENDEC is disabled, nSIROUT remains cleared to 0 (no light pulses generated), and signal transitions on SIRIN have no effect.
0	**UARTEN**	RW	0x0	UART Enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for either UART signals or SIR signals according to the setting of SIR Enable (SIREN, bit 1). When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

DESCRIPTION:

Use this register to define the FIFO level at which the UARTTXINTR and UARTRX-INTR are triggered. The interrupts are generated based on a transition through a level rather than being based on the level. That is, the design is such that the interrupts are generated when the fill level progresses through the trigger level. The bits are reset so that the trigger level is when the FIFOs are at the half-way mark.

EXAMPLE:

```
Empty Example.
```

22.4.4. UART Line Control Register Description

The UART Line Control Register contains integer and fractional part of the baud rate divisor value. It also contains the line control bits.

HW_UARTAPP_LINECTRL 0x8006C030
HW_UARTAPP_LINECTRL_SET 0x8006C034
HW_UARTAPP_LINECTRL_CLR 0x8006C038
HW_UARTAPP_LINECTRL_TOG 0x8006C03C

Table 786. HW_UARTAPP_LINECTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BAUD_DIVINT																RSVD		BAUD_DIVFRAC						SPS	WLEN		FEN	STP2	EPS	PEN	BRK

Table 787. HW_UARTAPP_LINECTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**BAUD_DIVINT**	RW	0x0	Baud Rate Integer [15:0]. The integer baud rate divisor.
15:14	**RSVD**	RO	0x0	Reserved, do not modify, read as zero.
13:8	**BAUD_DIVFRAC**	RW	0x0	Baud Rate Fraction [5:0]. The fractional baud rate divisor.
7	**SPS**	RW	0x0	Stick Parity Select. When bits 1, 2, and 7 of this register are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set, and bit 2 is 0, the parity bit is transmitted and checked as a 1. When this bit is cleared stick parity is disabled.
6:5	**WLEN**	RW	0x0	Word length [1:0]. The select bits indicate the number of data bits transmitted or received in a frame as follows: 11 = 8 bits, 10 = 7 bits, 01 = 6 bits, 00 = 5 bits.
4	**FEN**	RW	0x0	Enable FIFOs. If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, the FIFOs are disabled (character mode); that is, the FIFOs become 1-byte-deep holding registers.
3	**STP2**	RW	0x0	Two Stop Bits Select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
2	**EPS**	RW	0x0	Even Parity Select. If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0, then odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by Parity Enable (PEN, bit 1) being cleared to 0.
1	**PEN**	RW	0x0	Parity Enable. If this bit is set to 1, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.
0	**BRK**	RW	0x0	Send Break. If this bit is set to 1, a low-level is continually output on the UARTTXD output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two complete frames. For normal use, this bit must be cleared to 0.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

22.4.5. UART Interrupt Register Description

The UART Interrupt Register contains the interrupt enables and the interrupt status. The interrupt status bits report the unmasked state of the interrupts. To clear a particular interrupt status bit, write the bit-clear address with the particular bit set to 1. The enable bits control the UART interrupt output: a 1 will enable a particular inter-

rupt to assert the UART interrupt output, while a 0 will disable the particular interrupt from affecting the interrupt output. All the bits, except for the modem status interrupt bits, are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

HW_UARTAPP_INTR 0x8006C040
HW_UARTAPP_INTR_SET 0x8006C044
HW_UARTAPP_INTR_CLR 0x8006C048
HW_UARTAPP_INTR_TOG 0x8006C04C

Table 788. HW_UARTAPP_INTR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1					OEIEN	BEIEN	PEIEN	FEIEN	RTIEN	TXIEN	RXIEN	DSRMIEN	DCDMIEN	CTSMIEN	RIMIEN	RSVD2					OEIS	BEIS	PEIS	FEIS	RTIS	TXIS	RXIS	DSRMIS	DCDMIS	CTSMIS	RIMIS

Table 789. HW_UARTAPP_INTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSVD1**	RO	0x0	Reserved, read as zero, do not modify.
26	**OEIEN**	RW	0x0	Overrun Error Interrupt Enable.
25	**BEIEN**	RW	0x0	Break Error Interrupt Enable.
24	**PEIEN**	RW	0x0	Parity Error Interrupt Enable.
23	**FEIEN**	RW	0x0	Framing Error Interrupt Enable.
22	**RTIEN**	RW	0x0	Receive Timeout Interrupt Enable.
21	**TXIEN**	RW	0x0	Transmit Interrupt Enable.
20	**RXIEN**	RW	0x0	Receive Interrupt Enable.
19	**DSRMIEN**	RW	0x0	nUARTDSR Modem Interrupt Enable. (Unsupported in STMP3600.)
18	**DCDMIEN**	RW	0x0	nUARTDCD Modem Interrupt Enable. (Unsupported in STMP3600.)
17	**CTSMIEN**	RW	0x0	nUARTCTS Modem Interrupt Enable.
16	**RIMIEN**	RW	0x0	nUARTRI Modem Interrupt Enable. (Unsupported in STMP3600.)
15:11	**RSVD2**	RO	0x0	Reserved, read as zero, do not modify.
10	**OEIS**	RW	0x0	Overrun Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
9	**BEIS**	RW	0x0	Break Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
8	**PEIS**	RW	0x0	Parity Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
7	**FEIS**	RW	0x0	Framing Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
6	**RTIS**	RW	0x0	Receive Timeout Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
5	**TXIS**	RW	0x0	Transmit Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.

Table 789. HW_UARTAPP_INTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	**RXIS**	RW	0x0	Receive Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
3	**DSRMIS**	RW	0x0	nUARTDSR Modem Interrupt Status. (Unsupported in STMP3600.) To clear this bit, write the bit-clear address with the particular bit set to 1.
2	**DCDMIS**	RW	0x0	nUARTDCD Modem Interrupt Status. (Unsupported in STMP3600.) To clear this bit, write the bit-clear address with the particular bit set to 1.
1	**CTSMIS**	RW	0x0	nUARTCTS Modem Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
0	**RIMIS**	RW	0x0	nUARTRI Modem Interrupt Status. (Unsupported in STMP3600.) To clear this bit, write the bit-clear address with the particular bit set to 1.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

22.4.6. UART Data Register Description

The UART Data Register is the receive and transmit data register. Receive (read) and transmit (write) up to four data characters per APB cycle.

HW_UARTAPP_DATA 0x8006C050

Table 790. HW_UARTAPP_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 791. HW_UARTAPP_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x0	In DMA mode, up to 4 Received/Transmit characters can be accessed at a time. In PIO mode, only one character can be accessed at a time. The status register contains the receive data flags and valid bits.

DESCRIPTION:

For words to be transmitted: 1) If the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO; 2) If the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). The write operation initiates transmission from the PrimeCell UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted. Note: With the use of APB byte-enables you can write 1, 2, or 4 valid bytes sumultaneously to the TXFIFO. The invalid bytes will also take up space in the TXFIFO. So every write cycle will consume 4 bytes in

the TXFIFO. If TXFIFO is disabled, you must only write the LSByte of the DATA register.

For received words: 1) If the FIFOs are enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO; 2) if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data bytes (up to 4) are read by performing reads from the 32-bit DATA register. The status information can be read by a read of the UART Status register.

The Overrun Error bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it. The Break Error bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received. When the Parity Error bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the LCR_H register. In FIFO mode, this error is associated with the character at the top of the FIFO. When the Framing Error bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.

EXAMPLE:

```
Empty Example.
```

22.4.7. UART Status Register Description

The UART Status Register contains the various flags and receive status. If the status is read from this register, then the status information for break, framing and parity corresponds to the data character read from the UART Data Register prior to reading the UART Status Register. The status information for overrun is set immediately when an overrun condition occurs.

HW_UARTAPP_STAT 0x8006C060

Table 792. HW_UARTAPP_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PRESENT	HISPEED	BUSY	CTS	TXFE	RXFF	TXFF	RXFE	RXBYTE_INVALID				OERR	BERR	PERR	FERR	RXCOUNT															

Table 793. HW_UARTAPP_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**PRESENT**	RO	0x1	This read-only bit indicates that the Application UART function is present when it reads back a one. This Application UART function is not available on a device that returns a zero for this bit field. UNAVAILABLE = 0x0 UARTAPP is not present in this product. AVAILABLE = 0x1 UARTAPP is present in this product.
30	**HISPEED**	RO	0x1	This read-only bit indicates that the high-speed function is present when it reads back a one. This high speed function is not available on a device that returns a zero for this bit field. UNAVAILABLE = 0x0 HISPEED is not present in this product. AVAILABLE = 0x1 HISPEED is present in this product.
29	**BUSY**	RO	0x0	UART Busy.
28	**CTS**	RO	0x0	Clear To Send.
27	**TXFE**	RO	0x1	Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART Line Control Register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.
26	**RXFF**	RO	0x0	Receive FIFO Full.
25	**TXFF**	RO	0x0	Transmit FIFO Full.
24	**RXFE**	RO	0x1	Receive FIFO Empty.
23:20	**RXBYTE_INVALID**	RW	0xf	The invalid state of the last read of Receive Data. Each bit corresponds to one byte of the RX data. (1 = invalid.)
19	**OERR**	RO	0x0	Overrun Error. This bit is set to 1 if data is received and the FIFO is already full. This bit is cleared to 0 by any write to the Status Register. The FIFO contents remain valid since no further data is written when the FIFO is full; only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
18	**BERR**	RW	0x0	Break Error. For PIO mode, this is for the last character read from the data register. For DMA mode, it will be set to 1 if any received character for a particular RXDMA command had a Break Error. To clear this bit, write a zero to it. Note that clearing this bit does not affect the interrupt status, which must be cleared by writing the interrupt register.
17	**PERR**	RW	0x0	Parity Error. For PIO mode, this is for the last character read from the data register. For DMA mode, it will be set to 1 if any received character for a particular RXDMA command had a Parity Error. To clear this bit, write a zero to it. Note that clearing this bit does not affect the interrupt status, which must be cleared by writing the interrupt register.

Table 793. HW_UARTAPP_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	**FERR**	RW	0x0	Framing Error. For PIO mode, this is for the last character read from the data register. For DMA mode, it will be set to 1 if any received character for a particular RXDMA command had a Framing Error. To clear this bit, write a zero to it. Note that clearing this bit does not affect the interrupt status, which must be cleared by writing the interrupt register.
15:0	**RXCOUNT**	RO	0x0	Number of bytes received during a Receive DMA command.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

22.4.8. UART Debug Register Description

The UART Debug Register contains the state of the DMA signals.

HW_UARTAPP_DEBUG 0x8006C070

Table 794. HW_UARTAPP_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																										TXDMARUN	RXDMARUN	TXCMDEND	RXCMDEND	TXDMARQ	RXDMARQ

Table 795. HW_UARTAPP_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:6	**RSVD1**	RO	0x0	Reserved, read as zero, do not modify.
5	**TXDMARUN**	RO	0x0	DMA Command Run Status: This bit reflects the state of the toggle signal for TXDMARUN.
4	**RXDMARUN**	RO	0x0	DMA Command Run Status: This bit reflects the state of the toggle signal for RXDMARUN.
3	**TXCMDEND**	RO	0x0	DMA Command End Status: This bit reflects the state of the toggle signal for UART_TXCMDEND.
2	**RXCMDEND**	RO	0x0	DMA Command End Status: This bit reflects the state of the toggle signal for UART_RXCMDEND.
1	**TXDMARQ**	RO	0x0	DMA Request Status: This bit reflects the state of the toggle signal for UART_TXDMAREQ. Note that TX burst request is not supported.
0	**RXDMARQ**	RO	0x0	DMA Request Status: This bit reflects the state of the toggle signal for UART_RXDMAREQ. Note that RX burst request is not supported.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

UARTAPP XML Revision: 1.42

23. DEBUG UART

This chapter describes the Debug UART included on the STMP36xx, how to operate it, and how to disable the FIFOs. Programmable registers are described in Section 23.4.

23.1. Overview

The Debug UART performs:

- Serial-to-parallel conversion on data received from a peripheral device
- Parallel-to-serial conversion on data transmitted to the peripheral device

The CPU reads and writes data and control/status information through the APBX interface. The transmit and receive paths are buffered with internal FIFO memories, enabling up to 16 bytes to be stored independently in both transmit and receive modes.

The Debug UART includes a programmable baud rate generator that creates a common transmit and receive internal clock from the 24-MHz UART internal reference clock input UARTCLK, which is tied internally to XCLK.

It offers similar functionality to the industry-standard 16C550 UART device and supports baud rates of up to 1Mbits/s (in high-speed configuration). Figure 108 shows a block diagram of the Debug UART. The Debug UART operation and baud rate values are controlled by the line control register (HW_UARTAPP_LINECTRL).

The Debug UART can generate a single combined interrupt, so output is asserted if any individual interrupt is asserted and unmasked. Interrupt sources include the receive (including timeout), transmit, modem status, and error conditions.

If a framing, parity, or break error occurs during reception, the appropriate error bit is set, and is stored in the FIFO. If an overrun condition occurs, the overrun register bit is set immediately, and FIFO data is prevented from being overwritten. You can program the FIFOs to be one-byte deep, providing a conventional double-buffered UART interface.

Unlike the Application UART, the Debug UART does not support DMA or flow control (CTS/RTS).

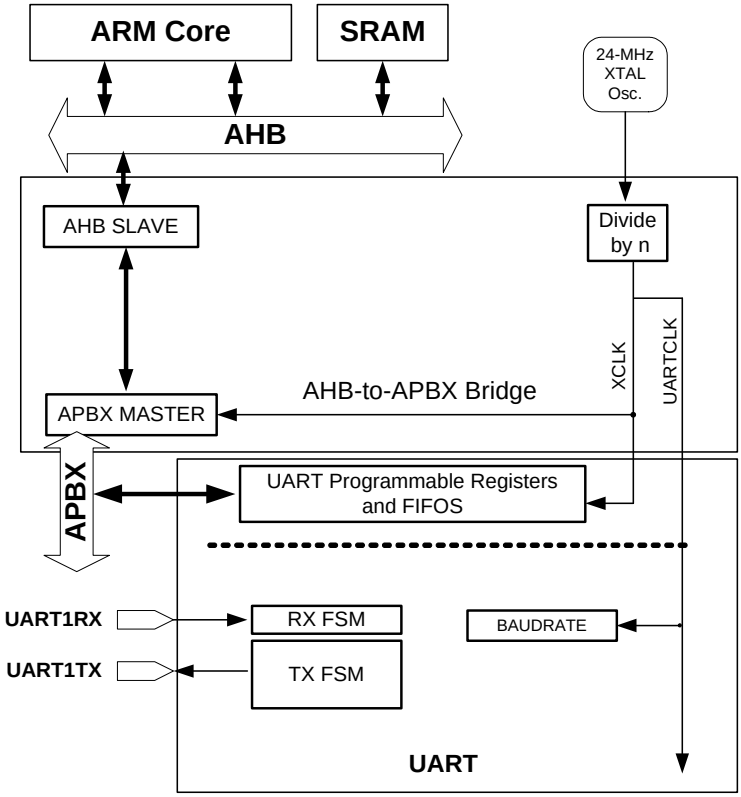


Figure 107. Debug UART Block Diagram

23.2. Operation

Control data is written to the Debug UART line control register. This register defines:

- Transmission parameters
- Word length
- Buffer mode
- Number of transmitted stop bits
- Parity mode
- Break generation
- Baud rate divisor

23.2.1. Fractional Baud Rate Divider

The baud rate divisor is calculated from the frequency of XCLK and the desired baud rate by using the following formula:

```
divisor = (XCLK * 4) / baud rate, rounded to the nearest integer
```

The divisor must be between 0x00000040 and 0x003FFFC0, inclusive. Program the lowest 6 bits of the divisor into BAUD_DIVFRAC, and the next 16 bits of the divisor into BAUD_DIVINT.

In the debug UART, HW_UARTDBGLCR_H, HW_UARTDBGIBRD, and HW_UARTDBGFBRD form a single 30-bit wide register (UARTLCR) that is updated on a single write strobe generated by an HW_UARTDBGLCR_H write. So, in order

to internally update the contents of HW_UARTDBGIBRD or HW_UARTDBGFBRD, a write to HW_UARTDBGLCR_H must always be performed at the end.

23.2.2. UART Character Frame

Figure 108 illustrates the UART character frame.

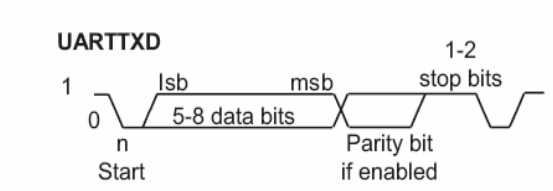


Figure 108. Debug UART Character Frame

23.2.3. Data Transmission or Reception

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information.

For transmission, data is written into the transmit FIFO. If the Debug UART is enabled, it causes a data frame to start transmitting with the parameters indicated in UARTLCR_H. Data continues to be transmitted until there is no data left in the transmit FIFO.

The BUSY signal goes HIGH as soon as data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted HIGH while data is being transmitted. BUSY is negated only when the transmit FIFO is empty and the last character has been transmitted from the shift register, including the stop bits. BUSY can be asserted HIGH even though the Debug UART might no longer be enabled.

For each sample of data, three readings are taken and the majority value is kept. In the following paragraphs, the middle sampling point is defined and one sample is taken either side of it.

- When the receiver is idle (UARTRXD continuously 1, in the marking state) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by Baud16, begins running and data is sampled on the eighth cycle of that counter in normal UART mode to allow for the shorter logic 0 pulses (half way through a bit period).
- The start bit is valid if UARTRXD is still LOW on the eighth cycle of Baud16, otherwise a false start bit is detected and it is ignored. If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked, if parity mode was enabled.
- Lastly, a valid stop bit is confirmed if UARTRXD is HIGH, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word (see Table 796).

23.2.4. Error Bits

Three error bits are stored in bits [10:8] of the receive FIFO and are associated with a particular character. An additional error indicating an overrun error is stored in bit 11 of the receive FIFO.

23.2.5. Overrun Bit

The overrun bit is not associated with the character in the receive FIFO. The overrun error is set when the FIFO is full, and the next character is completely received in the shift register. The data in the shift register is overwritten, but it is not written into the FIFO. When an empty location is available in the receive FIFO, and another character is received, the state of the overrun bit is copied into the receive FIFO along with the received character. The overrun state is then cleared. Table 796 shows the bit functions of the receive FIFO.

Table 796. Receive FIFO Bit Functions

FIFO BIT	FUNCTION
11	Overrun indicator
10	Break error
9	Parity error
8	Framing error
7:0	Received data

23.3. Disabling the FIFOs

FIFOs can be disabled. In this case, the transmit and receive sides of the PrimeCell UART have one-byte holding registers (the bottom entry of the FIFOs). The overrun bit is set when a word has been received and the previous one was not yet read.

In this implementation, the FIFOs are not physically disabled, but the flags are manipulated to give the illusion of a one-byte register.

23.4. Programmable Registers

This section describes the Debug UART's programable registers.

23.4.1. UART Data Register Description

For words to be transmitted: 1) If the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO 2) If the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). The write operation initiates transmission from the PrimeCell UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted. For received words: 1) If the FIFOs are enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO 2) If the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data byte is read by performing reads from the DR register along with the corresponding status information. The status information can also be read by a read of the HW_UARTDBGRSR_ECR register.

HW_UARTDBGDR 0x80070000

Table 797. HW_UARTDBGDR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED				OE	BE	PE	FE	DATA							

Table 798. HW_UARTDBGDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:12	**RESERVED**	RO	0x0	Reserved.
11	**OE**	RO	0x0	Overrun Error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.
10	**BE**	RO	0x0	Break Error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held low for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received.
9	**PE**	RO	0x0	Parity Error. When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the LCR_H register. In FIFO mode, this error is associated with the character at the top of the FIFO.
8	**FE**	RO	0x0	Framing Error. When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.
7:0	**DATA**	RW	0x0	Receive (read) data character. Transmit (write) data character.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.2. UART Receive Status Register (Read) and Error Clear Register (Write) Description

The RSR_ECR register is the receive status and error clear register. If the status is read from this register, then the status information for break, framing, and parity corresponds to the data character read from DR prior to reading RSR_ECR. The status information for overrun is set immediately when an overrun condition occurs. A write to RSR_ECR clears the framing, parity, break, and overrun errors.

HW_UARTDBGRSR_ECR 0x80070004

Table 799. HW_UARTDBGRSR_ECR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																								EC				OE	BE	PE	FE

Table 800. HW_UARTDBGRSR_ECR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:8	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
7:4	**EC**	RW	0x0	Error Clear. Any write to this bitfield clears the framing, parity, break, and overrun errors. The value is unpredictable when read.
3	**OE**	RW	0x0	Overrun Error. This bit is set to 1 if data is received and the FIFO is already full. This bit is cleared to 0 by any write to RSR_ECR. The FIFO contents remain valid, since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
2	**BE**	RW	0x0	Break Error.
1	**PE**	RW	0x0	Parity Error.
0	**FE**	RW	0x0	Framing Error.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.3. UART Flag Register Description

The FR register is the flag register.

HW_UARTDBGFR 0x80070018

Table 801. HW_UARTDBGFR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED							RI	TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS

Table 802. HW_UARTDBGFR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:9	**RESERVED**	RO	0x0	Reserved, do not modify, read as zero.
8	**RI**	RO	0x0	Ring Indicator. This bit is the complement of the UART ring indicator (nUARTRI) modem status input. That is, the bit is 1 when the modem status input is 0.
7	**TXFE**	RO	0x1	Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the LCR_H register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.
6	**RXFF**	RO	0x0	Receive FIFO Full.
5	**TXFF**	RO	0x0	Transmit FIFO Full.
4	**RXFE**	RO	0x1	Receive FIFO Empty.
3	**BUSY**	RO	0x0	UART Busy.
2	**DCD**	RO	0x0	Data Carrier Detect.
1	**DSR**	RO	0x0	Data Set Ready.
0	**CTS**	RO	0x0	Clear To Send.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.4. UART IrDA Low-Power Counter Register Description

The UART IrDA Low-Power Counter Register is an 8-bit read/write register that stores a low-power counter divisor value used to divide down the UARTCLK to generate the IrLPBaud16 signal.

HW_UARTDBGILPR 0x80070020

Table 803. HW_UARTDBGILPR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																								ILPDVSR							

Table 804. HW_UARTDBGILPR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:8	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
7:0	**ILPDVSR**	RW	0x0	IrDA Low Power Divisor [7:0]. 8-bit low-power divisor value.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.5. UART Integer Baud Rate Divisor Register Description

The IBRD register is the integer part of the baud rate divisor value.

HW_UARTDBGIBRD 0x80070024

Table 805. HW_UARTDBGIBRD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																BAUD_DIVINT															

Table 806. HW_UARTDBGIBRD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:0	**BAUD_DIVINT**	RW	0x0	Baud Rate Integer [15:0]. The integer baud rate divisor.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.6. UART Fractional Baud Rate Divisor Register Description

The FBRD register is the fractional part of the baud rate divisor value.

HW_UARTDBGFBRD 0x80070028

Table 807. HW_UARTDBGFBRD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																								RESERVED		BAUD_DIVFRAC					

Table 808. HW_UARTDBGFBRD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:8	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
7:6	**RESERVED**	RO	0x0	Not documented.
5:0	**BAUD_DIVFRAC**	RW	0x0	Baud Rate Fraction [5:0]. The fractional baud rate divisor.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.7. UART Line Control Register, High Byte Description

The LCR_H is the Line Control Register.

HW_UARTDBGLCR_H 0x8007002C

Table 809. HW_UARTDBGLCR_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED								SPS	WLEN		FEN	STP2	EPS	PEN	BRK

Table 810. HW_UARTDBGLCR_H Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:8	**RESERVED**	RO	0x0	Reserved, do not modify, read as zero.
7	**SPS**	RW	0x0	Stick Parity Select. When bits 1, 2, and 7 of this register are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set, and bit 2 is 0, the parity bit is transmitted and checked as a 1. When this bit is cleared, stick parity is disabled.
6:5	**WLEN**	RW	0x0	Word Length [1:0]. The select bits indicate the number of data bits transmitted or received in a frame as follows: 11 = 8 bits, 10 = 7 bits, 01 = 6 bits, 00 = 5 bits.
4	**FEN**	RW	0x0	Enable FIFOs. If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, the FIFOs are disabled (character mode); that is, the FIFOs become 1-byte-deep holding registers.
3	**STP2**	RW	0x0	Two Stop Bits Select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
2	**EPS**	RW	0x0	Even Parity Select. If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0, then odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by Parity Enable (PEN, bit 1) being cleared to 0.

Table 810. HW_UARTDBGLCR_H Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**PEN**	RW	0x0	Parity Enable. If this bit is set to 1, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.
0	**BRK**	RW	0x0	Send Break. If this bit is set to 1, a low-level is continually output on the UARTTXD output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two complete frames. For normal use, this bit must be cleared to 0.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.8. UART Control Register Description

The CR is the Control Register.

HW_UARTDBGCR 0x80070030

Table 811. HW_UARTDBGCR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																CTSEN	RTSEN	OUT2	OUT1	RTS	DTR	RXE	TXE	LBE	RESERVED				SIRLP	SIREN	UARTEN

Table 812. HW_UARTDBGCR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15	**CTSEN**	RW	0x0	CTS Hardware Flow Control Enable. Not implemented.
14	**RTSEN**	RW	0x0	RTS Hardware Flow Control Enable. Not implemented.
13	**OUT2**	RW	0x0	This bit is the complement of the UART Out2 (nUARTOut2) modem status output. Not implemented.
12	**OUT1**	RW	0x0	This bit is the complement of the UART Out1 (nUARTOut1) modem status output. Not implemented.
11	**RTS**	RW	0x0	Request To Send. Not implemented.
10	**DTR**	RW	0x0	Data Transmit Ready. Not implemented.

Table 812. HW_UARTDBGCR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9	**RXE**	RW	0x1	Receive Enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for either UART signals or SIR signals according to the setting of SIR Enable (SIREN, bit 1). When the UART is disabled in the middle of reception, it completes the current character before stopping.
8	**TXE**	RW	0x1	Transmit Enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for either UART signals or SIR signals according to the setting of SIR Enable (SIREN, bit 1). When the UART is disabled in the middle of transmission, it completes the current character before stopping.
7	**LBE**	RW	0x0	Loop Back Enable. If this bit is set to 1 and the SIR Enable bit is set to 1 and the test register TCR bit 2 (SIRTEST) is set to 1, then the nSIROUT path is inverted and fed through to the SIRIN path. The SIRTEST bit in the test register must be set to 1 to override the normal half-duplex SIR operation. This must be the requirement for accessing the test registers during normal operation, and SIRTEST must be cleared to 0 when loopback testing is finished. This feature reduces the amount of external coupling required during system test. If this bit is set to 1 and the SIRTEST bit is set to 0, the UARTTXD path is fed through to the UARTRXD path. In either SIR mode or normal mode, when this bit is set, the modem outputs are also fed through to the modem inputs.
6:3	**RESERVED**	RO	0x0	Reserved, do not modify, read as zero.
2	**SIRLP**	RW	0x0	IrDA SIR Low-Power Mode. Not supported.
1	**SIREN**	RW	0x0	SIR Enable. If this bit is set to 1, the IrDA SIR ENDEC is enabled. Not supported.
0	**UARTEN**	RW	0x0	UART Enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for either UART signals or SIR signals, according to the setting of SIR Enable (SIREN, bit 1). When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.9. UART Interrupt FIFO Level Select Register Description

The IFLS register is the Interrupt FIFO Level Select Register. Use the IFLS register to define the FIFO level at which the UARTTXINTR and UARTRXINTR are triggered. The interrupts are generated based on a transition through a level rather

than being based on the level. That is, the design is such that the interrupts are generated when the fill level progresses through the trigger level. The bits are reset so that the trigger level is when the FIFOs are at the half-way mark.

HW_UARTDBGIFLS 0x80070034

Table 813. HW_UARTDBGIFLS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED										RXIFLSEL			TXIFLSEL		

Table 814. HW_UARTDBGIFLS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:6	**RESERVED**	RO	0x0	Reserved, do not modify, read as zero.
5:3	**RXIFLSEL**	RW	0x2	Receive Interrupt FIFO Level Select. The trigger points for the receive interrupt are as follows: NOT_EMPTY = 0x0 Trigger on FIFO not empty, i.e., at least 1 of 8 entries. ONE_QUARTER = 0x1 Trigger on FIFO full to at least 2 of 8 entries. ONE_HALF = 0x2 Trigger on FIFO full to at least 4 of 8 entries. THREE_QUARTERS = 0x3 Trigger on FIFO full to at least 6 of 8 entries. SEVEN_EIGHTHS = 0x4 Trigger on FIFO full to at least 7 of 8 entries. INVALID5 = 0x5 Reserved. INVALID6 = 0x6 Reserved. INVALID7 = 0x7 Reserved.
2:0	**TXIFLSEL**	RW	0x2	Transmit Interrupt FIFO Level Select. The trigger points for the transmit interrupt are as follows: EMPTY = 0x0 Trigger on FIFO empty, i.e., no entries. ONE_QUARTER = 0x1 Trigger on FIFO less than 2 of 8 entries. ONE_HALF = 0x2 Trigger on FIFO less than 4 of 8 entries. THREE_QUARTERS = 0x3 Trigger on FIFO less than 6 of 8 entries. SEVEN_EIGHTHS = 0x4 Trigger on FIFO less than 7 of 8 entries. INVALID5 = 0x5 Reserved. INVALID6 = 0x6 Reserved. INVALID7 = 0x7 Reserved.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.10. UART Interrupt Mask Set/Clear Register Description

The IMSC register is the Interrupt Mask Set/Clear Register. On a read, this register gives the current value of the mask on the relevant interrupt. On a write of 1 to the

particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

HW_UARTDBGIMSC 0x80070038

Table 815. HW_UARTDBGIMSC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRMIM	DCDMIM	CTSMIM	RIMIM

Table 816. HW_UARTDBGIMSC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:11	**RESERVED**	RO	0x0	Reserved, do not modify, read as zero.
10	**OEIM**	RW	0x0	Overrun Error Interrupt Mask. On a read, the current mask for the OEIM interrupt is returned. On a write of 1, the mask of the OEIM interrupt is set. A write of 0 clears the mask.
9	**BEIM**	RW	0x0	Break Error Interrupt Mask.
8	**PEIM**	RW	0x0	Parity Error Interrupt Mask.
7	**FEIM**	RW	0x0	Framing Error Interrupt Mask.
6	**RTIM**	RW	0x0	Receive Timeout Interrupt Mask.
5	**TXIM**	RW	0x0	Transmit Interrupt Mask.
4	**RXIM**	RW	0x0	Receive Interrupt Mask.
3	**DSRMIM**	RW	0x0	nUARTDSR Modem Interrupt Mask.
2	**DCDMIM**	RW	0x0	nUARTDCD Modem Interrupt Mask.
1	**CTSMIM**	RW	0x0	nUARTCTS Modem Interrupt Mask.
0	**RIMIM**	RW	0x0	nUARTRI Modem Interrupt Mask.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.11. UART Raw Interrupt Status Register Description

The UART Raw Interrupt Status Register is a read-only register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect. All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

HW_UARTDBGRIS 0x8007003C

Table 817. HW_UARTDBGRIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	DSRRMIS	DCDRMIS	CTSRMIS	RIRMIS

Table 818. HW_UARTDBGRIS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:11	**RESERVED**	RO	0x0	Reserved, read as zero, do not modify.
10	**OERIS**	RO	0x0	Overrun Error Interrupt Status.
9	**BERIS**	RO	0x0	Break Error Interrupt Status.
8	**PERIS**	RO	0x0	Parity Error Interrupt Status.
7	**FERIS**	RO	0x0	Framing Error Interrupt Status.
6	**RTRIS**	RO	0x0	Receive Timeout Interrupt Status.
5	**TXRIS**	RO	0x0	Transmit Interrupt Status.
4	**RXRIS**	RO	0x0	Receive Interrupt Status.
3	**DSRRMIS**	RO	0x0	nUARTDSR Modem Interrupt Status.
2	**DCDRMIS**	RO	0x0	nUARTDCD Modem Interrupt Status.
1	**CTSRMIS**	RO	0x0	nUARTCTS Modem Interrupt Status.
0	**RIRMIS**	RO	0x0	nUARTRI Modem Interrupt Status.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.12. UART Masked Interrupt Status Register Description

The UART Masked Interrupt Status Register 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. All the bits except for the modem status interrupt bits (bits 3 to 0) are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

HW_UARTDBGMIS 0x80070040

Table 819. HW_UARTDBGMIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	DSRMMIS	DCDMMIS	CTSMMIS	RIMMIS

Table 820. HW_UARTDBGMIS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:11	**RESERVED**	RO	0x0	Reserved, read as zero, do not modify.
10	**OEMIS**	RO	0x0	Overrun Error Masked Interrupt Status.
9	**BEMIS**	RO	0x0	Break Error Masked Interrupt Status.
8	**PEMIS**	RO	0x0	Parity Error Masked Interrupt Status.
7	**FEMIS**	RO	0x0	Framing Error Masked Interrupt Status.
6	**RTMIS**	RO	0x0	Receive Timeout Masked Interrupt Status.
5	**TXMIS**	RO	0x0	Transmit Masked Interrupt Status.
4	**RXMIS**	RO	0x0	Receive Masked Interrupt Status.
3	**DSRMMIS**	RO	0x0	nUARTDSR Modem Masked Interrupt Status.
2	**DCDMMIS**	RO	0x0	nUARTDCD Modem Masked Interrupt Status.
1	**CTSMMIS**	RO	0x0	nUARTCTS Modem Masked Interrupt Status.
0	**RIMMIS**	RO	0x0	nUARTRI Modem Masked Interrupt Status.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.13. UART Interrupt Clear Register Description

The UART Interrupt Clear Register is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

HW_UARTDBGICR 0x80070044

Table 821. HW_UARTDBGICR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC

Table 822. HW_UARTDBGICR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:11	**RESERVED**	RO	0x0	Reserved, read as zero, do not modify.
10	**OEIC**	WO	0x0	Overrun Error Interrupt Clear.
9	**BEIC**	WO	0x0	Break Error Interrupt Clear.
8	**PEIC**	WO	0x0	Parity Error Interrupt Clear.
7	**FEIC**	WO	0x0	Framing Error Interrupt Clear.
6	**RTIC**	WO	0x0	Receive Timeout Interrupt Clear.
5	**TXIC**	WO	0x0	Transmit Interrupt Clear.
4	**RXIC**	WO	0x0	Receive Interrupt Clear.
3	**DSRMIC**	WO	0x0	nUARTDSR Modem Interrupt Clear.
2	**DCDMIC**	WO	0x0	nUARTDCD Modem Interrupt Clear.
1	**CTSMIC**	WO	0x0	nUARTCTS Modem Interrupt Clear.
0	**RIMIC**	WO	0x0	nUARTRI Modem Interrupt Clear.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

23.4.14. UART DMA Control Register Description

The UART DMA Control Register is a read/write register. All the bits are cleared to 0 on reset.

HW_UARTDBGDMACR 0x80070048

Table 823. HW_UARTDBGDMACR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNAVAILABLE																RESERVED													DMAONERR	TXDMAE	RXDMAE

Table 824. HW_UARTDBGDMACR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNAVAILABLE**	RO	0x0	The UART IP only implements 16- and 8-bit registers, so the top two or three bytes of every 32-bit register are always unavailable.
15:3	**RESERVED**	RO	0x0	Reserved, read as zero, do not modify.
2	**DMAONERR**	RW	0x0	DMA On Error.
1	**TXDMAE**	RW	0x0	Transmit DMA Enable. Not implemented.
0	**RXDMAE**	RW	0x0	Receive DMA Enable. Not implemented.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

UARTDBG XML Revision: 1.22

24. IRDA CONTROLLER

This chapter describes the IrDA controller included on the STMP36xx. Programmable registers are described in Section 24.4.

24.1. Overview

The STMP36xx provides a complete solution for implementing IR communications in embedded applications, including the following features:

- IrDA protocol controller
- High-performance IrDA protocol controller
  - Compatible with IrDA Serial Infrared Physical Layer Specification (IrPHY), Version 1.4
  - Supports VFIR, FIR, MIR, and SIR rates.
- Two independent DMA channels

A block diagram of the IrDA controller is shown in Figure 109.

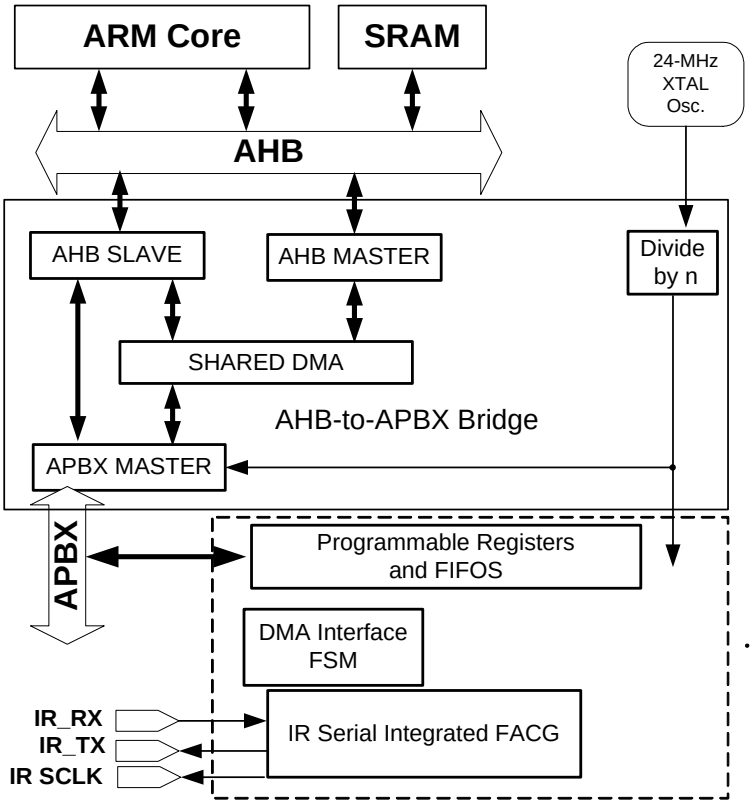


Figure 109. IrDA Controller Block Diagram

The STMP36xx uses a low-cost, low-power, IrDA Very Fast Infrared (VFIR) controller integrated circuit for enabling IrDA communication capabilities in embedded designs. An IR communication subsystem can be implemented simply by adding an IrDA compatible transceiver.

The IrDA controller on the STMP36xx is compatible with the IrDA Serial Infrared Physical Layer Specification (IrPHY) Version 1.4. It supports a host interface, an IR protocol controller, and memory for buffering. The IrDA controller on the STMP36xx supports data rates up to 16 Mbits per second (VFIR), including all slower data rates (FIR, MIR, and SIR).

The IR protocol controller supports the various IrDA encode and decode operations required by the various data rates. This allows the STMP36xx to interface to a wide range of IR capable devices at the highest performance.

The IrDA-compatible controller is programmed and controlled by software operating on the embedded host processor. The IrDA controller is software-configurable via the APBX bus through a set of accessible registers. Details about the register set and a simple SPI protocol are defined in Section 24.4. Typically, an IrDA software stack, application software, and STMP36xx driver provides IrDA-compatible operation.

The STMP36xx is ideal for mobile applications where size and power consumption are important. When combined with a VFIR transceiver, highly portable devices such as digital media players, mobile phone handsets, and digital still cameras can provide a private, high-performance, wireless data transfer capability.

24.2. Operation

The IR controller is designed to meet the IrDA Physical Layer Specification, Version 1.4. For more information and detailed specifications, refer to http://www.irda.org. All data rates that are called for in the specification are supported, including:

- SIR: 2400 bps, 9600 bps, 19.2 kbps, 38.4 kbps, 57.6 kbps and 115.2 kbps
- MIR: 576 kbps and 1.152 Mbps
- FIR: 4 Mbps
- VFIR: 16 Mbps

24.2.1. DMA Operation

The IR controller resides on the APBX bus as a DMA slave. There are two independent DMA channels for IR, one for transmission and one for receiving. For APBX DMA Channel 6, which is the IrDA RX channel, the first PIO word in the DMA command is CTRL1. However, for APBX DMA Channel 7, which is the IrDA TX, the first PIO word in a DMA command is CTRL1. For a detailed description of how the DMA works, refer to Chapter 11.

The IR controller encodes data from the APBX TX DMA channel and sends it out over IR_TX to the transceiver. The incoming IR_RX is sampled and decoded, then transferred via the ABPX DMA RX channel. Data is processed one byte (8 bits) at a time. The module also supports the serial interface using IR_SCLK and communicates with the transceiver device.

24.2.2. IR Transmit Processing

The maximum size allowed is 2050 bytes per IR specifications. The size field on transmits can be 0; this is meaningful only if a speed change is requested.

The IR block receives packets for transmission through an asynchronous FIFO. When the TX block first detects the FIFO is non-empty, it checks to see if RX or the serial interface is currently active. If not, transmission begins immediately. Otherwise, it waits until there is no other activity before starting transmission.

The IR TX block stalls the DMA request if it is disabled or if its FIFO is full. If the TX block is processing data faster than the APBX DMA is sending them, then an underflow condition in the TX FIFO can occur. This is a catastrophic failure for the current frame because gaps in IR are not allowed. In such cases, the underflow flag pulses, and the error interrupt is generated, if it is enabled. A TX_IRQ is asserted for every completion of a frame, if enabled.

24.2.3. IR Receive Processing

IR block starts a DMA request once data is available in the RX FIFO. The length of the received frame is unknown until the end-of-frame stop flag is detected, when the length of a RX DMA transaction has to be determined. The RX DMA interface issues a 32-bit status word for each DMA block (the total DMA length is the programmed length + 4 bytes). If the status word is 0xFFFFFFFF, the current frame is not finished yet, and more data is available for retrieval. If the RX DMA block size is larger than the remaining IR RX data, zeros are stuffed for the required DMA transfer size, and the status word indicates the end of the frame. Table 825 describes the RX status bit field definitions.

Table 825. RX Status Bit Field Definitions

BITS	LABEL	DEFINITION
23	RXTOOBIG	The frame was discarded because it was too big for the available buffer.
22	RXBOFINFRAME	A 0xC0 (BOF) was detected in the body of an incoming IR frame. The frame reception was consequently restarted after the BOF and the previous data was discarded
21	RSVD1	
20	RXMISSEDEOF	No 0xC1 (EOF) was detected in the incoming IR frame.
19	RXMISSEDBOF	No 0xC0 (BOF) was detected in the incoming frame. Can only occur in SIR when framing is active and RXBOFOVER is set.
18	RXFRAMEABORT	The frame was aborted due to error or transmitter abort.
17	RXCRCERROR	The frame had an invalid CRC.
16	RXSUCCESS	1 if RX packet was successful. (No error bits set).
15:0	RXSIZE	Length of the RX frame in byte.

The IR RX block stalls the DMA request if it is disabled or if its FIFO is empty. If the RX block is receiving data faster than the APBX DMA reading from it, then an overflow condition in the RX FIFO can occur. In such cases, the overflow flag pulses and the error interrupt is generated if it is enabled. An RX_IRQ is generated for every completion of a frame, if enabled.

24.2.4. IR Serial Interface

IR can send Serial Command to the IR Transceiver through the IR Serial Command Interface. At power up, it is required that the IR transceiver be reset before accessing it. This can be done by writing a 1 to the INIT bit in the HW_IR_TCCTRL register. The host should then wait until the BUSY bit is low in the same register before attempting to access the IR transceiver again.

Once reset, commands can be sent to the IR transceivers to put the transceiver into the desired operation speed and power mode. As an example, a 1-byte register

write command looks like Figure 110 (from the Vishay Semiconductors VFIR transceiver TFDU8108 specification).

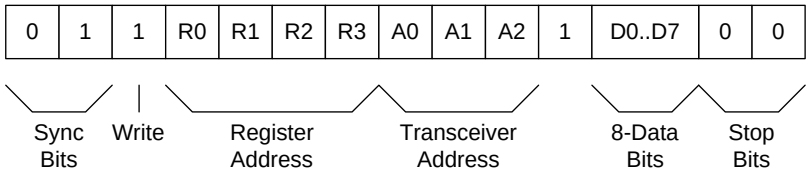


Figure 110. Example of 1-Byte Serial Interface Write Command

The bits indicated are driven out on IR_TX, and data is clocked using IR_SCLK. After the DATA, ADDR, INDX and C fields in the HW_IR_TCCTRL register are programmed, the IR issues the read/write command to transceiver through the serial interface.

24.2.5. IR Clock Configuration

There are two clocks in the IR controller. One is the IR_CLK, which varies according the operation mode and speed. The other clock is the oversampling clock IROV_CLK, which runs at a multiple of the IR_CLK frequency. The appropriate clock frequency (as listed in Table 826) could be generated by programming the HW_CLKCTRL_IRCLKCTRL register manually. However, the preferred way to get the desired IR clock frequency is to assert the AUTO_DIV bit in the HW_CLKCTRL_IRCLKCTRL register. The clock controller then automatically programs the IR_DIV and IROV_DIV based on the speed and mode information from IR.

Table 826. IR Clock Divider

MODE/SPEED	IR_DIV	IROV_DIV
SIR 2400 bps	768	260
SIR 9600 bps	192	260
SIR 19.2 kps	96	260
SIR 38.4 kps	48	260
SIR 57.6 kps	32	260
SIR 115.2 kbps	16	260
MIR 0.576 Mbps	16	52
MIR 1.152 Mbps	16	26
FIR 4.0 Mbps	5	12
VFIR 16.0 Mbps	5	4

24.3. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

24.4. Programmable Registers

This section describes the IrDA controller's programable registers.

24.4.1. IR Control Register Description

The IR Control Register contains global IR configuration and enables.

HW_IR_CTRL	0x80078000
HW_IR_CTRL_SET	0x80078004
HW_IR_CTRL_CLR	0x80078008
HW_IR_CTRL_TOG	0x8007800C

Table 827. HW_IR_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSVD2			MTA			MODE		SPEED			RSVD1					TC_TIME_DIV						TC_TYPE	SIR_GAP			SIPEN	TCEN	TXEN	RXEN

Table 828. HW_IR_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. Also used as a clock gate. RUN = 0x0 Allow IR to operate normally. RESET = 0x1 Hold IR in reset.
30	**CLKGATE**	RW	0x1	Set this bit zero for normal operation. Setting this bit to one (default) gates all of the block level clocks off to miniminize AC energy consumption.
29:27	**RSVD2**	RO	0x0	Reserved
26:24	**MTA**	RW	0x0	Minimum Turnaround Time. The least delay from the last receive to when a transmission can begin. MTA_10MS = 0x0 Turnaround Time > 10 ms MTA_5MS = 0x1 Turnaround Time > 5 ms MTA_1MS = 0x2 Turnaround Time > 1 ms MTA_500US = 0x3 Turnaround Time > 0.5 ms MTA_100US = 0x4 Turnaround Time > 0.1 ms MTA_50US = 0x5 Turnaround Time > 0.05 ms MTA_10US = 0x6 Turnaround Time > 0.01 ms MTA_0 = 0x7 Disabled
23:22	**MODE**	RW	0x0	IRDA Mode. SIR = 0x0 MIR = 0x1 FIR = 0x2 VFIR = 0x3
21:19	**SPEED**	RW	0x0	Speed Select. Select Speed in SIR/MIR mode. SPD000 = 0x0 SIR 9600bps MIR 1.152 Mbps SPD001 = 0x1 SIR 19.2Kbps MIR 0.576 Mbps SPD010 = 0x2 SIR 38.4 Kbps SPD011 = 0x3 SIR 57.6 Kbps SPD100 = 0x4 SIR 115.2 Kbps SPD101 = 0x5 SIR 2400 bps
18:14	**RSVD1**	RO	0x0	Reserved

Table 828. HW_IR_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
13:8	**TC_TIME_DIV**	RW	0x0	Divider from APBX Clock Period for Transceiver Control Timing
7	**TC_TYPE**	RW	0x0	Transceiver Control Type. 0=Serial Interface 1=Temic
6:4	**SIR_GAP**	RW	0x0	Gap between bytes in SIR. Number of bit-times to wait before deciding transmission has been blocked. GAP_10K = 0x0 10,000 IR_CLK cycles GAP_5K = 0x1 5,000 IR_CLK cycles GAP_1K = 0x2 1,000 IR_CLK cycles GAP_500 = 0x3 500 IR_CLK cycles GAP_100 = 0x4 100 IR_CLK cycles GAP_50 = 0x5 50 IR_CLK cycles GAP_10 = 0x6 10 IR_CLK cycles GAP_0 = 0x7 Disabled
3	**SIPEN**	RW	0x0	SIP Enable. Set to 1 to enable generation of SIP in modes faster than SIR.
2	**TCEN**	RW	0x0	Transceiver Enable. Set to 1 to enable the serial interface or Temic pulse generator.
1	**TXEN**	RW	0x0	Transmit Enable. Set to 1 to enable the transmit generator.
0	**RXEN**	RW	0x0	Receive Enable. Set to 1 to enable the receive parser.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example
```

24.4.2. IR Transmit DMA Control Register Description

The IR Transmit DMA Control Register configures the transmit DMA.

HW_IR_TXDMA 0x80078010
HW_IR_TXDMA_SET 0x80078014
HW_IR_TXDMA_CLR 0x80078018
HW_IR_TXDMA_TOG 0x8007801C

Table 829. HW_IR_TXDMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RUN	RSVD2	EMPTY	INT	CHANGE	NEW_MTA			NEW_MODE		NEW_SPEED			BOF_TYPE	XBOFS						XFER_COUNT											

Table 830. HW_IR_TXDMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RUN**	RW	0x0	Tells the IR to execute the Transmit DMA command. The IR will clear this bit at the end of transmit execution.
30	**RSVD2**	RO	0x0	Reserved
29	**EMPTY**	RW	0x0	Indicates there is no data associated with this descriptor. This is a speed-change only transaction. If this bit is set, XFER_COUNT must be set to 0.
28	**INT**	RW	0x0	If set, will generate a speed-change interrupt at end of frame. Note this interrupt will occur regardless of whether CHANGE is set. If software wants to change speeds at end of the frame, CHANGE must be set.
27	**CHANGE**	RW	0x0	If set, an update to MODE, SPEED, and MTA register fields will occur at end of frame.
26:24	**NEW_MTA**	RW	0x0	New MTA setting to take effect at the end of this frame. See MTA field in CTRL register for encoding.
23:22	**NEW_MODE**	RW	0x0	New Mode to change to at end of this frame. See MODE field in CTRL register for encoding.
21:19	**NEW_SPEED**	RW	0x0	New Speed to change to at end of this frame. See SPEED field in CTRL register for encoding.
18	**BOF_TYPE**	RW	0x0	Select which version of XBOF to use.
17:12	**XBOFS**	RW	0x0	Number of Extra BOFS to transmit in SIR.
11:0	**XFER_COUNT**	RW	0x0	Number of bytes in the frame to transmit. Data may be in multiple DMA descriptors. If this register is written to, it is assumed a new frame is starting.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

24.4.3. IR Receive DMA Register Description

The IR Receive DMA Control Register configures the receive DMA.

HW_IR_RXDMA 0x80078020
HW_IR_RXDMA_SET 0x80078024
HW_IR_RXDMA_CLR 0x80078028
HW_IR_RXDMA_TOG 0x8007802C

Table 831. HW_IR_RXDMA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RUN	RSVD																					XFER_COUNT									

Table 832. HW_IR_RXDMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RUN**	RW	0x0	Tell the IR to execute the RX DMA command. The IR will clear this bit at the end of receive execution.
30:10	**RSVD**	RO	0x0	Reserved
9:0	**XFER_COUNT**	RW	0x0	Number of words to receive in a data chunk.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

24.4.4. IR Debug Control Register Description

The IR Debug Control Register includes configuration bits normally used for debugging only.

HW_IR_DBGCTRL 0x80078030
HW_IR_DBGCTRL_SET 0x80078034
HW_IR_DBGCTRL_CLR 0x80078038
HW_IR_DBGCTRL_TOG 0x8007803C

Table 833. HW_IR_DBGCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2																			VFIRSWZ	RXFRMOFF	RXCRCOFF	RXINVERT	TXFRMOFF	TXCRCOFF	TXINVERT	INTLOOPBACK	DUPLEX	MIO_RX	MIO_TX	MIO_SCLK	MIO_EN

Table 834. HW_IR_DBGCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:13	**RSVD2**	RO	0x0	Reserved
12	**VFIRSWZ**	RW	0x0	If set, swaps order of VFIR data bit pairs. NORMAL = 0 {d1,d2} = LSB, MSB SWAP = 1 {d1,d2} = MSB, LSB
11	**RXFRMOFF**	RW	0x0	If set, tries to capture SIR frames without BOF or EOF.
10	**RXCRCOFF**	RW	0x0	If set, turns off CRC checking on received frames. CRC bytes are still sent to the host.
9	**RXINVERT**	RW	0x0	If set, inverts IR_RX before processing.
8	**TXFRMOFF**	RW	0x0	If set, prevents IR from doing IRDA framing on transmits.
7	**TXCRCOFF**	RW	0x0	If set, prevents IR from calculating and inserting CRC into the Transmit frame.
6	**TXINVERT**	RW	0x0	If set, inverts IR_TX before outputting.
5	**INTLOOPBACK**	RW	0x0	If set, internally routes IR_TX to IR_RX. Use in conjunction with DUPLEX for loopback testing
4	**DUPLEX**	RW	0x0	Put IR in Duplex mode for testing.
3	**MIO_RX**	RO	0x0	Read Value on IR_RX.
2	**MIO_TX**	RW	0x0	Value to drive out on IR_TX if MIO_EN=1.
1	**MIO_SCLK**	RW	0x0	Value to drive out on IR_SCLK if MIO_EN=1.
0	**MIO_EN**	RW	0x0	MIO Enable. If set, the values written into this register get output on IR_TX and IR_SCLK.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

24.4.5. IR Interrupt Register Description

The IR Interrupt Register contains the interrupt enables and the interrupt status. The interrupt status bits report the unmasked state of the interrupts. To clear a particular interrupt status bit, write the bit-clear address with the particular bit set to 1. The enable bits control the interrupt output: a 1 will enable a particular interrupt to assert the UART interrupt output, while a 0 will disable the particular interrupt from affecting the interrupt output.

HW_IR_INTR	0x80078040
HW_IR_INTR_SET	0x80078044
HW_IR_INTR_CLR	0x80078048
HW_IR_INTR_TOG	0x8007804C

Table 835. HW_IR_INTR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2									RXABORT_IRQ_EN	SPEED_IRQ_EN	RXOF_IRQ_EN	TXUF_IRQ_EN	TC_IRQ_EN	RX_IRQ_EN	TX_IRQ_EN	RSVD1								RXABORT_IRQ	SPEED_IRQ	RXOF_IRQ	TXUF_IRQ	TC_IRQ	RX_IRQ	TX_IRQ

Table 836. HW_IR_INTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:23	**RSVD2**	RO	0x0	Reserved
22	**RXABORT_IRQ_EN**	RW	0x0	Receive Abort Interrupt Enable. DISABLED = 0x0 No Interrupt Request Enabled. ENABLED = 0x1 Interrupt Request Enabled.
21	**SPEED_IRQ_EN**	RW	0x0	Speed Change Interrupt Enable. DISABLED = 0x0 No Interrupt Request Enabled. ENABLED = 0x1 Interrupt Request Enabled.
20	**RXOF_IRQ_EN**	RW	0x0	Receive Overflow Interrupt Enable. DISABLED = 0x0 No Interrupt Request Enabled. ENABLED = 0x1 Interrupt Request Enabled.
19	**TXUF_IRQ_EN**	RW	0x0	Transmit Underflow Interrupt Enable. DISABLED = 0x0 No Interrupt Request Enabled. ENABLED = 0x1 Interrupt Request Enabled.
18	**TC_IRQ_EN**	RW	0x0	Transceiver Control Interrupt Enable. DISABLED = 0x0 No Interrupt Request Enabled. ENABLED = 0x1 Interrupt Request Enabled.
17	**RX_IRQ_EN**	RW	0x0	IR Receive Interrupt Enable. DISABLED = 0x0 No Interrupt Request Enabled. ENABLED = 0x1 Interrupt Request Enabled.
16	**TX_IRQ_EN**	RW	0x0	Transmit Interrupt Enable. DISABLED = 0x0 No Interrupt Request Enabled. ENABLED = 0x1 Interrupt Request Enabled.
15:7	**RSVD1**	RO	0x0	Reserved
6	**RXABORT_IRQ**	RW	0x0	Recieve Abort Interrupt Status. Indicates RXEN was turned off while a valid frame was being received. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
5	**SPEED_IRQ**	RW	0x0	Speed Change Interrupt Status. Indicates the completion of a speed change. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
4	**RXOF_IRQ**	RW	0x0	Receive Overflow Interrupt Status. Indicates a FIFO overflow condition while receiving a frame. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

Table 836. HW_IR_INTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**TXUF_IRQ**	RW	0x0	Transmit Underflow Interrupt Status. Indicates a FIFO underflow condition while attempting to transmit a frame. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
2	**TC_IRQ**	RW	0x0	Transceiver Control Interrupt Status. Indicates a transceiver control cycle has completed. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
1	**RX_IRQ**	RW	0x0	Receive Interrupt Status. Indicates a complete frame has been received and buffered. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
0	**TX_IRQ**	RW	0x0	Transmit Interrupt Status. Indicates a complete frame has been transmitted. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

24.4.6. IR RX Data Register Description

The IR RX Data Register is the DMA data register. It transmits (writes) or receives (reads) up to four data characters per APBX cycle.

HW_IR_DATA 0x80078050

Table 837. HW_IR_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 838. HW_IR_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x0	4 bytes of data.

DESCRIPTION:

This register is shared by the transmit and receive DMA channels. Since transmit is always a write and receive is always a read, this does not create any conflicts. A PIO write to this register pushes up to four bytes into the IR TXFIFO. A PIO read of this register reads up to four bytes from the IR RXFIFO.

EXAMPLE:

```
Empty Example.
```

24.4.7. IR Status Register Description

The IR Status Register contains flags and status of the IR block.

HW_IR_STAT 0x80078060

Table 839. HW_IR_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PRESENT	MODE_ALLOWED		ANY_IRQ	RSVD2					RXABORT_SUMMARY	SPEED_SUMMARY	RXOF_SUMMARY	TXUF_SUMMARY	TC_SUMMARY	RX_SUMMARY	TX_SUMMARY	RSVD1													MEDIA_BUSY	RX_ACTIVE	TX_ACTIVE

Table 840. HW_IR_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**PRESENT**	RO	0x1	This read-only bit indicates that the application IR function is present when it reads back a one. This application IR function is not available on a device that returns a zero for this bit field. UNAVAILABLE = 0x0 IR is not present in this product. AVAILABLE = 0x1 IR is present in this product.
30:29	**MODE_ALLOWED**	RO	0x0	This read-only field indicates the maximum mode IR that is allowed. VFIR = 0x0 VFIR speeds and below are allowed. FIR = 0x1 FIR speeds and below are allowed. MIR = 0x2 SIR and MIR are allowed. SIR = 0x3 Only SIR is allowed.
28	**ANY_IRQ**	RO	0x0	Any enabled interrupt requesting service. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
27:23	**RSVD2**	RO	0x0	Reserved
22	**RXABORT_SUMMARY**	RO	0x0	Receive Abort Interrupt enabled and requesting. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
21	**SPEED_SUMMARY**	RO	0x0	Speed Change Interrupt enabled and requesting. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
20	**RXOF_SUMMARY**	RO	0x0	Receive Overflow Interrupt enabled and requesting. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
19	**TXUF_SUMMARY**	RO	0x0	Transmit Underflow Interrupt enabled and requesting. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
18	**TC_SUMMARY**	RO	0x0	Transceiver Control Interrupt enabled and requesting. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
17	**RX_SUMMARY**	RO	0x0	IR Receive Interrupt enabled and requesting. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

Table 840. HW_IR_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	**TX_SUMMARY**	RO	0x0	Transmit Interrupt enabled and requesting. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
15:3	**RSVD1**	RO	0x0	Reserved
2	**MEDIA_BUSY**	RO	0x0	Media busy indicates IR is currently sending or has detected an active transmitter in the medium.
1	**RX_ACTIVE**	RO	0x0	IR Receive is currently receiving a valid IRDA frame.
0	**TX_ACTIVE**	RO	0x0	IR Transmit is currently busy transmitting a frame.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

24.4.8. *IR Transceiver Control Register Description*

The IR Transceiver Control Register controls both Temic style and serial interface transceivers.

HW_IR_TCCTRL 0x80078070
HW_IR_TCCTRL_SET 0x80078074
HW_IR_TCCTRL_CLR 0x80078078
HW_IR_TCCTRL_TOG 0x8007807C

Table 841. HW_IR_TCCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INIT	GO	BUSY	RSVD				TEMIC	EXT_DATA								DATA								ADDR				INDX			C

Table 842. HW_IR_TCCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**INIT**	WO	0x0	A write to this register will start a reset cycle for serial interface transceivers. Ignored for Temic.
30	**GO**	WO	0x0	A write to this register will start a control cycle. For Temic, it starts a speed change pulse. For serial interface, it starts to send out the command in fields 23:0
29	**BUSY**	RO	0x0	While a serial interface command or Temic pulse is still being processed, this bit will read 1.
28:25	**RSVD**	RO	0x0	Reserved
24	**TEMIC**	RW	0x0	Temic Pulse value to send. Only used if TC_TYPE=1. LOW = 0x0 Low Speed Pulse HIGH = 0x1 High Speed Pulse

Table 842. HW_IR_TCCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:16	**EXT_DATA**	RW	0x0	Extended Data. Only valid for serial interface writes with an extended index.
15:8	**DATA**	RW	0x0	Data or Extended Index. If INDX=1111, this is an extended index field. If INDX!=special command(1111), this is the data field.
7:5	**ADDR**	RW	0x0	Trasceiver Address for Serial Interface Command.
4:1	**INDX**	RW	0x0	Index Field for Serial Interface Command.
0	**C**	RW	0x0	C Bit for Serial Interface Command. Indicates transfer direction. 0=Read, 1=Write

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

24.4.9. IR Serial Interface Read Data Register Description

The IR Serial Interface Read Data Register contains the return data on a serial interface read command.

HW_IR_SI_READ 0x80078080

Table 843. HW_IR_SI_READ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																							ABORT	DATA							

Table 844. HW_IR_SI_READ Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:9	**RSVD1**	RO	0x0	Reserved
8	**ABORT**	RO	0x0	When high, indicates the last Serial Transceiver read was not completed. This occurs when the transceiver fails to drive a START bit to the 4230 within four IR_SCLK cycles
7:0	**DATA**	RO	0x0	Data returned by the transceiver on a read command.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

24.4.10. IR Debug Register Description

This is the IR Debug Register.

HW_IR_DEBUG 0x80078090

Table 845. HW_IR_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1																										TXDMAKICK	RXDMAKICK	TXDMAEND	RXDMAEND	TXDMAREQ	RXDMAREQ

Table 846. HW_IR_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:6	**RSVD1**	RO	0x0	Reserved
5	**TXDMAKICK**	RO	0x0	DMA kick toggle line for transmitter.
4	**RXDMAKICK**	RO	0x0	DMA kick toggle line for receiver.
3	**TXDMAEND**	RO	0x0	Transmit DMA Command End signal value.
2	**RXDMAEND**	RO	0x0	Receive DMA Command End signal value.
1	**TXDMAREQ**	RO	0x0	Transmit DMA Request signal value.
0	**RXDMAREQ**	RO	0x0	Receive DMA Request signal value.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

IR XML Revision: 1.29

25. AUDIOIN/ADC

This chapter describes the AUDIOIN/ADC module implemented on the STMP36xx, including DMA, sample rate conversion, and internal operation. Programmable registers are described in Section 25.6.

25.1. Overview

The STMP36xx features an audio record path that consists of a sigma-delta analog-to-digital converter (ADC), followed by the AUDIOIN digital multi-stage Finite Impulse Response (FIR) filter.

The microphone or line input is oversampled by the ADC, and the 1-bit digital stream is input to a cascaded-integrator comb filter, where the signal is parallelized, sent through a high-pass filter to remove DC offset, and the sample rate is converted to the AUDIOIN's internal rate. Next, the signal is filtered using a three-stage FIR filter. The resultant parallel PCM samples are then transferred to a buffer in memory using the APBX bridge DMA, where it can be read by system software.

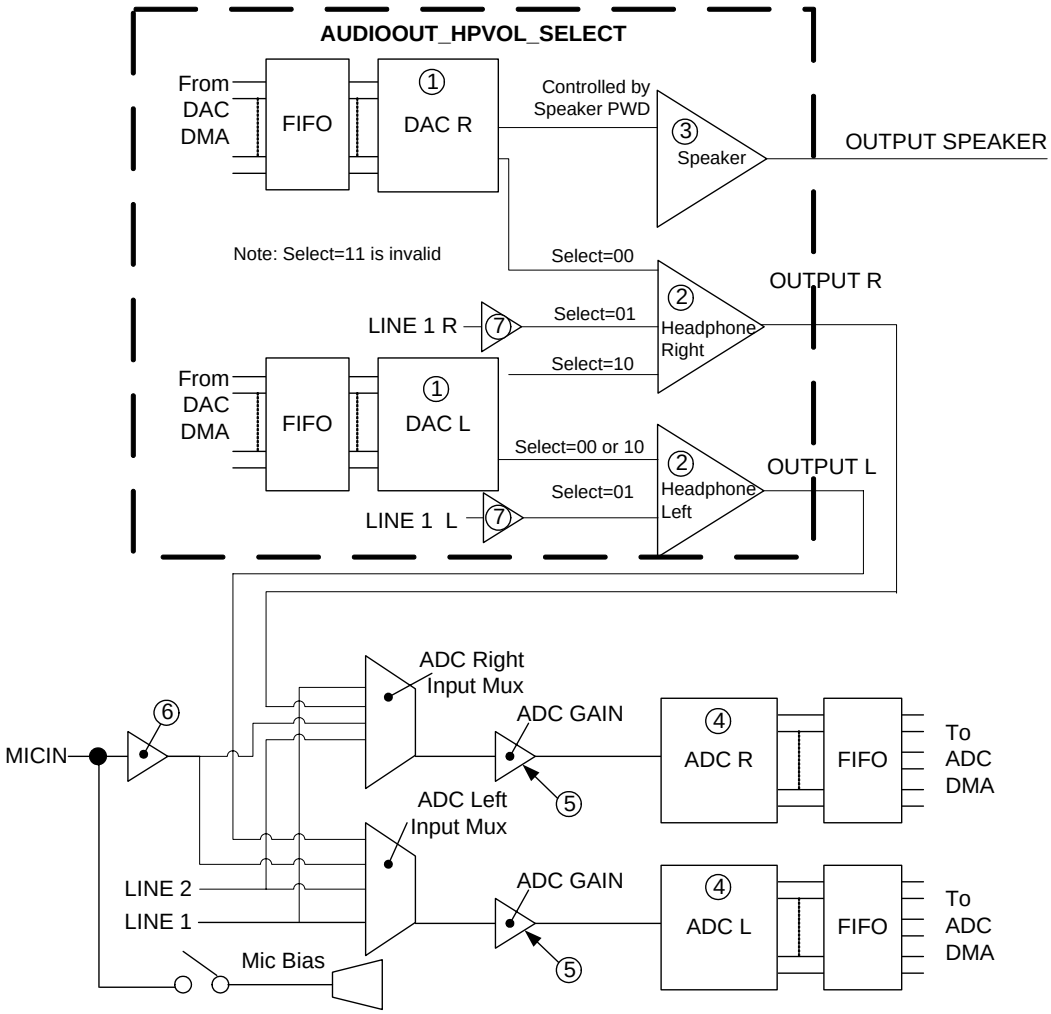
The analog audio source can be selected from one of three possible inputs:

- Mono microphone input
- Stereo line inputs
- Looped back from the stereo headphone amplifier

The AUDIOIN module implements the following functions:

- Serial to parallel bit-stream integrator/averager
- Sample rate converting (SRC) cascaded-integrator comb (CIC) filter
- High-pass filter (HPF)
- Three-stage downsampling FIR filter: 7-tap (8:4), 11-tap (4:2), 33-tap (2:1) supporting conversion from quarter, half, full, double, and quad sample rates that are multiples of the standard 32 kHz, 44.1 kHz, and 48 kHz rates
- 16- or 32-bit PCM sample widths
- APBX bridge DMA interface
- Independent control of each channel's volume (including mute)
- DAC-to-ADC internal loopback for product development
- Control bit fields used for analog ADC settings

Figure 111 shows the audio path and control options, and Figure 112 is a high-level block diagram of the AUDIOIN module.



Notes:

1. audioout_dacvolume: Digital volume control.
2. audioout_hpvol: Analog volume control.
3. audioout_spkrvol: Analog volume control that works on the speaker amp output.
4. audioin_adcvolume: Digital volume control.
5. audioin_adcvol: Analog volume control that controls the ADC Gain block.
6. audioin_micline_micgain: Analog volume control that controls the mic amp.
7. atten_line bit

Figure 111. Mixed Signal Audio Elements

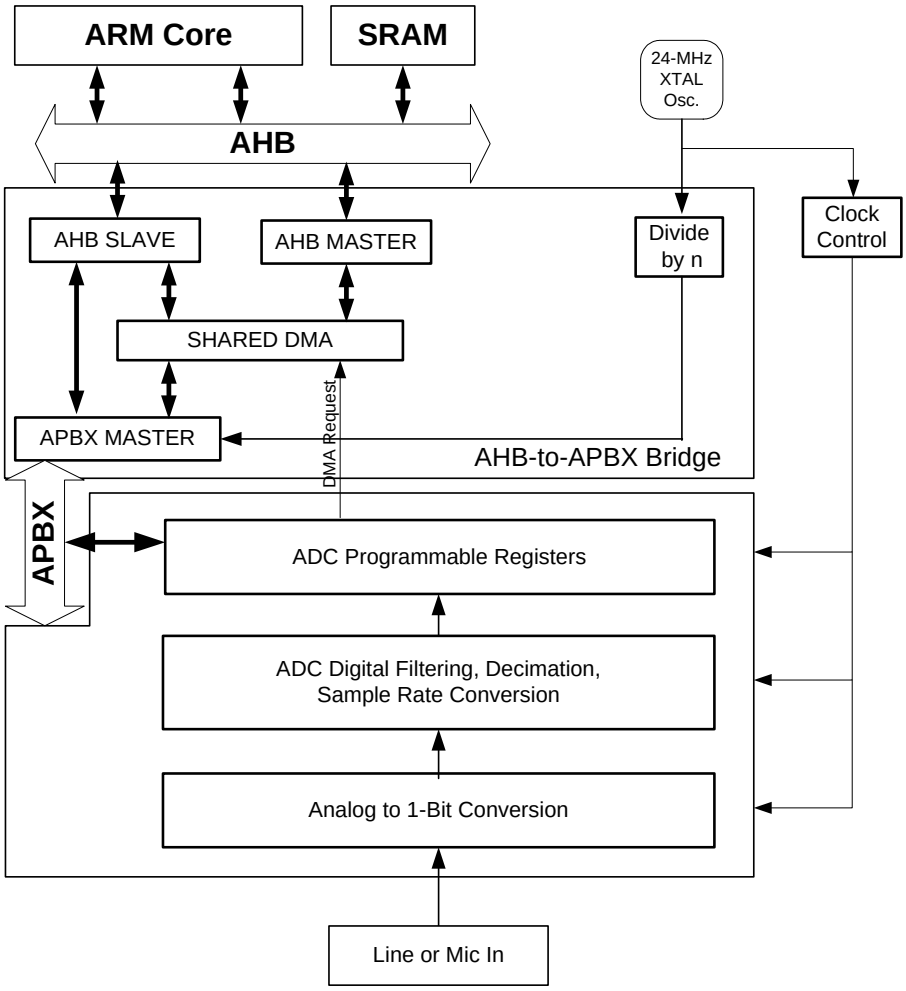


Figure 112. AUDIOIN/ADC Block Diagram

25.2. Operation

The first step in receiving audio to the AUDIOIN module requires the analog-to-digital converter (ADC). The STMP36xx includes a high-performance analog stereo sigma-delta ADC. It converts analog audio to two (left and right channel) single-bit digital streams that are input to the AUDIOIN module, along with a clock that runs at the sigma-delta oversampling clock rate. The AUDIOIN module includes hardware for oversampling, decimation, and arbitrary sample rate conversion. The 1-bit stream is input to a cascaded-integrator comb filter where serial-to-parallel data conversion, as well as sample rate conversion, takes place, along with a high-pass filter to eliminate DC offset. Serial audio is first input to an averager that initially converts samples to 8-bit values. The CIC then interpolates/decimates as well as sign-extends the parallel data, converting the samples from the programmed standard external sample rate to the AUDIOIN module's internal rate. The resultant 24-bit PCM samples are then stored to the module's RAM.

These 24-bit samples are then filtered using a three-stage FIR filter, consisting of 7, 11, and 33 taps, respectively. The AUDIOIN contains a sequencer, multiply-accumulate hardware, and a set of filter coefficients that performs successive iterations on the data stored in RAM. Intermediate data that is calculated along the taps/stages of the FIR are also stored in the AUDIOIN's RAM. The resultant filtered PCM data is then stored in a FIFO that can either be directly accessed by the host CPU or read by the STMP36xx's AHB-APBX bridge DMA engine to store the data in on- or off-chip memory to allow access to system software.

In most cases, access to the AUDIOIN's data is made by the AHB-APBX bridge DMA. DMA channel 0 is dedicated to the AUDIOIN module. The DMA moves data from the AUDIOIN's memory-mapped data register to a RAM buffer every time a request is made. The buffer may be in on- or off-chip RAM. It is also possible for the CPU to manually move data from the AUDIOIN data register while monitoring either the FIFO or DMA request status bits in the AUDIOIN debug register (HW_AUDIOIN_ADCDEBUG).

Also present on the STMP36xx is an audio playback path called AUDIOOUT/DAC. Although each functions independently of one another, both the AUDIOIN and AUDIOOUT blocks share their FIR filter (sequencer/RAM/coefficients) and DMA controller. This combined module is titled the "digital filter" or DIGFILT. The register descriptions that follow both refer to each path independently (AUDIOIN and AUDIOOUT) as well as a whole (DIGFILT), due to the fact that clocks and resets affect either the shared resources or the design as a whole.

In order to configure the AUDIOIN/ADC for operation, the user must first clear the clock gate (CLKGATE) and soft reset (SFTRST) bits within the AUDIOIN control register (HW_AUDIOIN_CTRL). The run bit should remain off (zero), while all other control bits are initialized. It is important to note that there are also a number of control bits within the AUDIOOUT's address space that control functions within the analog ADC. The user must clear the clock gate and soft reset of the AUDIOOUT block in order to program these bits. Next, the bridge DMA controller channel 0 should be programmed and enabled to collect input audio samples to one or more RAM buffers. Finally, the run bit should be set to start AUDIOIN/ADC operation.

Each 32-bit register within the AUDIOIN's address space is aliased to four adjacent words. The first word is used for normal read-write access while the subsequent three words are contained within the register's set-clear-toggle (SCT) address space. Only bits that are written to with a one in this space are affected. For example, writing a one to bit using the register's set address sets that particular bit, while maintaining the state of all other bits. This convention allows easy bit manipulation without requiring the standard read-modify-write procedure. Bits that are written with a one to the register's clear address clear the bit, while the toggle address causes bits to invert their current state.

25.2.1. AUDIOIN DMA

The DMA is typically controlled by a linked list of descriptors. The descriptors are usually circularly linked, causing the DMA to cycle through the set of DMA buffers. The DMA can be programmed to assert an IRQ when some or all of the buffers have been filled.

For example, AUDIOIN DMA descriptor 0 may program the DMA to fill a buffer, set the done IRQ, and fetch descriptor 1. Descriptor 1 programs the DMA to fill the next buffer. The DMA continues to operate normally while the IRQ is asserted. The CPU

needs to respond to the IRQ before the DMA has filled all of the buffers. The DMA ISR clears the IRQ flag and informs the operating system that the buffers are filled.

In general, software copies data out of the buffers or adjusts the descriptors to point to other empty buffers. Software should also take advantage of the DMA's counting semaphore feature to synchronize the addition of new descriptors to the chain.

The DMA can put the AUDIOIN's PCM data into any memory-mapped location. For 32-bit PCM data, the left-channel sample is stored first in the lowest address, followed by the corresponding right-channel sample in the next word address (+4 bytes). For 16-bit mode, sample pairs are stored in each word. Right samples are stored in the upper half-word while left samples are stored in the lower half-word. Because the AUDIOIN always operates on stereo data, the PCM buffer should always have an integer number of words. The audio data values are in two's complement format, where full-scale values range from 0x7FFFFFFF to 0x80000000 for 32-bit data or 0x7FFF to 0x8000 for 16-bit data.

In addition to the DMA IRQ used to indicate a filled AUDIOIN buffer, the module also has an overflow and underflow IRQ. Underflows should never occur, because (by design) the DMA should never attempt to read more data than is present within the AUDIOIN's FIFO. However, if the AUDIOIN ever attempts to write data into a full FIFO, an overflow occurs. This causes the overflow flag to be set in the AUDIOIN control register (HW_AUDIOIN_CTRL). If the overflow/underflow IRQ enable bit is set, then this condition also asserts an interrupt. The interrupt is cleared by writing a one to the overflow flag in the HW_AUDIOIN_CTRL's SCT clear address space. An AUDIOIN underflow is typically caused by the DMA running out of new buffers, or if the AHB or APBX is stalled or are otherwise unable to meet the bandwidth requirements at the current operating frequency. If the counting semaphore reaches 0, the DMA stops processing new descriptors and stops moving data from the AUDIOIN's data register (HW_AUDIOIN_DATA).

25.3. ADC Sample Rate Converter and Internal Operation

Table 847 contains the required value of the HW_AUDIOOUT_ADCSSR register for various common sample rates. To make small sample rate adjustments (for example to track Fs fluctuations during a mix with an FM output to the DAC), the user may change the last few LSBs of the SRC_FRAC bit field to speed or slow the rate of sample consumption until equilibrium between the ADC's sample rate and the rate of another audio stream is met. Note that, unlike the DAC, only small deviations to SRC_FRAC can be made. The only valid values for BASEMULT, SRC_HOLD, and SRC_INT are listed in Table 847.

Table 847. Bit Field Values for Standard Sample Rates

SAMPLE RATE	HW_AUDIOOUT_ADCSSR			
$Fsample_{ADC}$	BASEMULT	SRC_HOLD	SRC_INT	SRC_FRAC
192,000 Hz	0x4	0x0	0x0F	0x13FF
176,400 Hz	0x4	0x0	0x11	0x0037
128,000 Hz	0x4	0x0	0x17	0x0E00
96,000 Hz	0x2	0x0	0x0F	0x13FF
88,200 Hz	0x2	0x0	0x11	0x0037
64,000 Hz	0x2	0x0	0x17	0x0E00
48,000 Hz	0x1	0x0	0x0F	0x13FF

Table 847. Bit Field Values for Standard Sample Rates (Continued)

SAMPLE RATE	HW_AUDIOOUT_ADCSSR			
$Fsample_{ADC}$	BASEMULT	SRC_HOLD	SRC_INT	SRC_FRAC
44,100 Hz	0x1	0x0	0x11	0x0037
32,000 Hz	0x1	0x0	0x17	0x0E00
24,000 Hz	0x1	0x1	0x0F	0x13FF
22,050 Hz	0x1	0x1	0x11	0x0037
16,000 Hz	0x1	0x1	0x17	0x0E00
12,000 Hz	0x1	0x3	0x0F	0x13FF
11,025 Hz	0x1	0x3	0x11	0x0037
8,000 Hz	0x1	0x3	0x17	0x0E00

Note: Sample rates greater than 48 kHz can only be used when the AUDIOOUT is disabled, and 44.1 kHz is the maximum sample rate at which both the AUDIOIN and AUDIOOUT can operate simultaneously.

For any of the desired sample rates, the internal sample-rate conversion factor is calculated according to the following formula:

$$SRConv_{ADC} = 65536 * [Fanalog_{ADC})/ (8 * Fsample_{ADC})]$$

The 1-bit sigma delta A/D converter is always sampled on a submultiple of the 24.0-MHz crystal oscillator frequency, as specified in the HW_CCR_ADCDIV register (see Figure 113). This divider generates sample strobes at $Fanalog_{ADC}$ where the divisors available come from the set {4,6,8,12,16,24}. It is recommended that ADCDIV always be set to 000 so that a 6.0-MHz 1-bit A/D sample rate is used. The sample strobe is used to integrate the 1-bit A/D values. As shown in Figure 113, these integrated values are filtered and then delivered to the ADC DMA to write into on-chip RAM.

Notice that the integrators run continuously while the filters produce samples at the decimated rate. Depending on the decimation or over-sample ratio of the CIC filter engine, the integrators will produce samples of various precisions and scale factors. The filtered values written to the ADC FIFO are signed 16-bit or 24-bit numbers with the conversion data LSB-justified, i.e., downscaled in the lower end of the word.

The scale factor column of the 48-kHz family of sample rates satisfies the property:

$$24.576 \text{ MHz} = Q*Fsample_{ADC} \text{ where Q comes from the set of integers}$$

These sample rates include 48 kHz, 32 kHz, 24 kHz, 16 kHz, 12 kHz, and 8 kHz.

There are also the members of the 44.1-kHz family, whose members satisfy the property:

$$16.9344 \text{ MHz} = Q*Fsample_{ADC} \text{ where Q comes from the set of integers}$$

These sample rates include 44.1 kHz, 22.05 kHz, and 11.025 kHz.

Since 24.576 kHz and 16.9344 MHz are relatively prime to 24.0 MHz, members of the 48-kHz family and 44.1-kHz family are related to the 24.0-MHz source clock by the relationship:

24.0 MHz=P*Fsample$_{ADC}$, where P is a rational number

The A/D block includes a variable rate or rational decimator as shown in Figure 113 to accommodate these sample rates. Rational numbers in the ADC are approximated with a scaled fixed-point 24-bit value. In this case, the decimal point falls between bit 15 and bit 16. Therefore, the lower two bytes hold the fractional part, while the upper byte holds the whole number portion of the scaled fixed point. The position register uses this scaled fixed-point representation to hold the number of 1-bit samples to be dropped (decimated) to find the next sample at which to produce a filtered multibit sigma delta A/D value to send to the DMA. Whenever the whole number part (bits 23:16) is zero, then a sample is produced.

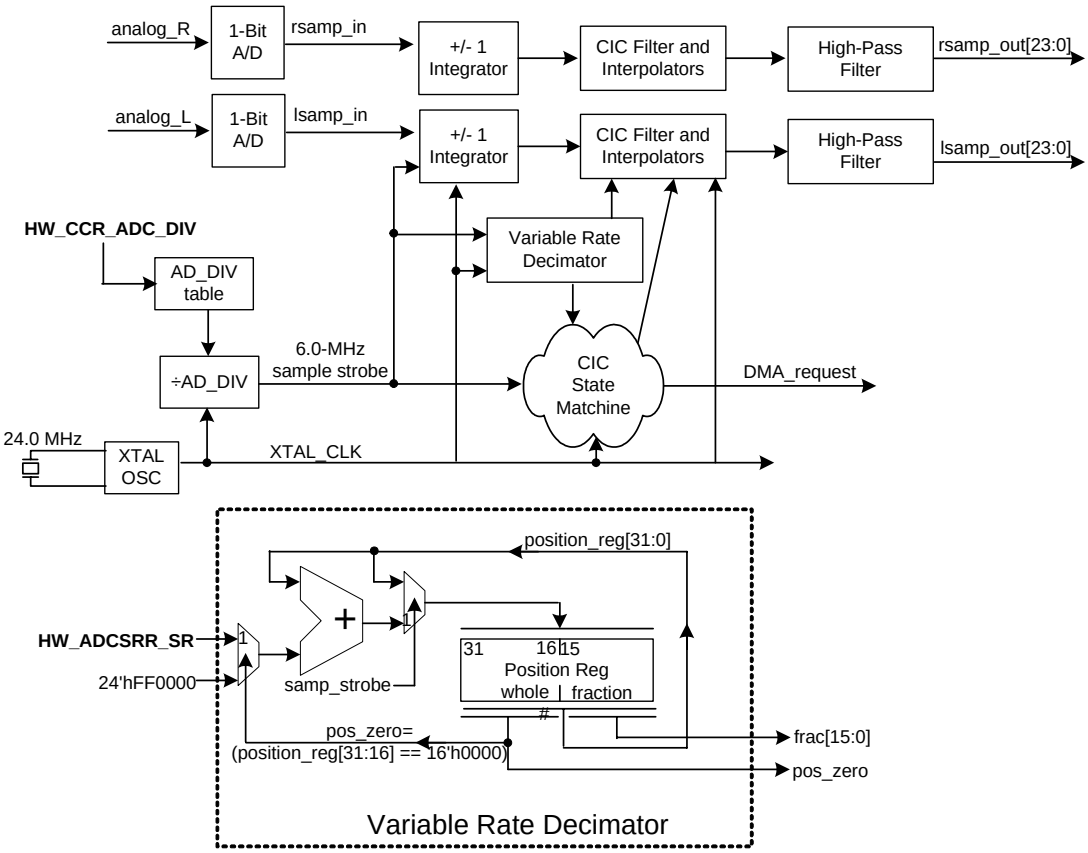


Figure 113. Variable-Rate A/D Converter

The range of values of the samples stored into the on-chip RAM is proportional to the square of the over-sample rate (OSR) used in the capture process. The larger the OSR, the longer period the integrators run in the ADC. As a result, the range of values seen for the same signal wave form captured at the same sample rate but with two different OSR will be different.

For example:

- An 8-kHz microphone captured at F_{ADC} = 6.0 MHz will be 36 times smaller than the values resulting from capturing the same source signal at F_{ADC} = 1.0 MHz.

- The peak range of values seen in a capture of a signal at 44.1 kHz with $Fanalog_{ADC}$ = 6.0 MHz is ±3200 decimal.
- The oversample ratio in this case is OSR= 136.054.
- Calculate a magnitude constant, K_{filter} for ADC's filter from this as $K_{filter} = OSR^2/\text{Peak Value} = (136.054)^2/3200 = 5.7846$.
- For any OSR in any sample rate, the peak value can be approximated by $Value_{peak} = OSR^2 / K_{filter}$.

In signal processing, one frequently normalizes the range of values to ±1.0, as seen in a fixed-point scaled integer[1]. For a 24-bit DSP, the fixed point is placed between bit 23 and the sign bit (bit 24) (bit 1 = 2^0). So the desired maximum excursion is then ± 2^{23} or ±8388608, decimal.

One can calculate a normalization constant to multiply all incoming samples for each sampling condition from the following equation (note that OSR is fixed at 6 MHz for the STMP36xx):

$ScaleFactor = 2^{23} * K_{filter} / OSR^2$

If the incoming sample stream is multiplied, sample by sample, by ScaleFactor, then normalized ±1.0 samples result. All data output from the DIGFILT ADC are scaled according to this equation.

25.4. Microphone

The external microphone needs a bias voltage to enable it to operate. This bias voltage can be generated externally using discrete components as shown in Figure 114. Or, if either the LRADC0 or LRADC1 pin is available, it can be used to supply a bias voltage from an on-chip generator, as shown in Figure 115. To enable the generation of the microphone bias voltage on pin LRADC0 or LRADC1, the two MIC_RESISTOR bits in the HW_AUDIOIN_MICLINE register need to be written with required values for desired internal resistor selection. To select either pin LRADC1 or LRADC0 as the microphone bias source, write the MIC_SELECT bit in the HW_AUDIOIN_MICLINE register as follows: 0 for pin LRADC0, 1 for pin LRADC1.

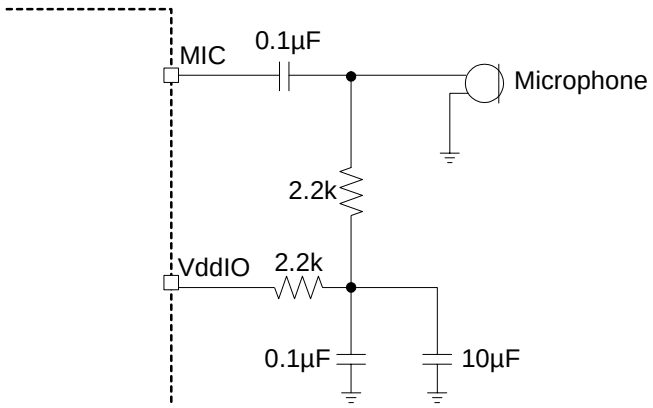


Figure 114. External Microphone Bias Generation

1.A normalized two's complement 24-bit number cannot actually express a value of +1.0 without overflowing.

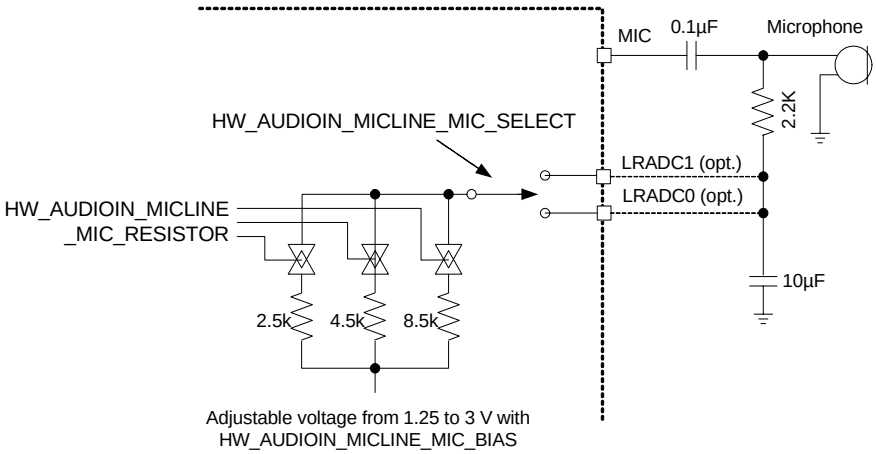


Figure 115. Internal Microphone Bias Generation

25.5. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

25.6. Programmable Registers

The following registers provide control for programmable elements of the AUDIOIN/ACD block.

25.6.1. AUDIOIN Control Register Description

The AUDIOIN Control Register provides overall control of the digital portion of the analog-to-digital converter.

HW_AUDIOIN_CTRL 0x8004C000
HW_AUDIOIN_CTRL_SET 0x8004C004
HW_AUDIOIN_CTRL_CLR 0x8004C008
HW_AUDIOIN_CTRL_TOG 0x8004C00C

Table 848. HW_AUDIOIN_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSRVD3										DMAWAIT_COUNT				RSRVD1					LR_SWAP	EDGE_SYNC	INVERT_1BIT	OFFSET_ENABLE	HPF_ENABLE	WORD_LENGTH	LOOPBACK	FIFO_UNDERFLOW_IRQ	FIFO_OVERFLOW_IRQ	FIFO_ERROR_IRQ_EN	RUN

Table 849. HW_AUDIOIN_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	AUDIOIN Module Soft Reset. Setting this bit to one forces a reset to portions of DIGFILT that control audio input and then gates the clocks off because the CLKGATE bit's reset state is to disable clocks. This bit must be cleared to zero for normal operation. Note that the CLKGATE bit does not affect SFTRST, because it must remain writeable during clock gating.
30	**CLKGATE**	RW	0x1	AUDIOIN Clock Gate Enable. When this bit is set to 1, it gates off the clocks to the portions of the DIGFILT block that control only input audio functions. It does not affect portions of the block that control AUDIOOUT. Clear the bit to zero for normal AUDIOIN operation. Note that when this bit is set, it remains writeable during clock gating so that it may be disabled by the user.
29:21	**RSRVD3**	RO	0x0	Reserved
20:16	**DMAWAIT_COUNT**	RW	0x0	DMA Request Delay Count. This bit field specifies the number of APBX clock cycles (0 to 31) to delay before each DMA request. This field acts as a throttle on the bandwidth consumed by the DIGFILT block. This field can be loaded by the DMA.
15:11	**RSRVD1**	RO	0x0	Reserved
10	**LR_SWAP**	RW	0x0	Left/Right Input Channel Swap Enable. Setting this bit to one swaps the left and right serial audio inputs from the ADC before being parallelized and having the sample rate converted by the AUDIOIN's CIC block.
9	**EDGE_SYNC**	RW	0x0	Serial Input Clock Edge Sync Select. This bit selects the edge of the ADC's serial input clock upon which the CIC-filter synchronizes for data receive. 0=Rising edge. 1=Falling edge
8	**INVERT_1BIT**	RW	0x0	Invert Serial Audio Input Enable. When set, this bit inverts the 1-bit serial input of both left and right channels from the ADC's sigma-delta modulator. 0=Normal operation. 1=Invert L/R serial audio input to the CIC block.
7	**OFFSET_ENABLE**	RW	0x1	ADC Analog High-Pass Filter Offset Calculation Enable. When this bit is set, the ADC's high pass filter actively adjusts the serial audio input, removing DC offset present within the signal. Active DC offset only takes place when the HPF_ENABLE bit is set. Once DC offset has been achieved, this bit can be cleared to maintain a constant level of offset. After clearing this bit, the HPF_ENABLE bit should remain set to maintain a constant DC offset.

Table 849. HW_AUDIOIN_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	**HPF_ENABLE**	RW	0x1	ADC High-Pass Filter Enable. When this bit is set, the ADC's analog high pass filter is enabled. Once enabled, the OFFSET_ENABLE bit can be set to cause the filter to begin removing DC offset from the incoming serial analog data. Once DC offset has been removed, the OFFSET_ENABLE bit should be cleared while the HPF_ENABLE bit remains set.
5	**WORD_LENGTH**	RW	0x0	PCM Audio Bit Size Select. This bit selects the size of the parallel PCM data collected by the AUDIOIN's input FIFO. 0=32-bit PCM samples. 1=16 bit samples. Note that the PCM audio data output from the FIR filter stages is 24 bits. For 16-bit operation, the resultant data is normalized by dropping the least significant 8 bits. For 32-bit mode, the two's complement PCM data is sign extended to 32 bits.
4	**LOOPBACK**	RW	0x0	AUDIOOUT-to-AUDIOIN Loopback Enable. Setting this bit to one connects the AUDIOOUT's digital serial data from the SDM module to the AUDIOIN's serial digital input to the CIC module, bypassing the analog DAC and ADC. This test mode provides a digital-only loopback which ties the output filter chain back to the input filter chain. This bit should be cleared to zero for normal operation.
3	**FIFO_UNDERFLOW_IRQ**	RW	0x0	FIFO Underflow Interrupt Status Bit. This bit is set by hardware if the AUDIOIN's FIFO underflows any time during operation. It is reset by software by writing a one to the SCT clear address space. An interrupt is issued to the host processor if this bit is set and FIFO_ERROR_IRQ_EN=1. Note that underflows should not occur by design because requests to the DMA are not made unless there is data present within the FIFO, and would indicate a serious DMA error.
2	**FIFO_OVERFLOW_IRQ**	RW	0x0	FIFO Overflow Interrupt Status Bit. This bit is set by hardware if the AUDIOIN's FIFO overflows due to a DMA request that is not serviced in time. It is reset by software writing a one to the SCT clear address space. An interrupt is issued to the host processor if this bit is set and FIFO_ERROR_IRQ_EN=1.
1	**FIFO_ERROR_IRQ_EN**	RW	0x0	FIFO Error Interrupt Enable. Set this bit to one to enable an AUDIOIN interrupt request to the host processor when either the FIFO overflow or underflow status bits are set. Note that this bit does not affect the state of the underflow/overflow status bits, but rather their ability to signal an interrupt to the CPU.
0	**RUN**	RW	0x0	AUDIOIN Enable. Setting this bit to one causes the AUDIOIN to begin converting data. Once 8 words of audio input samples are collected in its FIFO, it makes a DMA service request. Clearing this bit to zero stops data conversion and also causes the CLKGATE bit to be set.

DESCRIPTION:

The AUDIOIN Control Register contains bit fields used to control and monitor AUDIOIN operation including: reset, clocks, DMA transfers, analog ADC signal interface, high-pass filter operation, PCM data size, test, and interrupt control.

EXAMPLE:

```
HW_AUDIOIN_CTRL.RUN = 1; // start AUDIOIN conversion
```

25.6.2. AUDIOIN Status Register Description

The AUDIOIN Status Register is used to determine if the digital-to-analog converter is operational.

HW_AUDIOIN_STAT 0x8004C010
HW_AUDIOIN_STAT_SET 0x8004C014
HW_AUDIOIN_STAT_CLR 0x8004C018
HW_AUDIOIN_STAT_TOG 0x8004C01C

Table 850. HW_AUDIOIN_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ADC_PRESENT	RSRVD3																														

Table 851. HW_AUDIOIN_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**ADC_PRESENT**	RO	0x1	AUDIOIN Functionality Present. This status bit is set to one in products that include the AUDIOIN/ADC. If this bit is zero, the AUDIOIN/ADC is permanently disabled and cannot be operated by the user.
30:0	**RSRVD3**	RO	0x0	Reserved

DESCRIPTION:

The AUDIOIN Status Register provides an indication of the presence of the ADC functionality.

EXAMPLE:

```
unsigned TestValue= HW_AUDIOIN_STAT.ADC_PRESENT;
```

25.6.3. AUDIOIN Sample Rate Register Description

The AUDIOIN Sample Rate Register is used to specify the sample rate from which the incoming serial audio data is converted as it is received by the CIC module from the analog ADC.

HW_AUDIOIN_ADCSRR 0x8004C020
HW_AUDIOIN_ADCSRR_SET 0x8004C024
HW_AUDIOIN_ADCSRR_CLR 0x8004C028
HW_AUDIOIN_ADCSRR_TOG 0x8004C02C

Table 852. HW_AUDIOIN_ADCSRR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OSR	BASEMULT			RSRVD2	SRC_HOLD			RSRVD1				SRC_INT				RSRVD0			SRC_FRAC												

Table 853. HW_AUDIOIN_ADCSRR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**OSR**	RO	0x0	AUDIOIN Oversample Rate. Note that for the STMP36xx, the oversample rate is fixed at 6 MHz. OSR6 = 0x0 AUDIOIN oversample rate at 6 MHz. OSR12 = 0x1 AUDIOIN oversample rate at 12 MHz.
30:28	**BASEMULT**	RW	0x1	Base Sample Rate Multiplier. This bit field is used to configure the ADC's sample rate to one of three ranges: single, double, or quad. This multiply factor is used to achieve sample rates greater than the standard rates of 32/44.1/48 kHz. A value of 0x1 should be used when selecting half and quarter sample rates. Note that sample rates greater than 48 kHz may only be used when the AUDIOOUT is disabled, and 44.1 kHz is the maximum sample rate at which both the AUDIOIN and AUDIOOUT can operate simultaneously. SINGLE_RATE = 0x1 Single-rate multiplier (for 48/44.1/32 kHz as well as half and quarter rates). DOUBLE_RATE = 0x2 Double-rate multiplier (for 96/88.2/64 kHz). QUAD_RATE = 0x4 Quad-rate multiplier (for 192/176.4/128 kHz).
27	**RSRVD2**	RO	0x0	Reserved. Always write a zero to this bit field.
26:24	**SRC_HOLD**	RW	0x0	Sample Rate Conversion Hold Factor. This bit is used to hold a sample of a variable number of clock cycles in order to generate half and quarter sample rates when dividing down the AUDIOIN's internal rate using the equation: output_sample_rate = (6x10^6 * BASEMULT) / (SRC_INT.SRC_FRAC * 8 * (SRC_HOLD + 1)). Refer to the sample rate table earlier in this chapter that provides a list of bit field values required to achieve all common sample rates.
23:21	**RSRVD1**	RO	0x0	Reserved. Always write zeros to this bit field.
20:16	**SRC_INT**	RW	0x11	Sample Rate Conversion Integer Factor. This bit field is the integer portion of a divide term used to sample-rate-convert the AUDIOIN's internal rate using the equation; output_sample_rate = (6x10^6 * BASEMULT) / (SRC_INT.SRC_FRAC * 8 * (SRC_HOLD + 1)). Refer to the sample rate table earlier in this chapter that provides a list of bit field values required to achieve all common sample rates.

Table 853. HW_AUDIOIN_ADCSRR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:13	**RSRVD0**	RO	0x0	Reserved. Always write zeros to this bit field.
12:0	**SRC_FRAC**	RW	0x37	Sample Rate Conversion Fraction Factor. This bit field is the fractional portion of a divide term used to sample-rate-convert the AUDIOIN's internal rate using the equation; output_sample_rate = (6x10^6 * BASEMULT) / (SRC_INT.SRC_FRAC * 8 * (SRC_HOLD + 1)). Refer to the sample rate table earlier in this chapter that provides a list of bit field values required to achieve all common sample rates.

DESCRIPTION:

The AUDIOIN Sample Rate Register contains bit fields used to specify the rate at which the ADC samples incoming analog audio.

EXAMPLE:

```
// Program the ADC for a sample rate of 48 kHz:
HW_AUDIOOUT_DACSRR.BASEMULT = 0x1; // quad-rate
HW_AUDIOOUT_DACSRR.SRC_HOLD = 0x0; // 0 for full- double- quad-rates
HW_AUDIOOUT_DACSRR.SRC_INT = 0xF; // 15 for the integer portion
HW_AUDIOOUT_DACSRR.SRC_FRAC = 0x13FF; // the fractional portion
```

25.6.4. AUDIOIN Volume Register Description

The AUDIOIN Volume Register is used to adjust the signal level of the recorded audio input from the ADC.

HW_AUDIOIN_ADCVOLUME 0x8004C030
HW_AUDIOIN_ADCVOLUME_SET 0x8004C034
HW_AUDIOIN_ADCVOLUME_CLR 0x8004C038
HW_AUDIOIN_ADCVOLUME_TOG 0x8004C03C

Table 854. HW_AUDIOIN_ADCVOLUME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD6			VOLUME_UPDATE_LEFT	RSRVD5		EN_ZCD	RSRVD4	VOLUME_LEFT								RSRVD3			VOLUME_UPDATE_RIGHT	RSRVD2			RSRVD1	VOLUME_RIGHT							

Table 855. HW_AUDIOIN_ADCVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSRVD6**	RO	0x00	Reserved
28	**VOLUME_UPDATE_LEFT**	RO	0x1	Left Channel Volume Update Pending. This bit is set to one by the hardware when an AUDIOIN volume update is pending, i.e., waiting on a zero crossing on the left channel. The bit is set following a write to the VOLUME_LEFT bit field and is cleared when the attenuation value is applied to the PCM input stream (at a zero-crossing). This status bit is not used when EN_ZCD=0.
27:26	**RSRVD5**	RO	0x00	Reserved
25	**EN_ZCD**	RW	0x0	Enable Zero Cross Detect. This bit enables/disables use of the zero cross detect circuit in the ADC (rather than enabling the circuit itself). When enabled, changes to the volume bit fields are not applied until it is detected that the input signal's sign bit toggles (crosses zero amplitude). When disabled, changes to the volume bit fields take effect immediately when written.
24	**RSRVD4**	RO	0x0	Reserved
23:16	**VOLUME_LEFT**	RW	0xff	Left Channel Volume Setting. This bit field is used to establish the incoming PCM audio signal strength during record. Volume ranges from full scale -0.5dB (0xFE) to -100dB (0x37). Each increment of this bit field causes a half dB increase in volume. Note that values 0x00-0x37 all produce the same attenuation level of -100dB, and a value of 0xFF is reserved. Also note that the several bit fields exist for the analog ADC that should be used to adjust the realitive gain of the input signal to the AUDIOIN block.VOLUME_LEFT and VOLUME_RIGHT must be set to identical values whenever attenuation is changed.
15:13	**RSRVD3**	RO	0x00	Reserved
12	**VOLUME_UPDATE_RIGHT**	RO	0x1	Right Channel Volume Update Pending. This bit is set to one by the hardware when an AUDIOIN volume update is pending, i.e., waiting on a zero crossing on the right channel. The bit is set following a write to the VOLUME_RIGHT bit field and is cleared when the attenuation value is applied to the PCM input stream (at a zero-crossing). This status bit is not used when EN_ZCD=0.
11:9	**RSRVD2**	RO	0x00	Reserved

Table 855. HW_AUDIOIN_ADCVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	**RSRVD1**	RO	0x0	Reserved
7:0	**VOLUME_RIGHT**	RW	0xff	Right Channel Volume Setting. This bit field is used to establish the incoming PCM audio signal strength during record. Volume ranges from full scale -0.5dB (0xFE) to -100dB (0x37). Each increment of this bit field causes a half dB increase in volume. Note that values 0x00-0x37 all produce the same attenuation level of -100dB, and a value of 0xFF is reserved. Also note that the several bit fields exist for the analog ADC that should be used to adjust the realitive gain of the input signal to the AUDIOIN block. VOLUME_RIGHT and VOLUME_LEFT must be set to identical values whenever attenuation is changed.

DESCRIPTION:

The AUDIOIN Volume Register allows volume control of the left and right channels. Always program the VOLUME_LEFT and VOLUME_RIGHT bit fields to identical values whenever attenuation is changed. Input audio can be attenuated in 0.5-dB steps, from full scale down to a minimum of -100 dB. This register is also used to enable/control volume updates such that they are only applied when PCM values cross zero to prevent unwanted audio artifacts.

EXAMPLE:

```
HW_AUDIOIN_ADCVOLUME.U = 0x00ff00ff; maximum volume for left and right channels.
```

25.6.5. AUDIOIN Debug Register Description

The AUDIOIN Debug Register is used for testing and debugging the AUDIOIN block.

HW_AUDIOIN_ADCDEBUG 0x8004C040
HW_AUDIOIN_ADCDEBUG_SET 0x8004C044
HW_AUDIOIN_ADCDEBUG_CLR 0x8004C048
HW_AUDIOIN_ADCDEBUG_TOG 0x8004C04C

Table 856. HW_AUDIOIN_ADCDEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ENABLE_ADCDMA	RSRVD1																											ADC_DMA_REQ_HAND_SHAKE_CLK_CROSS	SET_INTERRUPT3_HAND_SHAKE	DMA_PREQ	FIFO_STATUS

Table 857. HW_AUDIOIN_ADCDEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**ENABLE_ADCDMA**	RW	0x0	AUDIOIN Digital Path Test Enable. This bit is used solely for development and debug and is not functional on production parts. When enabled, it causes the AUDIOIN's serial audio data input to bypass the CIC block, to be assembled into 32-bit words and transferred out to memory using the AUDIOOUT's DMA Channel 1. Unlike loopback, this test mode provides a means of verifying the digital portion of the AUDIOIN/ADC logic without causing the audio data to pass through the AUDIOOUT's FIR filter stages.
30:4	**RSRVD1**	RO	0x00	Reserved
3	**ADC_DMA_REQ_HAND_SHAKE_CLK_CROSS**	RO	0x0	DMA Request Sync Status. This bit reflects the current state of the second flop on the chain of three flip-flops used to synchronize the AUDIOIN's DMA request signal from the module's internal 24-MHz clock to the APBX's memory clock domain. This bit is only intended for test.
2	**SET_INTERRUPT3_HAND_SHAKE**	RO	0x0	Interrupt[3] Status. This bit reflects the current state of the APBX interface state machine's internal interrupt[3] signal used to prioritize channels 0 and 1 DMA requests from the DIGFILT. This bit is only intended for test.

Table 857. HW_AUDIOIN_ADCDEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**DMA_PREQ**	RO	0x0	DMA Request Status. This bit reflects the current state of the AUDIOIN's DMA request signal. DMA requests are issued any time the request signal toggles. This bit can be polled by software, in order to manually move samples from the AUDIOIN's FIFO to a memory buffer when the AUDIOIN's DMA channel is not used.
0	**FIFO_STATUS**	RO	0x0	FIFO Status. This bit is set when the AUDIOIN's FIFO contains any amount of valid data and is cleared when the FIFO is empty.

DESCRIPTION:

The AUDIOIN Debug Register provides read-only access of various internal AUDIOIN module signals to assist in debug and validation, as well as control of ADCDMA test mode.

EXAMPLE:

```
unsigned tempStatus = HW_AUDIOIN_ADCDEBUG.FIFO_STATUS;
```

25.6.6. ADC Mux Volume and Select Control Register Description

This register controls operation of the analog ADC input mux.

HW_AUDIOIN_ADCVOL 0x8004C050
HW_AUDIOIN_ADCVOL_SET 0x8004C054
HW_AUDIOIN_ADCVOL_CLR 0x8004C058
HW_AUDIOIN_ADCVOL_TOG 0x8004C05C

Table 858. HW_AUDIOIN_ADCVOL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2		SELECT_LEFT		RSRVD1		SELECT_RIGHT		RSRVD0															MUTE	GAIN_LEFT				GAIN_RIGHT			

Table 859. HW_AUDIOIN_ADCVOL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	**RSRVD2**	RO	0x0	Reserved
29:28	**SELECT_LEFT**	RW	0x0	ADC Left Channel Input Source Select. This bit field is used to select the analog input source of the ADC's left channel. 00=Microphone. 01=Line1. 10=Headphone. 11=Line2 (169-BGA only). Line2 left input is LRADC2.
27:26	**RSRVD1**	RO	0x0	Reserved

Table 859. HW_AUDIOIN_ADCVOL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:24	**SELECT_RIGHT**	RW	0x0	ADC Right Channel Input Source Select. This bit field is used to select the analog input source of the ADC's right channel. 00=Microphone. 01=Line1. 10=Headphone. 11=Line2 (169-BGA only).
23:9	**RSRVD0**	RO	0x0	Reserved
8	**MUTE**	RW	0x1	ADC Mute. When set, this bit mutes both the left and right channel analog inputs. 1=Mute. 0=Unmute. Note that mute is always applied immediately when written (unlike volume when EN_ZCD=1), therefore the user should always ramp down the channel's volume to the minimum level (-100 dB) before setting the mute bit.
7:4	**GAIN_LEFT**	RW	0x0	Left Channel ADC Gain. This bit selects the level of gain applied to the left channel analog input. Each increment of this field represents a 1.5dB gain. Programming a value of 0x0, applies a 0dB gain, 0x1 applies a 1.5dB gain, and so on up to a maximum gain of 22.5dB when a value of 0xF is used.
3:0	**GAIN_RIGHT**	RW	0x0	Right Channel ADC Gain. This bit selects the level of gain applied to the right channel analog input. Each increment of this field represents a 1.5-dB gain. Programming a value of 0x0, applies a 0-dB gain, 0x1 applies a 1.5-dB gain, and so on up to a maximum gain of 22.5 dB when a value of 0xF is used.

DESCRIPTION:

This register supplies the volume, mute, and input select controls for the analog ADC mux/gain amplifier.

EXAMPLE:

```
HW_AUDIOIN_ADCVOL.MUTE = 0;
```

25.6.7. Microphone and Line Control Register Description

This register provides the microphone and line control bits.

HW_AUDIOIN_MICLINE 0x8004C060
HW_AUDIOIN_MICLINE_SET 0x8004C064
HW_AUDIOIN_MICLINE_CLR 0x8004C068
HW_AUDIOIN_MICLINE_TOG 0x8004C06C

Table 860. HW_AUDIOIN_MICLINE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD6	ATTEN_LINE	DIVIDE_LINE1	DIVIDE_LINE2	RSRVD5			MIC_SELECT	RSRVD4		MIC_RESISTOR		RSRVD3	MIC_BIAS			RSRVD2							FORCE_MICAMP_PWRUP	RSRVD1						MIC_GAIN	

Table 861. HW_AUDIOIN_MICLINE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD6**	RO	0x0	Reserved
30	**ATTEN_LINE**	RW	0x0	Attenuate Line1 Input. This bit is used to attenuate the Line1 volume range down by >= 20 dB. Additionally, when this bit is set, the headphone volume intervals that are controlled by the VOL_LEFT and VOL_RIGHT bit fields in the HW_AUDIOOUT_HPVOL register are increased (<= 3 dB). This bit affects both the left and right channels.
29	**DIVIDE_LINE1**	RW	0x0	Attenuate Line1 Input. When used in conjunction with a 10K series input resistor on the Line1 pin, this bit causes the Line1 input signal to be attenuated by 6dB (+/-1.5 dB) t o allow a 1-Vrms input signal. This bit affects the left and right channels of both the ADC and DAC.
28	**DIVIDE_LINE2**	RW	0x0	Attenuate Line2 Input. When used in conjunction with a 10K series input resistor on the line2 pin, this bit causes the Line2 input signal to be attenuated by 6 dB (+/-1.5 dB) to allow a 1-Vrms input signal. This bit affects the left and right channels of both the ADC and DAC.
27:25	**RSRVD5**	RO	0x0	Reserved
24	**MIC_SELECT**	RW	0x0	Microphone Bias Pin Select. When MIC_RESISTOR is enabled (non-zero), this bit is used to select the pin source for the Micbias input voltage reference. 0=LRADC0. 1=LRADC1. Note that the LRADC pin that is selected for Micbias cannot also be used as an LRADC input.
23:22	**RSRVD4**	RO	0x0	Reserved

Table 861. HW_AUDIOIN_MICLINE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:20	**MIC_RESISTOR**	RW	0x0	Microphone Bias Resistor Select. Note that the analog ADC block must be powered on before turning on the Micbias circuit (ADC bit within the HW_AUDIOOUT_PWRDN register must be cleared to zero) 00=Off. 01=2.5 KOhm. 10=4.5 KOhm. 11=8.5 KOhm.
19	**RSRVD3**	RO	0x0	Reserved
18:16	**MIC_BIAS**	RW	0x0	Microphone Bias Voltage Select. 0=1.25 V, 1=1.50 V, up to 7=3.00 V (0.25-V increments)
15:9	**RSRVD2**	RO	0x00	Reserved
8	**FORCE_MICAMP_PWRUP**	RW	0x1	Force ADC Microphone Amplifier Powerup. If the ADC is powered down or is not set to the Mic for its input, then clearing this bit forces the microphone amplifier to powerup.
7:2	**RSRVD1**	RO	0x00	Reserved
1:0	**MIC_GAIN**	RW	0x0	Microphone Gain. 00=0 dB, 01=20 dB, 10=30 dB, 11=40 dB.

DESCRIPTION:

This register provides the microphone and line control bits.

EXAMPLE:

```
HW_AUDIOIN_MICLINE.MIC_GAIN = 0x2; // 30 dB
```

25.6.8. Analog Clock Control Register Description

This register provides analog clock control.

HW_AUDIOIN_ANACLKCTRL 0x8004C070
HW_AUDIOIN_ANACLKCTRL_SET 0x8004C074
HW_AUDIOIN_ANACLKCTRL_CLR 0x8004C078
HW_AUDIOIN_ANACLKCTRL_TOG 0x8004C07C

Table 862. HW_AUDIOIN_ANACLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSRVD3																								DITHER_ENABLE	SLOW_DITHER	INVERT_ADCCLK	RSRVD2	ADCDIV		

Table 863. HW_AUDIOIN_ANACLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	Analog Clock Gate. Set this bit to gate the clocks for the ADC converter and associated digital filter.
30:7	**RSRVD3**	RO	0x0	Reserved
6	**DITHER_ENABLE**	RW	0x0	ADC Dither Enable. When this bit is set, dither is enabled within the ADC.
5	**SLOW_DITHER**	RW	0x0	Slow Dither. When dither is enabled (DITHER_ENABLE=1), and this bit is set, ADC input signal dithering is slowed to half its normal rate.
4	**INVERT_ADCCLK**	RW	0x0	ADC clock invert. Set this bit to invert the ADC_CLK for the ADC sigma-delta converter and associated digital filters.
3	**RSRVD2**	RO	0x0	Reserved
2:0	**ADCDIV**	RW	0x0	ADC Analog Clock Divider. This bit field is used to select the oversampling clock rate used by the ADC. This bit field should only be changed per SigmaTel. 000=6 MHz. 001=4 MHz. 010/100=3 MHz. 011/101=2 MHz. 110=1.5 MHz. 111=1 MHz.

DESCRIPTION:

This register provides analog clock control.

EXAMPLE:

```
HW_AUDIOIN_ANACLKCTRL.ADCDIV = 0x2; // 3 MHz
```

25.6.9. AUDIOIN Read Data Register Description

The AUDIOIN Read Data Register provides access to incoming PCM audio samples.

HW_AUDIOIN_DATA 0x8004C080
HW_AUDIOIN_DATA_SET 0x8004C084
HW_AUDIOIN_DATA_CLR 0x8004C088
HW_AUDIOIN_DATA_TOG 0x8004C08C

Table 864. HW_AUDIOIN_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
HIGH																LOW															

Table 865. HW_AUDIOIN_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**HIGH**	RO	0x0000	Right Sample or Sample High Half-Word. For 16-bit sample mode, this field contains the right channel sample. For 32-bit sample mode, this field contains the most significant 16 bits of the 32-bit sample (either left or right).
15:0	**LOW**	RO	0x0000	Left Sample or Sample Low Half-Word. For 16-bit sample mode, this field contains the left channel sample. For 32-bit per sample mode, this field contains the least significant 16 bits of the 32-bit sample (either left or right).

DESCRIPTION:

The AUDIOIN Read Data Register provides 32-bit data transfers for the DMA, or, alternatively, can be directly read by the CPU. Each data value read from the register is transferred from the AUDIOIN's FIFO that contains the resultant audio data that has passed through it's digital FIR filter stages. These 32-bit values contain either one 32-bit sample or two 16-bit samples, depending on how the data size is programmed. Note that the PCM audio data output from the FIR filter stages is 24-bit. For 16-bit operation, the resultant data is normalized by dropping the least significant 8 bits. For 32-bit mode, the two's complement PCM data is sign extended to 32 bits.

EXAMPLE:

```
  unsigned long TestValue= HW_AUDIOIN_DATA.U; // read a 32 bit value from the read data reg-
ister in CPU diagnostic (non-DMA) mode
```

AUDIOIN XML Revision: 1.54

26. AUDIOOUT/DAC

This chapter describes the AUDIOOUT/DAC module implemented on the STMP36xx, including DMA, sample rate conversion, internal operation, reference control settings, and headphone amplifier operation. Programmable registers are described in Section 26.7.

26.1. Overview

The STMP36xx features an audio playback path that consists of the AUDIOOUT digital multi-stage FIR filter, followed by a sigma-delta digital-to-analog converter (DAC). PCM audio samples are transferred from a buffer in memory using the APBX bridge DMA to the AUDIOOUT's FIFO. Sample pairs are processed by a three-stage finite impulse response filter. The resultant PCM samples are input to the sigma-delta modulation (SDM) block, where they are serialized, sample rate converted to the desired output rate, and output to the analog DAC.

The analog audio destination can be selected from one of three possible outputs:

- Stereo Headphone Amplifier Output
- Stereo Speaker Amplifier Output
- Stereo Line Output

The AUDIOOUT module provides the following functions:

- 1-bit sigma-delta DAC
- Three stage upsampling FIR filter: 33-tap (1:2), 11-tap (2:4), 7-tap (4:8), supporting conversion from quarter, half, full, double and quad sample rates that are multiples of the standard 32K, 44.1K, and 48K Hz rates
- Parallel-to-serial bit-stream decimator
- Sample rate converter (SRC)
- 16- or 32-bit PCM sample widths
- APBX bridge DMA interface
- Independent control of each channel's volume (including mute)
- SigmaTel 3D virtualization
- ADC-to-DAC internal loopback for product development
- Control bit fields used for analog DAC settings

Figure 116 shows a high-level diagram of the AUDIOOUT module. See Figure 111 on page 624 for a diagram of the audio path and control options.

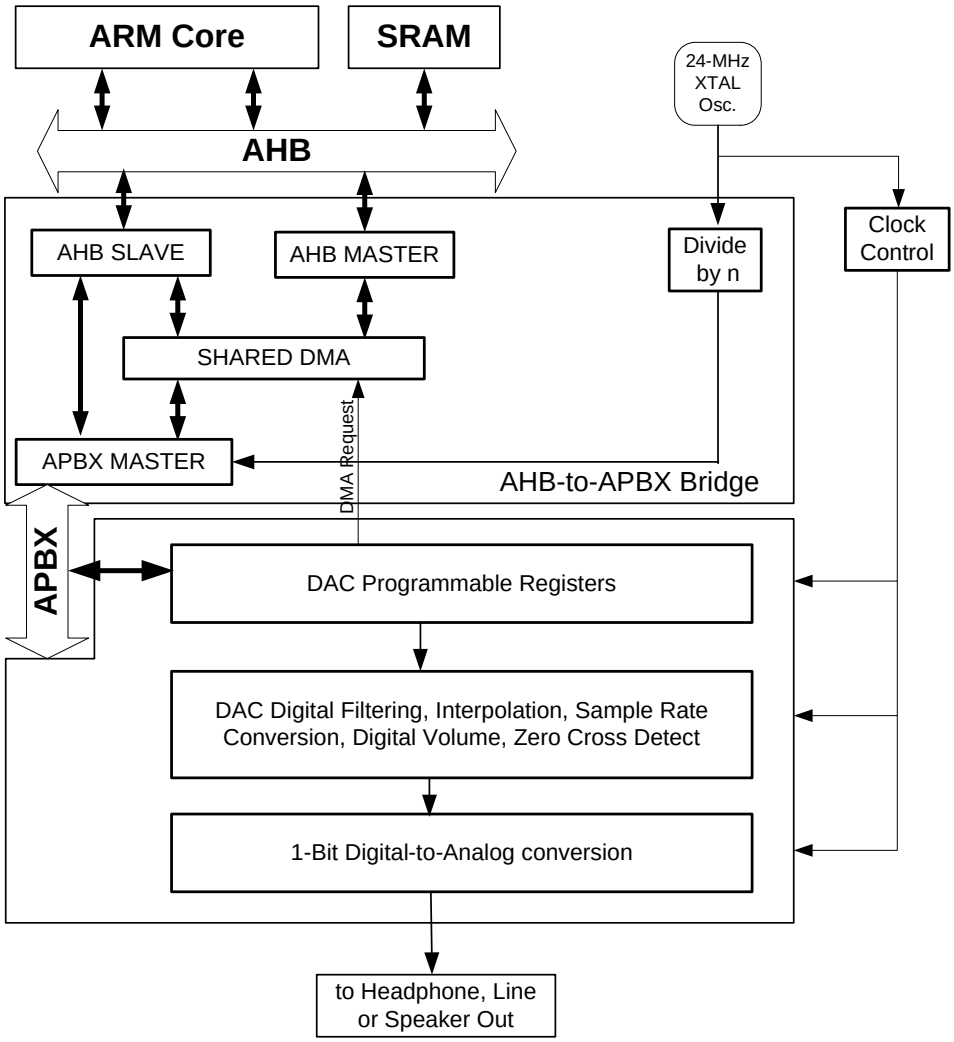


Figure 116. AUDIOOUT/DAC Block Diagram

26.2. Operation

Audio data conversion begins by either using the STMP36xx's AHB-APBX bridge DMA engine to write two's complement PCM data to the AUDIOOUT's input FIFO, or by writing the data directly to the AUDIOOUT's data register via the host CPU. The data is then normalized to 24-bit samples and then filtered using a 3-stage FIR filter, consisting of 33, 11, and 7 taps, respectively. The AUDIOOUT contains a sequencer, multiply-accumulate hardware, and a set of filter coefficients that performs successive iterations on the data stored in RAM. Intermediate data calculated along the taps/stages of the FIR are also stored in the AUDIOOUT's RAM. The AUDIOOUT module includes hardware for interpolation, sample and hold, and sigma-delta modulation that is applied to the filtered parallel PCM data. The resultant oversampled 1-bit serial stream is then output to the high-performance analog stereo sigma-delta DAC.

In most cases, the AUDIOOUT's PCM data is transferred by the AHB-APBX bridge DMA. DMA channel 1 is dedicated to the AUDIOOUT module. The DMA moves data to the AUDIOOUT's memory-mapped data register from a RAM buffer every time a request is made. The buffer may be in on-chip or off-chip RAM. It is also possible for the CPU to manually move data to the AUDIOOUT data register while monitoring either the FIFO or DMA request status bits in the AUDIOOUT debug register (HW_AUDIOOUT_DACDEBUG).

Also present on the STMP36xx is an audio record path called AUDIOIN/ADC. Although each functions independently of one another, both the AUDIOOUT and AUDIOIN blocks share their FIR filter (sequencer/RAM/coefficients) and DMA controller. This combined module is titled the "digital filter" or DIGFILT. The register descriptions that follow refer both to each path independently (AUDIOOUT and AUDIOIN) as well as a whole (DIGFILT), due to the fact that clocks and resets affect either the shared resources or the design as a whole.

In order to configure the AUDIOOUT/DAC for operation, the user must first clear the clock gate (CLKGATE) and soft reset (SFTRST) bits within the AUDIOOUT control register (HW_AUDIOOUT_CTRL). The run bit should remain off (zero), while all other control bits are initialized. It is important to note that there are also a number of control bits within the AUDIOOUT's address space that control functions within the analog DAC. Next, the bridge DMA controller channel 1 should be programmed and enabled to supply output audio samples from one or more RAM buffers. Finally, the run bit should be set to start AUDIOOUT/DAC operation. After the 8-word AUDIOOUT FIFO is filled, conversion begins.

Each 32-bit register within the AUDIOOUT's address space is aliased to four adjacent words. The first word is used for normal read-write access, while the subsequent three words are contained within the register's set-clear-toggle (SCT) address space. Only bits that are written to with a one in this space are affected. For example, writing a one to a bit using the register's set address, sets that particular bit while maintaining the state of all other bits. This convention allows easy bit manipulation without requiring the standard read-modify-write procedure. Bits that are written with a one to the register's clear address clear the bit, while the toggle address causes bits to invert their current state.

26.2.1. AUDIOOUT DMA

The DMA is typically controlled by a linked list of descriptors. Generally, the descriptors are circularly linked to cause the DMA to cycle through the set of DMA buffers. The DMA can be programmed to assert an IRQ when some or all of the buffers have been transmitted. For example, AUDIOOUT DMA descriptor 0 may program the DMA to transmit a buffer, set the done IRQ, and fetch descriptor 1. Descriptor 1 programs the DMA to transmit the next buffer. The DMA continues to operate normally while the IRQ is asserted. The CPU needs to respond to the IRQ before the DMA has transmitted all of the buffers with new data. The DMA ISR clears the IRQ flag and prepares buffers and/or descriptors with new data. In general, software copies new data into the buffers or adjust the descriptors to point to existing buffers. Software should also take advantage of the DMA's counting semaphore feature to synchronize the addition of new descriptors to the chain.

The DMA can take PCM data from any memory-mapped location. For 32-bit PCM data, the left channel sample is stored first in the lowest address, followed by the corresponding right channel sample in the next word address (+4 bytes). For 16-bit mode, sample pairs are stored in each word. Right samples are stored in the upper

half word, while left samples are stored in the lower half word. Because the AUDIOOUT always operates on stereo data, the PCM buffer should always have an integer number of words. It is not possible to play mono data unless the mono samples are each repeated twice in memory, once for the left channel and once for the right channel. The audio data values are in two's complement format, where full scale values range from 0x7FFFFFFF to 0x80000000 for 32-bit data or 0x7FFF to 0x8000 for 16-bit data.

In addition to the DMA IRQ used for AUDIOOUT buffer refill, the AUDIOOUT also has an underflow and overflow IRQ. Overflows should never occur, because (by design) the DMA should never attempt to write more data than there is space available within the AUDIOOUT's FIFO. However, if the DMA does not keep up with requests and the FIFO is emptied by the AUDIOOUT's filter stages, an underflow occurs. This causes the underflow flag to be set in the AUDIOOUT control register (HW_AUDIOOUT_CTRL). If the overflow/underflow IRQ enable bit is set, then this condition also asserts an interrupt. The interrupt is cleared by writing a one to the underflow flag in the HW_AUDIOOUT_CTRL's SCT clear address space. An AUDIOOUT underflow is typically caused by the DMA running out of new buffers, or if the AHB or APBX is stalled or is otherwise unable to meet the bandwidth requirements at the current operating frequency. If the counting semaphore reaches 0, the DMA stops processing new descriptors and stops moving data to the AUDIOOUT's data register (HW_AUDIOOUT_DATA).

In some cases, it may be desirable to synchronize the DAC clock speed with some other reference. Examples include a system playing from a network stream or digital FM receiver. In these cases, the AUDIOOUT sample rate register can be adjusted to speed up or slow down the data rate. Software needs to periodically monitor the buffer positions to make corrections as necessary.

26.3. DAC Sample Rate Converter and Internal Operation

Table 866 contains the required value of the HW_AUDIOOUT_DACSSR register for various common sample rates. Although these are the standard rates, any sample rate from 8K to 192 kHz can be programmed.

Table 866. Bit Field Values for Standard Sample Rates

SAMPLE RATE	HW_AUDIOOUT_DACSSR			
$Fsample_{DAC}$	BASEMULT	SRC_HOLD	SRC_INT	SRC_FRAC
192,000 Hz	0x4	0x0	0x0F	0x13FF
176,400 Hz	0x4	0x0	0x11	0x0037
128,000 Hz	0x4	0x0	0x17	0x0E00
96,000 Hz	0x2	0x0	0x0F	0x13FF
88,200 Hz	0x2	0x0	0x11	0x0037
64,000 Hz	0x2	0x0	0x17	0x0E00
48,000 Hz	0x1	0x0	0x0F	0x13FF
44,100 Hz	0x1	0x0	0x11	0x0037
32,000 Hz	0x1	0x0	0x17	0x0E00
24,000 Hz	0x1	0x1	0x0F	0x13FF
22,050 Hz	0x1	0x1	0x11	0x0037

Table 866. Bit Field Values for Standard Sample Rates (Continued)

16,000 Hz	0x1	0x1	0x17	0x0E00
12,000 Hz	0x1	0x3	0x0F	0x13FF
11,025 Hz	0x1	0x3	0x11	0x0037
8,000 Hz	0x1	0x3	0x17	0x0E00

NOTE: Sample-rates greater than 48 kHz can only be used when the AUDIOIN is disabled, and 44.1 kHz is the maximum sample rate at which both the AUDIOOUT and AUDIOIN can operate simultaneously.

For any of the desired sample rates, a fractional sample-rate conversion factor is calculated within the DIGFILT according to the following equation

$$SRConv_{DAC} = 65536 * [(Fanalog_{DAC})/ (8 * Hold_{DAC} * Fsample_{DAC})]$$

If computed with the above explicit operator precedence, the resulting sample-rate conversion factor ($SRConv_{DAC}$) will be a 24-bit scaled fixed-point representation of the desired decimation factor.

The 1-bit sigma delta D/A converter is always sampled on a submultiple of the 24.0-MHz crystal oscillator frequency, as specified in the HW_CCR_DACDIV register (see Figure 117). This divider generates sample strobes at $Fanalog_{DAC}$ where the divisors available come from the set {4,6,8,12,16,24}. With HW_CCR_DACDIV set to zero, to divide by 4, $Fanalog_{DAC}$ becomes 6.0 MHz for a 24.0-MHz crystal. The sample strobe derived from this divider is used to interpolate the 1-bit D/A values. The 1-bit sigma delta modulator is effectively running at $Fanalog_{DAC}$. As shown in Figure 117, the 16-bit or 32-bit D/A values are extracted from on-chip RAM via the DMA. They are filtered to band-limit the audio stream. This filter runs on xtal_clk, but filters samples at the source sample rate, which is slower than the output D/A sample rate. In the process, this filter performs a fixed 1:8 interpolation or up-sample input stream. A single 24-bit sample at the output of the fixed filter is further interpolated up to the 1-bit D/A rate. The variable rate sample, hold and interpolate block performs this function.

It stalls the filter pipeline and DMA source, using the handshake lines that connect with the previous filter stage to supply samples at the correct over-sample ratio. The 1-bit DAC runs at the fixed sample rate of $Fanalog_{DAC}$ while the DMA fetches samples in burst at irregular intervals to maintain the required input to the modulator.

In this case, the 1-bit D/A is running at 6 MHz. The sample hold and interpolate block accepts a new sample from the filter at a (44.1 kHz * 8) = 352.8 kHz. It passes interpolated samples to the modulator at a 6.0-MHz rate. The sample, hold and interpolate block passes a source sample from the fixed 1:8 interpolation filter to the sigma delta modulator corresponding to every 8.503 $Fanalog_{DAC}$ samples. Recall that this is a variable rate interpolation stage that changes for every Over Sample Rate (OSR) setting in use.

If the desired sample rate $Fsample_{DAC}$= 44.1 kHz, for example, the sample hold and interpolate block will accept samples from fixed interpolation filter at 352.8 kHz, i.e., 8x the desired sample rate. There is a handshake pair (request/ack) between the variable rate sample hold and interpolate block and the fixed interpolating filter block. This handshake is used to pace the samples from the FIFO to 44.1 kHz.

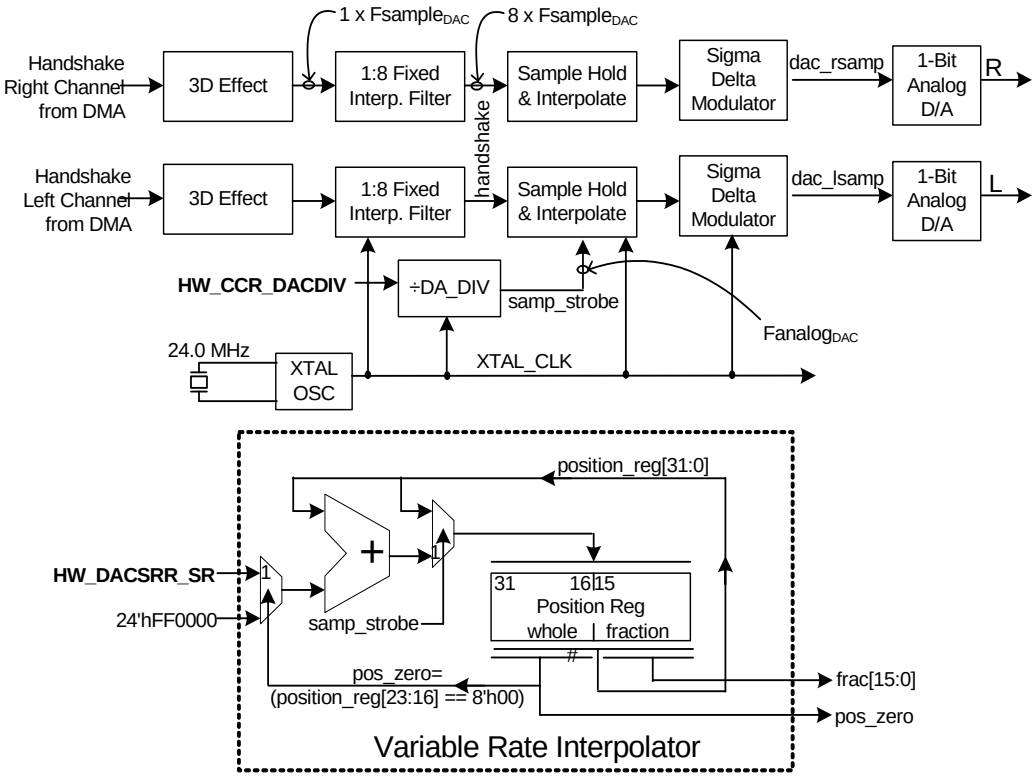


Figure 117. Stereo Sigma Delta D/A Converter

There are also members of the 48-kHz family whose members satisfy the property:

24.576 MHz = Q*Fsample$_{DAC}$

These sample rates include 48 kHz, 32 kHz, 24 kHz, 16 kHz, and 12 kHz.

There are also the members of the 44.1-kHz family whose members satisfy the property:

16.9344 MHz = Q*Fsample$_{ADC}$

where Q comes from the set of integers. These sample rates include 44.1 kHz, 22.05 kHz, and 11.025 kHz.

The D/A converter block includes a variable rate or rational interpolator to accommodate these sample rates, as shown in Figure 117. Rational numbers in the DAC are approximated with a scaled fixed-point 24-bit value. In this case, the decimal point falls between bit 15 and bit 16. Therefore, the lower two bytes hold the fractional part, while the upper byte holds the whole number portion of the scaled fixed point. The position register uses this scaled fixed-point representation for the number of 1-bit samples to be interpolated (generated) to find the next sample to be sent to the sigma delta modulator. Whenever the whole number part (bits 23:16) is zero, then the next DMA sample is consumed. For playback at 44.1 kHz, set the interpolator to generate 67.027 new interpolated samples between every DMA sample.

26.4. Reference Control Settings

The reference control register allows adjustment of reference voltages and currents. VBG is the internal bandgap voltage (no external pin). This is a stable voltage reference used to establish *all* other voltage and current references for the chip, including VAG, vrefp, the Li-Ion charge termination voltage, etc. All the voltage and current references on the chip are proportional to VBG.

VAG is the analog ground voltage. It sets the DC bias on the input and output of all of the audio pins. This is typically set near VDDA/2. VAG also affects the peak output swing of the DAC. As VAG is lowered, the output swing of the DAC scales with it. However, at low VDDA levels, the analog performance can be improved by setting VAG somewhat below VDDA/2. The DAC is particularly susceptible to power supply noise if VDDA-VAG is not large enough. Table 867 shows the simulated PSRR at different VAG settings when VDDA=1.35V. The table includes typical and worst case results.

Table 867. Simulated PSRR for DAC at 1.35 VDDA

VAG CODE	VAG VALUE	TT 25C	SS –20C
1010	0.684	–62	
1001	0.664	–77	
1000	0.645	–85	
0111	0.625	–85	–36
0110	0.605		–53
0101	0.586		–60

26.5. Headphone

The STM35xx supports a conventional stereo headphone drive, as shown in Figure 118.

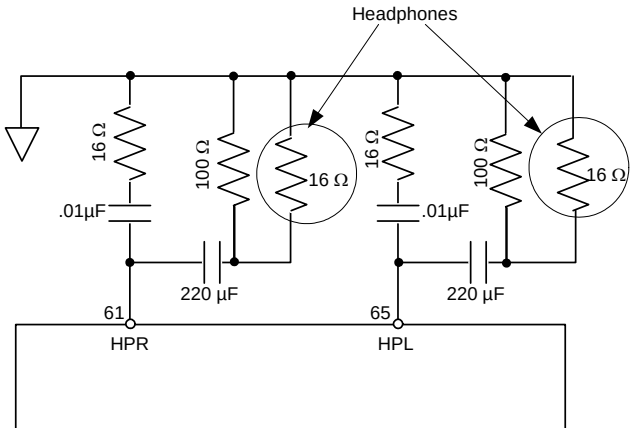


Figure 118. Conventional Stereo Headphone Application Circuit

In addition, as shown in Figure 119–Figure 121, the chip can generate an optional headphone common node circuit for the headphones that eliminates the need for the large and expensive DC blocking capacitors. It also improves the anti-pop performance. These benefits are obtained at a slight increase in power consumption, i.e., at 30 mV rms output, the resultant increase in power consumption is approximately 2.7 mW.

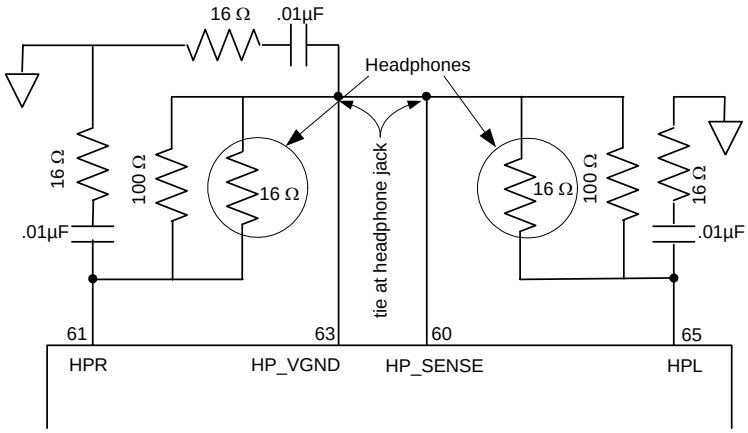


Figure 119. Stereo Headphone Application Circuit with Common Node

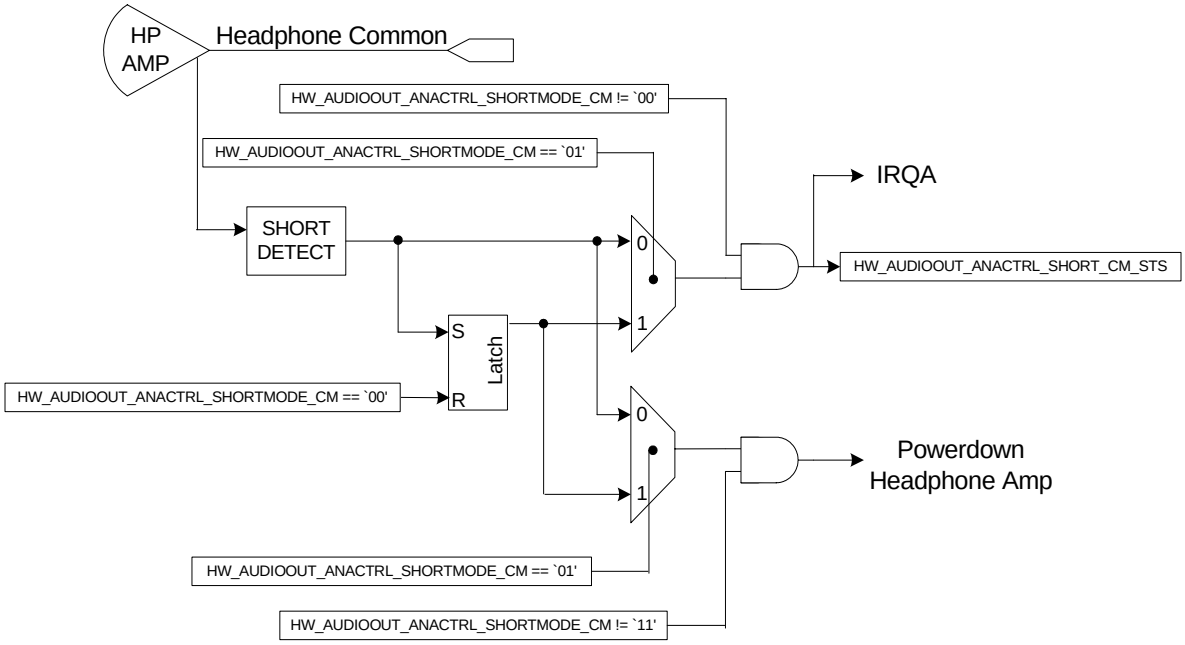


Figure 120. Stereo Headphone Common Short Detection and Powerdown Circuit

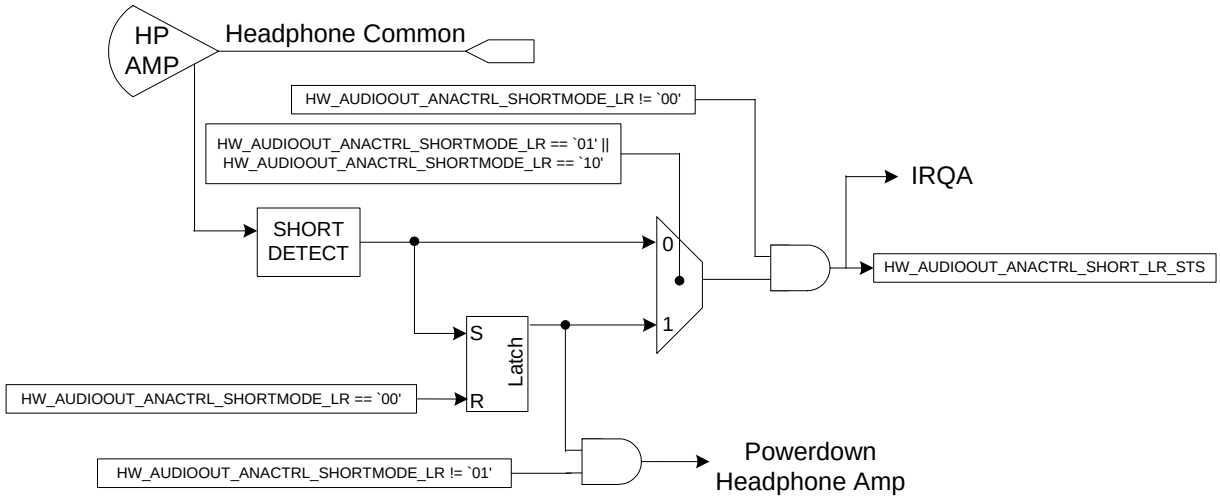


Figure 121. Stereo Headphone L/R/ Short Detection and Powerdown Circuit

26.5.1. Board Components

The headphone amplifier output requires a few external resistors and capacitors. There is a series RC compensation network that helps to guarantee the stability of the amplifier under all loading conditions. There is also a 100Ω resistor to the headphone ground, which has multiple functions:

- In cap mode operation (a DC blocking cap is present), the 100Ω resistor improves startup pop suppression.
- In capless mode operation (no DC blocking cap), the resistor avoids 60-Hz hum into a powered speaker when the player is turned off.
- In capless and cap mode players, the 100Ω resistor minimizes the power-off signal feedthrough from line-in to headphone out.
- In powerdown mode, there is a 100kΩ resistor between line-in and headphone out. The 100Ω load resistor keeps the powerdown feedthrough level at –60 dB. When the part is powered up, there is no signal path between line-in and headphone out, thus no bleedthrough.

26.5.2. Capless Mode Operation

The headphone amplifier is designed to work with or without a DC blocking cap. In capless mode, an amplifier is used to bias the headphone ground at the analog ground level (VAG), which is typically near VDDA/2. This avoids any DC signal across the output load.

Capless operation provides slight improvement to PSRR and low frequency performance. It slightly degrades SNR and THD (approximately 1 dB). The biggest advantage is the savings in cost and area by eliminating the large DC blocking caps. The biggest disadvantage is extra power consumption.

The capless mode of operation doubles the power consumed in the headphone amps. At normal listening levels, this has a fairly small effect on battery life. But with a full scale sine wave, it can significantly reduce the battery life.

With a sinusoidal signal, the headphone power consumption per channel with a 16Ω load is roughly:

Power=1mW + Vpeak*VDDA/ (16ohm * PI) (per channel)

This number is doubled in capless mode.

However, normal music files have a much higher peak-to-average ratio than a sinusoid. A PAR of 10 is typical in a music file, compared to a PAR of 1.414 for a sinusoid. So headphone power for a normal music file is:

Power=1mW + Vpeak*VDDA*2.8mW (per channel)

Again, this number is doubled in capless mode.

26.6. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

26.7. Programmable Registers

The following registers provide control for programmable elements of the AUDIOOUT/DAC block.

26.7.1. AUDIOOUT Control Register Description

The AUDIOOUT Control Register provides overall control of the digital portion of the digital-to-analog converter.

HW_AUDIOOUT_CTRL 0x80048000
HW_AUDIOOUT_CTRL_SET 0x80048004
HW_AUDIOOUT_CTRL_CLR 0x80048008
HW_AUDIOOUT_CTRL_TOG 0x8004800C

Table 868. HW_AUDIOOUT_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSRVD4										DMAWAIT_COUNT				RSRVD3	LR_SWAP	EDGE_SYNC	INVERT_1BIT	RSRVD2		SS3D_EFFECT		RSRVD1	WORD_LENGTH	DAC_ZERO_ENABLE	LOOPBACK	FIFO_UNDERFLOW_IRQ	FIFO_OVERFLOW_IRQ	FIFO_ERROR_IRQ_EN	RUN

Table 869. HW_AUDIOOUT_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	AUDIOOUT Module Soft Reset. Setting this bit to one forces a reset to portions of DIGFILT that control audio output and then gates the clocks off since the CLKGATE bit's reset state is to disable clocks. This bit must be cleared to zero for normal operation. Note that the CLKGATE bit does not affect SFTRST since it must remain writeable during clock gating. A note is included in the bit field descriptions below for those bits that are not affected by the AUDIOOUT's soft reset bit. These bits either control AUDIOIN or both AUDIOIN and AUDIOOUT functions.
30	**CLKGATE**	RW	0x1	AUDIOOUT Clock Gate Enable. When this bit is set to 1, it gates off the clocks to the portions of the DIGFILT block that control only output audio functions. It does not affect portions of the block that control AUDIOIN. Clear this bit to zero for normal AUDIOOUT operation. Note that when this bit is set, it remains writeable during clock gating so that it may be disabled by the user.
29:21	**RSRVD4**	RO	0x0	Reserved. Always write zeroes to this bit field.

Table 869. HW_AUDIOOUT_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20:16	**DMAWAIT_COUNT**	RW	0x0	DMA Request Delay Count. This bit field specifies the number of APBX clock cycles (0 to 31) to delay before each DMA request. This field acts as a throttle on the bandwidth consumed by the DIGFILT block. This field can be loaded by the DMA.
15	**RSRVD3**	RO	0x0	Reserved. Always write zeroes to this bit field.
14	**LR_SWAP**	RW	0x0	Left/Right Output Channel Swap Enable. Setting this bit to one swaps the left and right serial audio outputs from the SDM block to the analog DAC .
13	**EDGE_SYNC**	RW	0x0	Serial Output Clock Edge Sync Select. This bit selects the edge of the DAC's serial output clock upon which the SDM synchronizes for data transmit. 0=Rising edge. 1=Falling edge.
12	**INVERT_1BIT**	RW	0x0	Invert Serial Audio Output Enable. When set, this bit inverts the 1-bit serial output of both the left and right channels to the DAC's sigma-delta modulator. 0=Normal operation. 1=Invert L/R serial audio output from the SDM block.
11:10	**RSRVD2**	RO	0x0	Reserved. Always write zeroes to this bit field.
9:8	**SS3D_EFFECT**	RW	0x0	Virtual 3D Effect Enable. This bit field provides a virtual 3D effect for a two channel system by subtracting a portion of the opposite channel's content for each channel. Three reduction ratios are available (dB value represents amount of opposite channel content subtracted). 00=Off 01=Low (3 dB) 10=Medium (4.5 dB) 11=High (6 dB)
7	**RSRVD1**	RO	0x0	Reserved. Always write zeroes to this bit field.
6	**WORD_LENGTH**	RW	0x0	PCM Audio Bit Size Select. This bit selects the size of the parallel PCM data that is input to the AUDIOOUT's FIFO. 0=32-bit PCM samples. 1=16-bit samples. Note that the PCM audio data input to the FIR filter stages is 24 bit. For 16-bit operation, the data is first sign-extended to 24 bits. For 32-bit operation, the data is first normalized by dropping the least significant 8 bits.
5	**DAC_ZERO_ENABLE**	RW	0x0	Quiet Output Enable. When enabled, this bit causes the AUDIOOUT's SDM block to transmit alternating ones and zeros (...010101...). This function is used for pop-suppression while the AUDIOOUT is reconfigured and during periods when the modulator would othewise transmit zeros. Note that this function continues to operate when AUDIOOUT is clock-gated. Also note that the user must enable/disable this function while the AUDIOOUT is not clock-gated. This bit is reset by a power-on reset only.

Table 869. HW_AUDIOOUT_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	**LOOPBACK**	RW	0x0	AUDIOIN-to-AUDIOOUT Loopback Enable. Setting this bit to one routes the audio data received by the AUDIOIN's FIFO to the AUDIOOUT's FIFO. This test mode provides a loopback that does not use the DIGFILT's DMA memory inteface. This bit should be cleared to zero for normal operation.
3	**FIFO_UNDERFLOW_IRQ**	RW	0x0	FIFO Underflow Interrupt Status Bit. This bit is set by hardware if the AUDIOOUT's FIFO underflows any time during operation due to a DMA request that is not serviced in time. It is reset by software by writing a one to the corresponding bit in the HW_AUDIOOUT_CTRL_CLR register. An interrupt is issued to the host processor if this bit is set and FIFO_ERROR_IRQ_EN=1.
2	**FIFO_OVERFLOW_IRQ**	RW	0x0	FIFO Overflow Interrupt Status Bit. This bit is set by hardware if the AUDIOOUT's FIFO overflows. It is reset by software by writing a one to the corresponding bit in the HW_AUDIOOUT_CTRL_CLR register. An interrupt is issued to the host processor if this bit is set, and FIFO_ERROR_IRQ_EN=1. Note that overflows should not occur by design since requests to the DMA are not made unless there is adequate space present within the FIFO. Therefore, this condition would indicate a serious DMA error.
1	**FIFO_ERROR_IRQ_EN**	RW	0x0	FIFO Error Interrupt Enable. Set this bit to one to enable an AUDIOOUT interrupt request to the host processor when either the FIFO overflow or underflow status bits are set. Note that this bit does not affect the state of the underflow/overflow status bits, but rather their ability to signal an interrupt to the CPU.
0	**RUN**	RW	0x0	AUDIOOUT Enable. Setting this bit to one causes AUDIOOUT operation to start, beginning with a DMA request to fill its 8-word FIFO. Clearing this bit to zero stops data conversion and also causes the CLKGATE bit to be set.

DESCRIPTION:

The AUDIOOUT Control Register contains bit fields used to control and monitor AUDIOOUT operation including: reset, clocks, DMA transfers, data to the analog DAC module, PCM data size, test, and interrupt control.

EXAMPLE:

```
HW_AUDIOOUT_CTRL.RUN = 1; // start DAC conversion
```

26.7.2. AUDIOOUT Status Register Description

The AUDIOOUT Status Register is used to determine if the digital-to-analog converter is operational.

HW_AUDIOOUT_STAT 0x80048010
HW_AUDIOOUT_STAT_SET 0x80048014
HW_AUDIOOUT_STAT_CLR 0x80048018

HW_AUDIOOUT_STAT_TOG 0x8004801C

Table 870. HW_AUDIOOUT_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DAC_PRESENT	RSRVD1																														

Table 871. HW_AUDIOOUT_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**DAC_PRESENT**	RO	0x1	AUDIOOUT Functionality Present. This status bit is set to one in products that include the AUDIOOUT/DAC. If this bit is zero, the AUDIOOUT/DAC is permanently disabled and cannot be operated by the user.
30:0	**RSRVD1**	RO	0x0	Reserved. Always write zeroes to this bit field.

DESCRIPTION:

The AUDIOOUT Status Register provides an indication of the presence of the DAC functionality.

EXAMPLE:

```
unsigned statusValue = HW_AUDIOOUT_STAT.DAC_PRESENT;
```

26.7.3. AUDIOOUT Sample Rate Register Description

The AUDIOOUT Sample Rate Register is used to specify the sample rate that the parallel PCM audio data is converted to within the SDM module before being output to the analog DAC.

HW_AUDIOOUT_DACSRR 0x80048020
HW_AUDIOOUT_DACSRR_SET 0x80048024
HW_AUDIOOUT_DACSRR_CLR 0x80048028
HW_AUDIOOUT_DACSRR_TOG 0x8004802C

Table 872. HW_AUDIOOUT_DACSRR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
OSR	BASEMULT			RSRVD2	SRC_HOLD			RSRVD1				SRC_INT				RSRVD0			SRC_FRAC												

Table 873. HW_AUDIOOUT_DACSRR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**OSR**	RO	0x0	AUDIOOUT Oversample Rate. Note that for the STMP36xx, the oversample rate is fixed at 6 MHz. OSR6 = 0x0 AUDIOOUT oversample rate at 6 MHz. OSR12 = 0x1 AUDIOOUT oversample rate at 12 MHz.
30:28	**BASEMULT**	RW	0x1	Base Sample Rate Multiplier. This bit field is used to configure the DAC's sample rate to one of three ranges: single, double, or quad. This multiply factor is used to achieve sample rates greater than the standard rates of 32/44.1/48 kHz. A value of 0x1 should be used when selecting half and quarter sample rates. Note that sample rates greater than 48 kHz may only be used when the AUDIOIN is disabled, and 44.1 kHz is the maximum sample rate at which both the AUDIOIN and AUDIOOUT can operate simultaneously. SINGLE_RATE = 0x1 Single rate multiplier (for 48/44.1/32 kHz as well as half and quarter rates). DOUBLE_RATE = 0x2 Double rate multiplier (for 96/88.2/64 kHz). QUAD_RATE = 0x4 Quad rate multiplier (for 192/176.4/128 kHz).
27	**RSRVD2**	RO	0x0	Reserved. Always write a zero to this bit field.
26:24	**SRC_HOLD**	RW	0x0	Sample Rate Conversion Hold Factor. This bit is used to hold a sample of a variable number of clock cycles in order to generate half and quarter sample rates when dividing down the AUDIOOUT's internal rate using the equation: output_sample_rate = (6x10^6 * BASEMULT) / (SRC_INT.SRC_FRAC * 8 * (SRC_HOLD + 1)). Refer to the sample rate table earlier in this chapter that provides a list of bit field values required to achieve all common sample rates.
23:21	**RSRVD1**	RO	0x0	Reserved. Always write zeros to this bit field.
20:16	**SRC_INT**	RW	0x11	Sample Rate Conversion Integer Factor. This bit field is the integer portion of a divide term used to sample rate convert the AUDIOOUT's internal rate using the equation: output_sample_rate = (6x10^6 * BASEMULT) / (SRC_INT.SRC_FRAC * 8 * (SRC_HOLD + 1)). Refer to the sample rate table earlier in this chapter that provides a list of bit field values required to achieve all common sample rates.
15:13	**RSRVD0**	RO	0x0	Reserved. Always write zeros to this bit field.
12:0	**SRC_FRAC**	RW	0x37	Sample Rate Conversion Fraction Factor. This bit field is the fractional portion of a divide term used to sample rate convert the AUDIOOUT's internal rate using the equation: output_sample_rate = (6x10^6 * BASEMULT) / (SRC_INT.SRC_FRAC * 8 * (SRC_HOLD + 1)). Refer to the sample rate table earlier in this chapter that provides a list of bit field values required to achieve all common sample rates.

DESCRIPTION:

The AUDIOOUT Sample Rate Register provides a number of bit fields to direct the SDM module's hardware to sample-rate-convert the audio stream output to one of a

number of common sample rates. Note that these values can also be dynamically altered in small amounts in order to track variations in the outgoing audio stream from sources such as FM digital radio.

EXAMPLE:

```
// Program the DAC to output a sample rate of 48 kHz:
HW_AUDIOOUT_DACSRR.BASEMULT = 0x1; // quad-rate
HW_AUDIOOUT_DACSRR.SRC_HOLD = 0x0; // 0 for full- double- quad-rates
HW_AUDIOOUT_DACSRR.SRC_INT = 0xF; // 15 for the integer portion
HW_AUDIOOUT_DACSRR.SRC_FRAC = 0x13FF; // the fractional portion
```

26.7.4. AUDIOOUT Volume Register Description

The AUDIOOUT Volume Register is used to adjust the signal level of the playback audio output to the DAC.

HW_AUDIOOUT_DACVOLUME 0x80048030
HW_AUDIOOUT_DACVOLUME_SET 0x80048034
HW_AUDIOOUT_DACVOLUME_CLR 0x80048038
HW_AUDIOOUT_DACVOLUME_TOG 0x8004803C

Table 874. HW_AUDIOOUT_DACVOLUME

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4			VOLUME_UPDATE_LEFT	RSRVD3		EN_ZCD	MUTE_LEFT	VOLUME_LEFT								RSRVD2			VOLUME_UPDATE_RIGHT	RSRVD1			MUTE_RIGHT	VOLUME_RIGHT							

Table 875. HW_AUDIOOUT_DACVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSRVD4**	RO	0x00	Reserved. Always write zeroes to this bit field.
28	**VOLUME_UPDATE_LEFT**	RO	0x0	Left Channel Volume Update Pending. This bit is set to one by the hardware when an AUDIOOUT volume update is pending, i.e., waiting on a zero crossing on the left channel. The bit is set following a write to the VOLUME_LEFT bit field and is cleared when the attenuation value is applied to the PCM output stream (at a zero-crossing). This status bit is not used when EN_ZCD=0.
27:26	**RSRVD3**	RO	0x00	Reserved. Always write zeroes to this bit field.

Table 875. HW_AUDIOOUT_DACVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25	**EN_ZCD**	RW	0x0	Enable Zero Cross Detect. This bit enables/disables use of the zero cross detect circuit in the DAC (rather than enabling the circuit itself). When enabled, changes to the volume bit fields are not applied until it is detected that the output signal's sign bit toggles (crosses zero amplitude). When disabled, changes to the volume bit fields take effect immediately when written.
24	**MUTE_LEFT**	RW	0x1	Mute Left Channel. 0=unmute, 1=mute. Note that mute is always applied immediately when written (unlike volume when EN_ZCD=1). The channel volume should always be ramped down to the minimum level (-100dB) before setting the mute bit.
23:16	**VOLUME_LEFT**	RW	0xff	Left Channel Volume Setting. This bit field is used to establish the outgoing PCM audio signal strength during playback. Volume ranges from full scale -0.5dB (0xFE) to -100dB (0x37). Each increment of this bit field causes a half-dB increase in volume. Note that values 0x00-0x37 all produce the same attenuation level of -100dB, and a value of 0xFF is reserved. Also note that the several bit fields exist for the analog DAC that should be used to adjust the realitive gain of the output signal from the AUDIOOUT block.
15:13	**RSRVD2**	RO	0x00	Reserved. Always write zeroes to this bit field.
12	**VOLUME_UPDATE_RIGHT**	RO	0x0	Right Channel Volume Update Pending. This bit is set to one by the hardware when an AUDIOOUT volume update is pending, i.e., waiting on a zero crossing on the right channel. The bit is set following a write to the VOLUME_RIGHT bit field and is cleared when the attenuation value is applied to the PCM output stream (at a zero-crossing). This status bit is not used when EN_ZCD=0.
11:9	**RSRVD1**	RO	0x00	Reserved. Always write zeroes to this bit field.
8	**MUTE_RIGHT**	RW	0x1	Mute Right Channel. 0=Unmute, 1=Mute. Note that mute is always applied immediately when written (unlike volume when EN_ZCD=1), therefore the user should always ramp down the channel's volume to the minimum level (-100 dB) before setting the mute bit.
7:0	**VOLUME_RIGHT**	RW	0xff	Right Channel Volume Setting. This bit field is used to establish the outgoing PCM audio signal strength during playback. Volume ranges from full scale -0.5dB (0xFE) to -100dB (0x37). Each increment of this bit field causes a half-dB increase in volume. Note that values 0x00-0x37 all produce the same attenuation level of -100dB, and a value of 0xFF is reserved. Also note that the several bit fields exist for the analog DAC that should be used to adjust the realitive gain of the output signal from the AUDIOOUT block.

DESCRIPTION:

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

The AUDIOOUT Volume Register allows independent volume and mute control of the left and right channels. Output audio can be attenuated in 0.5-dB steps, from full-scale down to a minimum of -100 dB. This register is also used to enable/control volume updates such that they are only applied when PCM values cross zero to prevent unwanted audio artifacts.

EXAMPLE:

```
HW_AUDIOOUT_DACVOLUME.U = 0x01ff01ff; // mute both left and right channels
HW_AUDIOOUT_DACVOLUME.U = 0x00ff00ff; // maximum volume for left and right channels.
```

26.7.5. AUDIOOUT Debug Register Description

The AUDIOOUT Debug Register is used for test and debug of the AUDIOOUT block.

HW_AUDIOOUT_DACDEBUG 0x80048040
HW_AUDIOOUT_DACDEBUG_SET 0x80048044
HW_AUDIOOUT_DACDEBUG_CLR 0x80048048
HW_AUDIOOUT_DACDEBUG_TOG 0x8004804C

Table 876. HW_AUDIOOUT_DACDEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
ENABLE_DACDMA	RSRVD1																									SET_INTERRUPT1_CLK_CROSS	SET_INTERRUPT0_CLK_CROSS	SET_INTERRUPT1_HAND_SHAKE	SET_INTERRUPT0_HAND_SHAKE	DMA_PREQ	FIFO_STATUS

Table 877. HW_AUDIOOUT_DACDEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**ENABLE_DACDMA**	RW	0x0	AUDIOOUT Digital Path Test Enable. This bit is used solely for development and debugging, and is not functional on production parts. When enabled, it causes the AUDIOIOUT's serial audio data output to bypass the DAC analog block, to be assembled into 32-bit words and transferred out to memory using the AUDIOIN's DMA Channel 0. Unlike loopback, this test mode provides a means of verifying the digital portion of the AUDIOOOUT logic without causing the audio data to pass through the AUDIOIN's FIR filter stages.
30:6	**RSRVD1**	RO	0x00	Reserved. Always write zeroes to this bit field.

Table 877. HW_AUDIOOUT_DACDEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
5	**SET_INTERRUPT1_CLK_CROSS**	RO	0x0	Interrupt[1] Sync Status. This bit reflects the current state of the second flop on the chain of three flip-flops used to synchronize the AUDIOOUT's interrupt[1] signal used to prioritize channels 0 and 1 DMA requests from the DIGFILT. This signal is synchronized from the module's internal 24-MHz clock to the APBX's memory clock domain. This bit is intended for test only.
4	**SET_INTERRUPT0_CLK_CROSS**	RO	0x0	Interrupt[0] Sync Status. This bit reflects the current state of the second flop on the chain of three flip-flops used to synchronize the AUDIOOUT's interrupt[0] signal used to prioritize channel 0 and 1 DMA requests from the DIGFILT. This signal is synchronized from the module's internal 24-MHz clock to the APBX's memory clock domain. This bit is intended for test only.
3	**SET_INTERRUPT1_HAND_SHAKE**	RO	0x0	Interrupt[1] Status. This bit reflects the current state of the APBX interface state machine's internal interrupt[1] signal used to prioritize channel 0 and 1 DMA requests from the DIGFILT. This bit is intended for test only.
2	**SET_INTERRUPT0_HAND_SHAKE**	RO	0x0	Interrupt[0] Status. This bit reflects the current state of the APBX interface state machine's internal interrupt[0] signal used to prioritize channel 0 and 1 DMA requests from the DIGFILT. This bit is intended for test only.
1	**DMA_PREQ**	RO	0x0	DMA Request Status. This bit reflects the current state of the AUDIOOUT's DMA request signal. DMA requests are issued any time the request signal toggles. This bit can be polled by software, in order to manually move samples to the AUDIOOUT's FIFO from a memory buffer when the AUDIOOUT's DMA channel is not used.
0	**FIFO_STATUS**	RO	0x1	FIFO Status. This bit is set by hardware when the AUDIOOUT's FIFO contains any empty entries and is cleared when the FIFO is full.

DESCRIPTION:

The AUDIOOUT Debug Register provides read-only access of various internal AUDIOOUT module signals to assist in debug and validation, as well as control of DACDMA test mode.

EXAMPLE:

```
unsigned tempStatus = HW_AUDIOOUT_DACDEBUG.FIFO_STATUS;
```

26.7.6. Headphone Volume and Select Control Register Description

The Headphone Volume and Select Control Register provides volume, mute, and input select controls for the headphone.

HW_AUDIOOUT_HPVOL 0x80048050
HW_AUDIOOUT_HPVOL_SET 0x80048054
HW_AUDIOOUT_HPVOL_CLR 0x80048058
HW_AUDIOOUT_HPVOL_TOG 0x8004805C

Table 878. HW_AUDIOOUT_HPVOL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4						SELECT		RSRVD3							MUTE	RSRVD2			VOL_LEFT					RSRVD1				VOL_RIGHT			

Table 879. HW_AUDIOOUT_HPVOL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD4**	RO	0x00	Reserved. Always write zeroes to this bit field.
25:24	**SELECT**	RW	0x0	Master Output Select. 00=Stereo DAC. 01=Stereo Line1. 10=Mono DAC (from Left DAC). 11=Invalid. It is reset by a power-on reset only.
23:17	**RSRVD3**	RO	0x00	Reserved. Always write zeroes to this bit field.
16	**MUTE**	RW	0x1	Headphone Mute. It is reset by a power-on reset only.
15:13	**RSRVD2**	RO	0x00	Reserved. Always write zeroes to this bit field.
12:8	**VOL_LEFT**	RW	0x03	Left Headphone Gain Control. This bit field controls headphone gain. Each decrement represents a 2-dB step. +6 dB max volume in DAC mode. +12 dB max volume in LineIn mode. It is reset by a power-on reset only.
7:5	**RSRVD1**	RO	0x00	Reserved. Always write zeroes to this bit field.
4:0	**VOL_RIGHT**	RW	0x03	Headphone Right Volume Control. 2-dB volume steps. +6 dB max volume in DAC mode. +12 dB max volume in LineIn mode. It is reset by a power-on reset only.

DESCRIPTION:

This register provides volume, mute, and input select controls for the headphone.

EXAMPLE:

```
HW_AUDIOOUT_HPVOL.MUTE = 0;
```

26.7.7. Speaker Volume Control Register Description

This register provides volume and mute controls for the speaker.

HW_AUDIOOUT_SPKRVOL 0x80048060
HW_AUDIOOUT_SPKRVOL_SET 0x80048064
HW_AUDIOOUT_SPKRVOL_CLR 0x80048068
HW_AUDIOOUT_SPKRVOL_TOG 0x8004806C

Table 880. HW_AUDIOOUT_SPKRVOL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2															MUTE	RSRVD1												VOL			

Table 881. HW_AUDIOOUT_SPKRVOL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:17	**RSRVD2**	RO	0x00	Reserved. Always write zeroes to this bit field.
16	**MUTE**	RW	0x1	Speaker Mute. It is reset by a power-on reset only.
15:4	**RSRVD1**	RO	0x00	Reserved. Always write zeroes to this bit field.
3:0	**VOL**	RW	0x3	Speaker Volume Control. 2-dB volume steps. +12 dB max volume. Default = +6 dB. Speaker input is always Right DAC. It is reset by a power-on reset only.

DESCRIPTION:

This register provides the volume and mute controls for the speaker.

EXAMPLE:

```
HW_AUDIOOUT_SPKRVOL.MUTE = 0;
```

26.7.8. Audio Power-Down Control Register Description

The Audio Power-Down Control Register provides all power-down control bits.

HW_AUDIOOUT_PWRDN 0x80048070
HW_AUDIOOUT_PWRDN_SET 0x80048074
HW_AUDIOOUT_PWRDN_CLR 0x80048078
HW_AUDIOOUT_PWRDN_TOG 0x8004807C

Table 882. HW_AUDIOOUT_PWRDN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD7							SPEAKER	RSRVD6			SELFBIAS	RSRVD5			RIGHT_ADC	RSRVD4			DAC	RSRVD3			ADC	RSRVD2			CAPLESS	RSRVD1			HEADPHONE

Table 883. HW_AUDIOOUT_PWRDN Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSRVD7**	RO	0x00	Reserved. Always write zeroes to this bit field.
24	**SPEAKER**	RW	0x1	Speaker Power-Down. It is reset by a power-on reset only.
23:21	**RSRVD6**	RO	0x00	Reserved. Always write zeroes to this bit field.

Table 883. HW_AUDIOOUT_PWRDN Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	**SELFBIAS**	RW	0x0	Powers Down the Self-Bias Circuit. The reference uses a self-bias circuit during power-up that can be turned off with this bit. It is reset by a power-on reset only.
19:17	**RSRVD5**	RO	0x00	Reserved. Always write zeroes to this bit field.
16	**RIGHT_ADC**	RW	0x0	Right ADC Power Down. When enabled, powers down the ADC's right channel while allowing the left to function normally (mono). Note that, although this bit is located in the DAC address space, it is an ADC function and is reset by the ADC SFTRST bit.
15:13	**RSRVD4**	RO	0x00	Reserved. Always write zeroes to this bit field.
12	**DAC**	RW	0x1	Power Down DAC Analog Circuitry. It is reset by a power-on reset only.
11:9	**RSRVD3**	RO	0x00	Reserved. Always write zeroes to this bit field.
8	**ADC**	RW	0x1	Power Down ADC and Input Mux Circuitry. Note that, although this bit is located in the DAC address space, it is an ADC function and is reset by the ADC SFTRST bit.
7:5	**RSRVD2**	RO	0x00	Reserved. Always write zeroes to this bit field.
4	**CAPLESS**	RW	0x1	Power Down Headphone Common Amplifier Used in Capless Headphone. If this bit is high, then the capless circuit is powered down and the device is either OFF or operating in CAP MODE (AC-coupled). If the bit it low, then the device is ON AND in CAPLESS MODE (DC-coupled). This bit field is reset by a power-on reset only.
3:1	**RSRVD1**	RO	0x00	Reserved. Always write zeroes to this bit field.
0	**HEADPHONE**	RW	0x1	Master (Headphone) Power Down. It is reset by a power-on reset only.

DESCRIPTION:

The Audio Power-Down Register provides control to power-down sections of the audio analog circuit.

EXAMPLE:

```
HW_AUDIOOUT_PWRDN.DAC = 0;
```

26.7.9. AUDIOOUT Reference Control Register Description

This register provides the voltage and current reference control bits.

HW_AUDIOOUT_REFCTRL 0x80048080
HW_AUDIOOUT_REFCTRL_SET 0x80048084
HW_AUDIOOUT_REFCTRL_CLR 0x80048088
HW_AUDIOOUT_REFCTRL_TOG 0x8004808C

Table 884. HW_AUDIOOUT_REFCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4							XTAL_BGR_BIAS	RSRVD3	VBG_ADJ			LOW_PWR	LW_REF	BIAS_CTRL		RSRVD2		ADJ_ADC	ADJ_VAG	ADC_REFVAL				VAG_VAL				RSRVD1	DAC_ADJ		

Table 885. HW_AUDIOOUT_REFCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSRVD4**	RO	0x00	Reserved. Always write zeroes to this bit field.
24	**XTAL_BGR_BIAS**	RW	0x0	Switch the XTAL bias from self-bias to bandgap-based bias current. Also switches the source of the XTAL supply in series AA or LiIon to core supply to save power. Note that while this bit is located in the DAC address space, since it controls both DAC and ADC functions, it is not reset by the DAC's SFTRST bit. It is reset by a power-on reset only.
23	**RSRVD3**	RO	0x0	Reserved. Always write zeroes to this bit field.
22:20	**VBG_ADJ**	RW	0x0	Small adjustment for VBG value. Will affect ALL reference voltages. Expected to be used to tweak final Li-Ion charge voltage. 000=Nominal. 001=+0.25%. 010=+0.5%. 011=0.75%. 100=-0.25%. 101=-0.5%. 110=-0.75%. 111=-1.0%. Note that, while this bit is located in the DAC address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.
19	**LOW_PWR**	RW	0x0	Lowers power (~100 uA) in the bandgap amplifier. This mode is useful in USB suspend or standby when bandgap accuracy is not critical. Note that while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.
18	**LW_REF**	RW	0x0	Lowers ADC and VAG reference voltages in 11:8 and 7:4 by ~22%. This bit must be set if VDDA is <1.7 V. Otherwise, VAG and ADC reference PSRR will be poor. Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.

Table 885. HW_AUDIOOUT_REFCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:16	**BIAS_CTRL**	RW	0x0	Bias current control for all analog blocks: 00=Nominal. 01=-20%. 10=-10%. 11=+10%. Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only. WARNING: Adjusting the bias current also changes the LRADC max voltage reference level. These bits should only be used for test/debug, and not in an application.
15:14	**RSRVD2**	RO	0x0	Reserved. Always write zeroes to this bit field.
13	**ADJ_ADC**	RW	0x0	ADC Reference Voltage Adjust. When cleared, analog ADC reference is 1.5 V. When set, ADC reference is controlled by ADCREFVAL. The ADJ_ADC bit should be cleared when the ADC/AUDIOIN is not in use (improves DAC SNR). When the ADC and DAC are both enabled, the ADJ_ADC bit must be set or the VAG and ADC references will interfere with each other and degrade SNR. Note that, although this bit is located in the DAC address space, it is an ADC function and is reset by the ADC SFTRST bit.
12	**ADJ_VAG**	RW	0x0	When cleared, VAG is VDD/2 (resistor divider). When set, VAG is controlled by VAGVAL 7:4. Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT SFTRST bit. It is reset by a power-on reset only.
11:8	**ADC_REFVAL**	RW	0x0	ADC Reference Value (when ADJADC set): F=1.60 V. 0=1.225 V, 25-mV steps, also affected by LWREF. Note that, although this bit is located in the DAC address space, it is an ADC function and is reset by the ADC SFTRST bit.
7:4	**VAG_VAL**	RW	0x0	VAG Reference Value (when ADJVAG set): F=1.00 V, 0=0.625 V, 25-mV steps, also affected by LWREF. See section on selecting the VAG level earlier in this chapter. Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.
3	**RSRVD1**	RO	0x0	Reserved. Always write zeroes to this bit field.
2:0	**DAC_ADJ**	RW	0x0	Adjusts the reference current (signal swing) in the DAC : 000=Nominal. 001=+0.25 dB. 010=+0.5 dB. 011=+0.75 dB. 100==-0.25 dB. 101=-0.5 dB. 110=-0.75 dB. 111=-1.0 dB. It is reset by a power-on reset only.

DESCRIPTION:

The AUDIOOUT Reference Control Register provides control over the voltage and power for the audio analog circuits.

EXAMPLE:

```
HW_AUDIOOUT_REFCTRL.ADJ_VAG = 1;
```

26.7.10. Miscellaneous Audio Controls Register Description

This register provides miscellenous audio control bits.

HW_AUDIOOUT_ANACTRL 0x80048090
HW_AUDIOOUT_ANACTRL_SET 0x80048094
HW_AUDIOOUT_ANACTRL_CLR 0x80048098
HW_AUDIOOUT_ANACTRL_TOG 0x8004809C

Table 886. HW_AUDIOOUT_ANACTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD8			SHORT_CM_STS	RSRVD7			SHORT_LR_STS	RSRVD6		SHORTMODE_CM		RSRVD5	SHORTMODE_LR		RSRVD4		SHORT_LVLADJL			RSRVD3	SHORT_LVLADJR			RSRVD2		HP_HOLD_GND	HP_CLASSAB	RSRVD1	EN_SPKR_ZCD	ZCD_SELECTADC	EN_ZCD

Table 887. HW_AUDIOOUT_ANACTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSRVD8**	RO	0x00	Reserved. Always write zeroes to this bit field. It is reset by a power-on reset only.
28	**SHORT_CM_STS**	RW	0x0	Status of common mode amplifier short detection: 0=No short. To clear this interrupt and then rearm it: (1) Set HW_AUDIOOUT_ANACTRL_SHORTMODE_CM to 00. (2) Clear this bit. (3) Set HW_AUDIOOUT_ANACTRL_SHORTMODE_CM to 01. There are two sets of edge-triggered latches in this path. All three steps must be executed to rearm the short detect. Note that this interrupt is non-maskable within the AUDIOOUT block.
27:25	**RSRVD7**	RO	0x0	Reserved. Always write zeroes to this bit field.
24	**SHORT_LR_STS**	RW	0x0	Status of headphone amplifier short detection: 0=No short. To clear this interrupt and then rearm it: (1) Set HW_AUDIOOUT_ANACTRL_SHORTMODE_LR to 00. (2) Clear this bit. (3) Set HW_AUDIOOUT_ANACTRL_SHORTMODE_LR to 01. There are two sets of edge-triggered latches in this path. All three steps must be executed to rearm the short detect. Note that this interrupt is non-maskable within the AUDIOOUT block.
23:22	**RSRVD6**	RO	0x0	Reserved. Always write zeroes to this bit field.

Table 887. HW_AUDIOOUT_ANACTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:20	**SHORTMODE_CM**	RW	0x0	Headphone Common Mode Amplifier Short Control Mode. 00=Reset analog latch, HW power down on unlatched short signal. 01=Latch short signal. HW power down on latched signal. 10=Do not use. 11=Do not use.
19	**RSRVD5**	RO	0x00	Reserved. Always write zeroes to this bit field.
18:17	**SHORTMODE_LR**	RW	0x0	Headphone Left and Right Channel Short Control mode. 00=Reset analog latch, HW power-down disabled. 01=Latch short signal, HW power-down enabled. 10=Do not use. 11=Do not use.
16:15	**RSRVD4**	RO	0x0	Reserved. Always write zeroes to this bit field.
14:12	**SHORT_LVLADJL**	RW	0x0	Adjust the left headphone current short detect trip point: 000=Nominal. 001=-25%. 010=-50%. 011=-75%. 100=+25%. 101=+50%. 110=+75%. 111=+100%. It is reset by a power-on reset only.
11	**RSRVD3**	RO	0x0	Reserved. Always write zeroes to this bit field.
10:8	**SHORT_LVLADJR**	RW	0x0	Adjust the right headphone current short detect trip point: 000=Nominal. 001=-25%. 010=-50%. 011=-75%. 100=+25%. 101=+50%. 110=+75%. 111=+100%. It is reset by a power-on reset only.
7:6	**RSRVD2**	RO	0x0	Reserved. Always write zeroes to this bit field.
5	**HP_HOLD_GND**	RW	0x0	Hold Headphone Output to Ground (used for power-up/power-down procedures). It is reset by a power-on reset only.
4	**HP_CLASSAB**	RW	0x0	Default is 0 (ClassA mode). ClassA mode can be useful for power-up/power-down and short handling. This bit should be set (ClassAB mode) before starting audio signal. It is reset by a power-on reset only.
3	**RSRVD1**	RO	0x0	Reserved. Always write zeroes to this bit field.
2	**EN_SPKR_ZCD**	RW	0x0	Enable Zero Cross Detect for Speaker Amplifier. This is a separate circuit from the headphone and ADC ZCD. It is reset by a power-on reset only.
1	**ZCD_SELECTADC**	RW	0x0	Set to one to enable ZCD on ADCMux amplifier (and disable ZCD on headphone amplifier). Set to zero for ZCD on headphone amplifier. Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.
0	**EN_ZCD**	RW	0x0	Enable Zero Cross Detect for Headphone Amplifier and/or ADC Mux Amplifier. SELECTADC bit 21 chooses between headphone and ADC amplifier select (they share a ZCD circuit). Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.

DESCRIPTION:

This register provides miscellaneous audio control bits.

EXAMPLE:

```
HW_AUDIOOUT_ANACTRL.EN_ZCD = 1; // Enable zero cross detect.
```

26.7.11. Miscellaneous Test Audio Controls Register Description

This register provides miscellaneous audio test bits.

HW_AUDIOOUT_TEST 0x800480a0
HW_AUDIOOUT_TEST_SET 0x800480a4
HW_AUDIOOUT_TEST_CLR 0x800480a8
HW_AUDIOOUT_TEST_TOG 0x800480aC

Table 888. HW_AUDIOOUT_TEST

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD6	HP_ANTIPOP			RSRVD5	TM_ADCIN_TOHP	TM_SPEAKER	TM_HPCOMMON	HP_I1_ADJ		HP_IALL_ADJ		SPKR_I1_ADJ		SPKR_IALL_ADJ		RSRVD4		VAG_CLASSA	VAG_DOUBLE_I	RSRVD3		HP_CHOPCLK		RSRVD2		DAC_CHOPCLK		RSRVD1	DAC_CLASSA	DAC_DOUBLE_I	DAC_DIS_RTZ

Table 889. HW_AUDIOOUT_TEST Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD6**	RO	0x0	Reserved. Always write zeroes to this bit field.
30:28	**HP_ANTIPOP**	RW	0x0	Reserved
27	**RSRVD5**	RO	0x0	Reserved. Always write zeroes to this bit field.
26	**TM_ADCIN_TOHP**	RW	0x0	Testmode to pipe ADC Mux Out (ADC In) to Headphone Output pins. No longer have ADC filter pins, this allows visibility to ADC Mux amp performance. To use this mode, the headphone load and the headphone board compensation must be removed (the ADC amp cannot drive it). Note that, although this bit is located in the DAC address space, it is an ADC function and is reset by the ADC SFTRST bit.
25	**TM_SPEAKER**	RW	0x0	Testmode to Pipe Speaker Pos to Microphone input. This is used for analog loopback DAC-Speaker-Mic-ADC Mode. There should be no load on the Mic input pin during this mode.
24	**TM_HPCOMMON**	RW	0x0	Uses headphone common VAG, instead of vaggate in ADC Mux. This is used for analog loopback DAC-HP-ADC mode to include common amp in path.
23:22	**HP_I1_ADJ**	RW	0x0	Adjusts bias current in first stage of headphone amplifier : 00=Nominal. 01=-50%. 10=+100%. 11=+50%. It is reset by a power-on reset only.

Table 889. HW_AUDIOOUT_TEST Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:20	**HP_IALL_ADJ**	RW	0x0	Adjusts all bias current in headphone amplifier : 00=nom, 01=-50%, 10=+50%, 11=-40%. It is reset by a power-on reset only.
19:18	**SPKR_I1_ADJ**	RW	0x0	Adjusts bias current in 1st stage of speaker amplifier : 00=nom, 01=-50%, 10=+100%, 11=+50%. It is reset by a power-on reset only.
17:16	**SPKR_IALL_ADJ**	RW	0x0	Adjusts all bias current in speaker amplifier : 00=Nominal. 01=-50%. 10=+50%. 11=-40%. It is reset by a power-on reset only.
15:14	**RSRVD4**	RO	0x0	Reserved. Always write zeroes to this bit field.
13	**VAG_CLASSA**	RW	0x0	Set to one to disable ClassAB mode in VAG Amp. Will increase current by ~200 uA. Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.
12	**VAG_DOUBLE_I**	RW	0x0	Set to one to double ClassA current in VAG amplifier (+240uA). Note that, while this bit is located in the AUDIOOUT address space, since it controls both DAC and ADC functions, it is not reset by the AUDIOOUT's SFTRST bit. It is reset by a power-on reset only.
11:10	**RSRVD3**	RO	0x00	Reserved. Always write zeroes to this bit field.
9:8	**HP_CHOPCLK**	RW	0x0	Enable chopping in the headphone and microphone amplifiers: 00=Disabled. 01=48 kHz. 10=96 kHz. 11=192 kHz.
7:6	**RSRVD2**	RO	0x0	Reserved. Always write zeroes to this bit field.
5:4	**DAC_CHOPCLK**	RW	0x0	Enable chopping in the DAC amplifier: 00=Disabled. 01=384 kHz. 10=192 kHz. 11=96 kHz.
3	**RSRVD1**	RO	0x0	Reserved. Always write zeroes to this bit field.
2	**DAC_CLASSA**	RW	0x0	Set to one to disable ClassAB mode in DAC. Will increase power by ~600 uA.
1	**DAC_DOUBLE_I**	RW	0x0	Set to one to double ClassA current in DAC amplifier (+360 uA in each DAC).
0	**DAC_DIS_RTZ**	RW	0x0	Set to one to disable DAC RTZ mode. Test-only bit that disables the return-to-zero function. This bit should remain cleared.

DESCRIPTION:

This register provides miscellaneous audio test bits..

EXAMPLE:

```
HW_AUDIOOUT_TEST.TM_HPCOMMON = 1; // Use headphone common VAG.
```

26.7.12. BIST Control and Status Register Description

The BIST Control and Status Register provides overall control of the integrated BIST engine.

HW_AUDIOOUT_BISTCTRL 0x800480b0
HW_AUDIOOUT_BISTCTRL_SET 0x800480b4
HW_AUDIOOUT_BISTCTRL_CLR 0x800480b8
HW_AUDIOOUT_BISTCTRL_TOG 0x800480bC

Table 890. HW_AUDIOOUT_BISTCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0																												FAIL	PASS	DONE	START

Table 891. HW_AUDIOOUT_BISTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	**RSVD0**	RO	0x0	Reserved
3	**FAIL**	RO	0x0	BIST has failed.
2	**PASS**	RO	0x0	BIST has passed
1	**DONE**	RO	0x0	BIST has completed.
0	**START**	RW	0x0	Reserved. Always write a 0 to this bit field.

DESCRIPTION:

The BISTCTRL Register provides overall control of the integrated BIST engine.

EXAMPLE:

```
HW_AUDIOUT_BISTCTRL.U = 0x00000000;
```

26.7.13. Hardware BIST Status 0 Register Description

This register provides visibility into memory failures detected by the BIST engine.

HW_AUDIOOUT_BISTSTAT0 0x800480c0
HW_AUDIOOUT_BISTSTAT0_SET 0x800480c4
HW_AUDIOOUT_BISTSTAT0_CLR 0x800480c8
HW_AUDIOOUT_BISTSTAT0_TOG 0x800480cC

Table 892. HW_AUDIOOUT_BISTSTAT0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD0								DATA																							

Table 893. HW_AUDIOOUT_BISTSTAT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSVD0**	RO	0x0	Reserved
23:0	**DATA**	RO	0x0	Failing data at the failing address.

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

DESCRIPTION:

The AUDIOOUT BISTSTAT0 provides visibility into memory failures detected by the BIST engine.

EXAMPLE:

```
HW_AUDIOUT_BISTSTAT0.U = 0x00000000;
```

26.7.14. Hardware AUDIOUT BIST Status 1 Register Description

The AUDIOOUT BISTATTS1 provides visibility into memory failures detected by the BIST engine.

HW_AUDIOOUT_BISTSTAT1 0x800480d0
HW_AUDIOOUT_BISTSTAT1_SET 0x800480d4
HW_AUDIOOUT_BISTSTAT1_CLR 0x800480d8
HW_AUDIOOUT_BISTSTAT1_TOG 0x800480dC

Table 894. HW_AUDIOOUT_BISTSTAT1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD1			STATE					RSVD0																ADDR							

Table 895. HW_AUDIOOUT_BISTSTAT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSVD1**	RO	0x0	Reserved
28:24	**STATE**	RO	0x0	Fail state of the BIST engine.
23:8	**RSVD0**	RO	0x0	Failing data at the failing address.
7:0	**ADDR**	RO	0x0	Failing data at the failing address.

DESCRIPTION:

The AUDIOOUT BISTATTS1 provides visibility into memory failures detected by the BIST engine.

EXAMPLE:

```
HW_AUDIOUT_BISTATTS1.U = 0x00000000;
```

26.7.15. Analog Clock Control Register Description

This register provides analog clock control.

HW_AUDIOOUT_ANACLKCTRL 0x800480e0
HW_AUDIOOUT_ANACLKCTRL_SET 0x800480e4
HW_AUDIOOUT_ANACLKCTRL_CLR 0x800480e8
HW_AUDIOOUT_ANACLKCTRL_TOG 0x800480eC

Table 896. HW_AUDIOOUT_ANACLKCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CLKGATE	RSRVD3																										INVERT_DACCLK	RSRVD2	DACDIV		

Table 897. HW_AUDIOOUT_ANACLKCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**CLKGATE**	RW	0x1	Analog clock Gate. Set this bit to gate the clocks for the DAC converter and associated digital filter. It is reset by a power-on reset only.
30:5	**RSRVD3**	RO	0x0	Reserved
4	**INVERT_DACCLK**	RW	0x0	DAC Clock Invert. Set this bit to invert the DAC_CLK for the DAC sigma-delta converter and associated digital filters.
3	**RSRVD2**	RO	0x0	Reserved
2:0	**DACDIV**	RW	0x0	DAC Analog Clock Divider. This bit field is used to select the oversampling clock rate used by the ADC. This bit field should only be changed per SigmaTel. 000=6 MHz. 001=4 MHz. 010/100=3 MHz. 011/101=2 MHz. 110=1.5 MHz. 111=1 MHz.

DESCRIPTION:

This register provides analog clock control.

EXAMPLE:

```
HW_AUDIOOUT_ANACLKCTRL.INVERT_DACCLK = 1; // Invert DAC clock.
```

26.7.16. AUDIOOUT Write Data Register Description

The AUDIOOUT Write Data Register provides a means to output PCM audio samples.

HW_AUDIOOUT_DATA 0x800480f0
HW_AUDIOOUT_DATA_SET 0x800480f4
HW_AUDIOOUT_DATA_CLR 0x800480f8
HW_AUDIOOUT_DATA_TOG 0x800480fC

Table 898. HW_AUDIOOUT_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
HIGH																LOW															

Table 899. HW_AUDIOOUT_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**HIGH**	RW	0x0000	Right Sample or Sample High Half-Word. For 16-bit sample mode, this field contains the right channel sample. For 32-bit sample mode, this field contains the most significant 16 bits of the 32-bit sample (either left or right).
15:0	**LOW**	RW	0x0000	Left Sample or Sample Low Half-Word. For 16-bit sample mode, this field contains the left channel sample. For 32-bit per sample mode, this field contains the least significant 16 bits of the 32-bit sample (either left or right).

DESCRIPTION:

The AUDIOOUT Write Data Register provides 32-bit data transfers for the DMA or alternatively can be directly written by the CPU. Each data value written to the register is placed in the AUDIOOUT's FIFO, which in turn is used by the digital FIR filter stages. These 32-bit values contain either one 32-bit sample or two 16-bit samples, depending on how the data size is programmed. Note that the PCM audio data input to the FIR filter stages is 24 bit. For 16-bit operation, the input data is sign extended to 24 bits. For 32-bit mode, it is normalized by dropping the least significant 8 bits.

EXAMPLE:

```
  HW_AUDIOOUT_DATA.U = 0x12345678; // write 0x1234 to the right channel and 0x5678 to the
left channel in 16 bit per sample mode
  HW_AUDIOOUT_DATA.U = 0x12345678; // write 0x12345678 to either the left or right channel in
32 bit per sample mode.
```

AUDIOOUT XML Revision: 1.67

27. SPDIF TRANSMITTER

This chapter describes the SPDIF transmitter provided on the STMP36xx. It includes sections on interrupts, clocking, DMA operation, and PIO debug mode. Programmable registers are described in Section 27.6.

27.1. Overview

The Sony-Philips Digital Interface Format (SPDIF) transmitter module transmits data according to the SPDIF digital audio interface standard (IEC-60958). Figure 122 shows a block diagram of the SPDIF transmitter module.

Data samples are transmitted as blocks of 192 frames, each frame consisting of two 32-bit sub-frames.

A 32-bit sub-frame is composed of a 4-bit preamble, a 24-bit data payload (e.g., a left- or right-channel PCM sample), and a 4-bit status field. The status fields are encoded according to the IEC-60958 consumer specification, reflecting the contents of the HW_SPDIF_FRAMCTRL and HW_SPDIF_CTRL registers. See the IEC-60958 specification for proper programming of these fields.

The sub-frame is transmitted serially, LSB-first, using a biphase-mark channel-coding scheme. This encoding allows an SPDIF receiver to recover the embedded clock signal. NOTE: Sub-frame information can be changed “on-the-fly” but is not reflected in the serial stream until the current frame is transmitted. This ensures consistency of the frame and the generated parity appended to that frame.

The SPDIF transmitter operates at one of three register-selectable base sample rates: 32 kHz, 44.1 kHz, or 48 kHz. Double-rate output (64 kHz, 88.2 kHz, and 96 kHz) can also be selected using the HW_SPDIF_SRR_BASEMULT register. The data-clock required to transmit an SPDIF frame at these sample-rates is generated using a fractional clock-divider. This divider uses both edges of the 480-MHz clock directly from the output of the PLL, divided by 4. This divider is located in the CLKCTRL module where all system clocks are generated; the resultant clock (pcm_spdif_clk) is output to the SPDIF module to be used for data transmission.

NOTE: The SPDIF module only operates within IEC-60958 Consumer Audio specifications when the PLL and HW_CLKCTRL_SPDIFCTRL_FREQ_DIV are appropriately programmed to provide a 120-MHz clock to the SPDIF_CLKCTRL module. Lower frequency clocks may be used, but will not meet the above specifications.

IMPLEMENTATION NOTE: The output of the spdif_clk_gen module (the pcm_spdif_clk) must be declared as a generated clock, and any signals crossing between this clock-domain (pcm_spdif) and the apb_clk domain must be appropriately clock-crossed (2-flop, metastability circuit).

NOTE: A LUT may be implemented in the CLKCTRL module to allow clock-generation for SPDIF when PLL output frequency is programmed from 360–480 MHz.

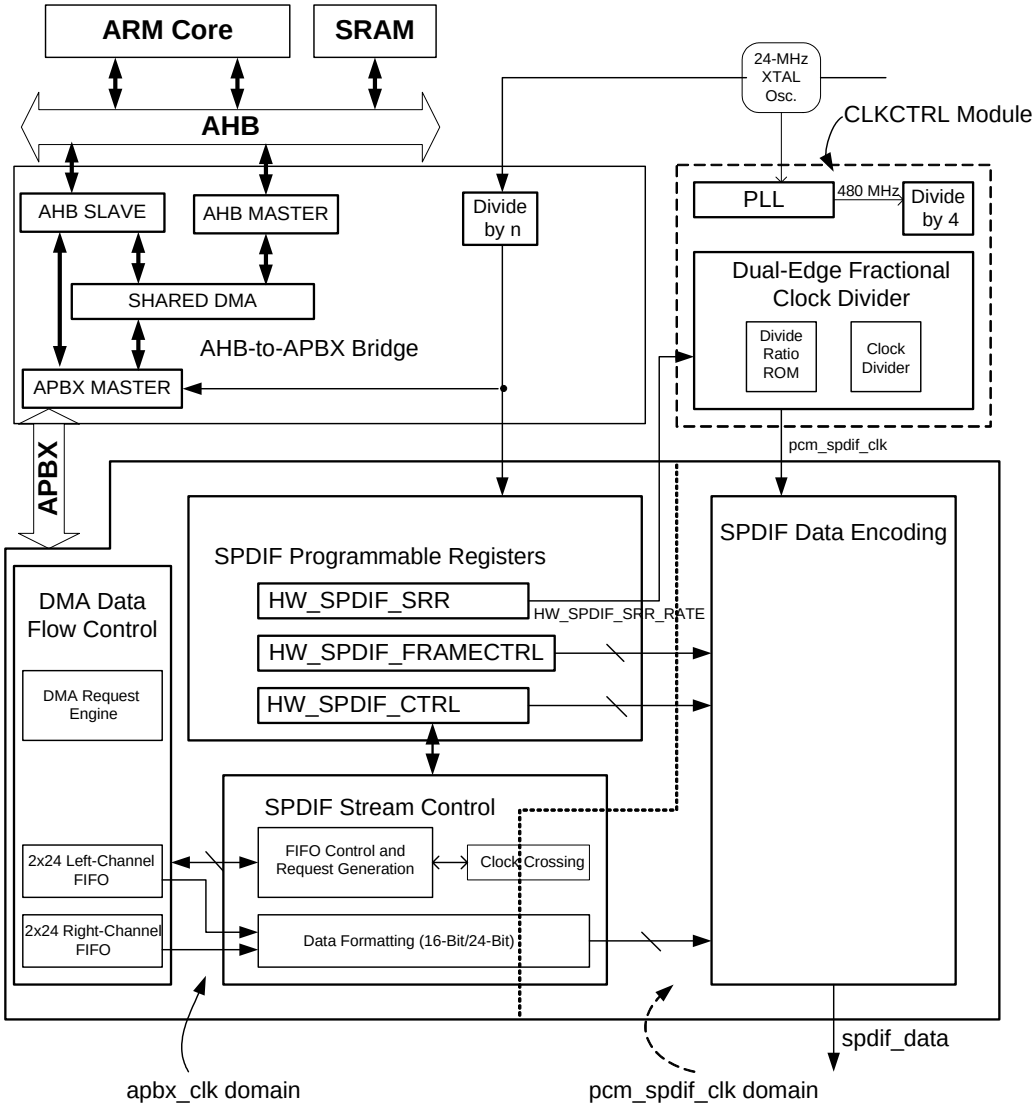


Figure 122. SPDIF Transmitter Block Diagram

The SPDIF module receives data by one of two methods:

- Software-directed PIO writes to the HW_SPDIF_WDATA register
- Appropriate programming of the DMA-engine. (See Chapter 11, "AHB-to-APBX Bridge with DMA" on page 257, for a detailed description of the DMA module and how to perform DMA data transfers to/from modules and memory.)

Once provided by the DMA, the received data is placed in a 2x24 word FIFO for each channel, left and right. At initialization, the FIFO is filled before SPDIF data transfer occurs. After this, data is requested whenever this FIFO has an empty entry or at a nominal rate corresponding to the programmed sample-rate in HW_SPDIF_SRR.

The behavior of the SPDIF module during or after a FIFO underflow is programmable. On detection of an underflow event, the SPDIF module sends the current sam-

ple for four frames before muting (sending zeros) the data stream based on the configuration of HW_SPDIF_FRAMECTRL_AMUTE. The final validity unit embedded within each frame dictates whether the receiver processes the data within that frame. HW_SPDIF_FRAMECTRL determines the behavior of this bit.

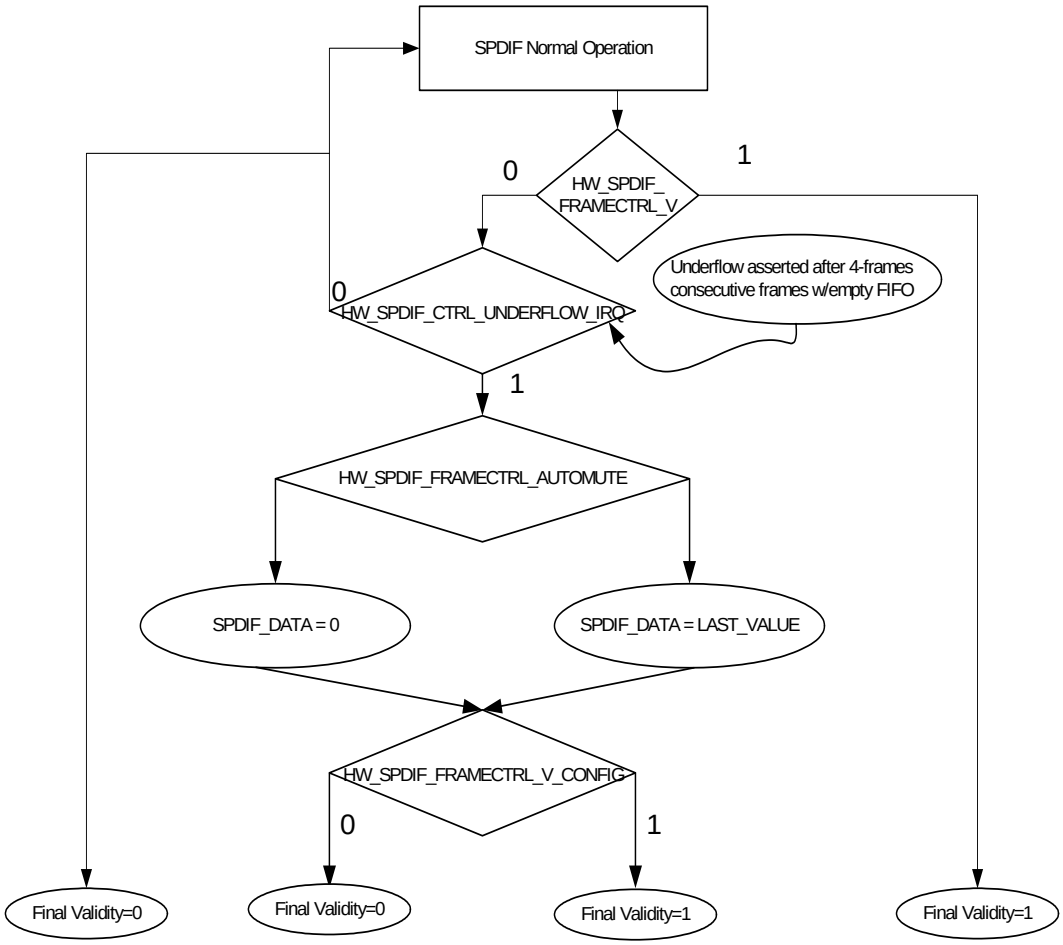


Figure 123. SPDIF Flow Chart

SPDIF data can be transmitted in one of two modes: 32-bit mode and 16-bit mode. Selection between these modes is done with the HW_SPDIF_CTRL_DATA_WIDTH register. In either case, data samples must be interleaved in main memory for proper behavior, although in 32-bit mode, 32-bit words are interleaved and in 16-bit mode, 16-bit words are interleaved.

- When HW_SPDIF_CTRL_DATA_WIDTH=0, 32-bit mode is enabled, and HW_SPDIF_WDATA contains either the left or right data sample. Since the SPDIF frame allows for transmission of only 24 bits, only the 24 MSBs stored in the HW_SPDIF_WDATA register will be transmitted.
- Alternately, when HW_SPDIF_CTRL_DATA_WIDTH=1, 16-bit mode is enabled, and the HW_SPDIF_WDATA register will contain one of each left AND right samples. The data transmitted in the SPDIF frame will be these 16 MSBs with 8 zeros appended in the LSB positions.

NOTE: If the data supplied actually represents a lower resolution analog-to-digital conversion, this information is not captured by the SPDIF transmitter, which always reports a 24-bit sample-size.

27.2. Interrupts

The SPDIF module contains a single interrupt source that is asserted on FIFO overflows and/or FIFO underflows. This interrupt is enabled by setting HW_SPDIF_CTRL_FIFO_ERR_IRQ_EN. On interrupt detection, the HW_SPDIF_CTRL_UNDERFLOW_IRQ and HW_SPDIF_CTRL_OVERFLOW_IRQ fields can be polled for the exact cause of the interrupt and appropriate action taken.

Note: These bits remain valid for polling, regardless of the state of the interrupt enable.

27.3. Clocking

The IEC-60958 specification outlines the requirements for SPDIF clocking. The SPDIF module is designed according to the Consumer Audio requirements. These dictate that:

- Average Sample-Rate Error must not exceed 1000 ppm
- Maximum Instantaneous Jitter must not exceed ~4.4 ns.

The jitter requirement implies either a single-phase of a >240-MHz clock or both phases of a 120-MHz clock. It also implies the use of a fractional divider for which the divisors are maintained to sufficient significant digits to yield the required ppm tolerance. The SPDIF module in the STMP36xx uses nine-bit fractional coefficients that yield an average frequency error of 52 ppm. These coefficients are determined according to the required clock-rates that are dictated by the sample rates implemented. The required clock frequencies provided by the CLKCTRL module for the implemented sample-rates are:

F(48 kHz) $\geq$ 6.144 MHz

F(44.1 kHz) $\geq$ 5.6448 MHz

F(32 kHz) $\geq$ 4.096 MHz

F(96 kHz) $\geq$ 12.288 MHz

F(88.2 kHz) $\geq$ 11.2896 MHz

F(64 kHz) $\geq$ 8.192 MHz

All clocks within the SPDIF module are gated according to the state of HW_SPDIF_CTRL_CLKGATE. When set, all clocks derived from the apb_clk are gated. Gating of the pcm_spdif_clk is accomplished through HW_CLKCTRL_SPDIFCLKCTRL_CLKGATE. A module-level reset is also provided in HW_SPDIF_CTRL_SFTRST. Setting this bit performs a module-wide reset and subsequent assertion of the HW_SPDIF_CTRL_CLKGATE.

NOTE: A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

27.4. DMA Operation

Using the SPDIF module in DMA mode involves configuring the appropriate DMA channel to provide the interleaved data blocks stored in memory. See Chapter 11, "AHB-to-APBX Bridge with DMA" on page 257 for detailed information on DMA programming. Once programmed, the DMA engine references a set of linked DMA descriptors stored by the CPU in main memory. These descriptors point to data blocks stored in system memory and also provide a mechanism for automated PIO writes before transfer of a data-block. Figure 124 describes a typical set of descriptors required to transmit two data-blocks.

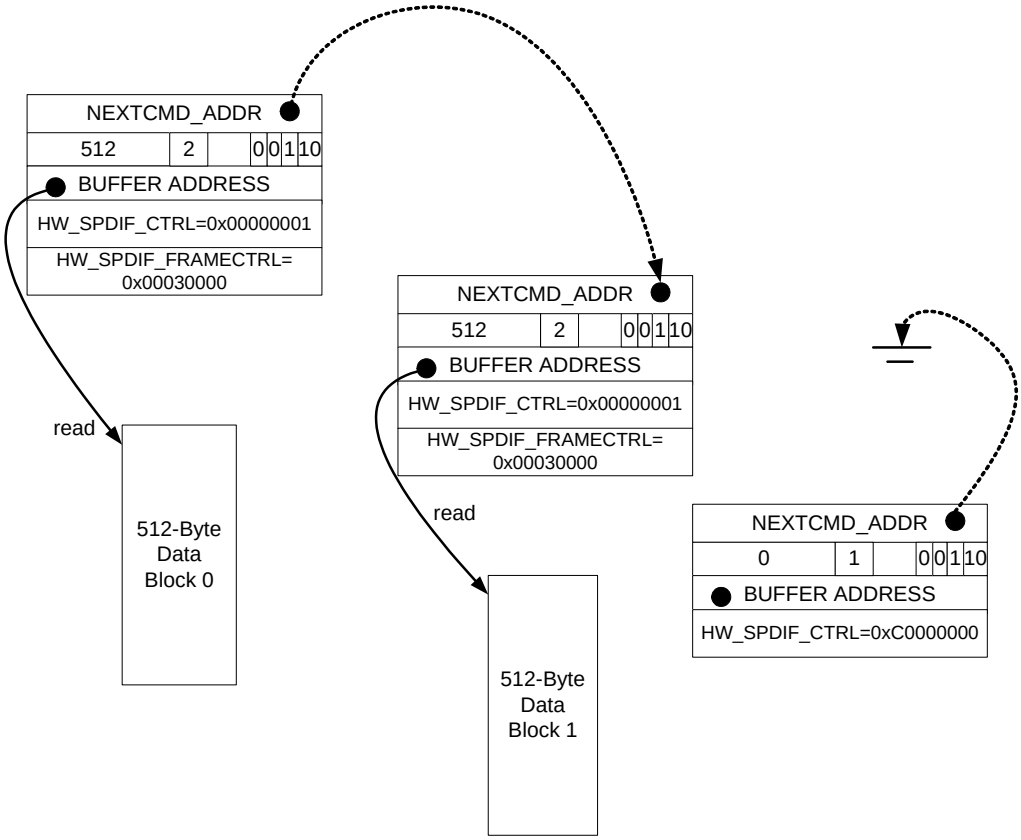


Figure 124. SPDIF DMA Two-Block Transmit Example

Here, the DMA is instructed to perform two PIO writes prior to toggling the DMA_PCMDKICK signal**:**

- HW_SPDIF_CTRL_UNDERFLOW_IRQ_EN is set to enable interrupts on FIFO underflow detect
- HW_SPDIF_FRAMECTRL_AMUTE and HW_SPDIF_FRAMECTRL_V_CONFIG are set to mute and tag the data stream as INVALID on a FIFO underflow.

The DMA engine is then programmed to transfer 512-bytes to the SPDIF module.

Additionally, the SPDIF module contains a mechanism for "throttling" DMA requests to the DMA engine. This circuit is programmed using the HW_SPDIF_CTRL_DMAWAIT

field and corresponds to the number of cycles of the apb_clk to wait before toggling the DMA_PREQ signal to the DMA engine.

NOTE: Considering that the bandwidth requirements of the SPDIF module are minimal (not in excess of 96 kHz) and burst requests occur only in pairs, this field can be ignored for most, if not all, applications.

There is a floor APBX frequency below which the SPDIF cannot work without errors. That frequency can be calculated as follows:

- Assume that there are 6 other blocks apart from SPDIF on the APBX bus, and it takes 4 APBX clock cycles to service each block. If the number of clock cycles required to service each block changes, change the equations accordingly.
- Assume that HW_SPDIF_CTRL_DMA_WAIT is less than DMA LATENCY. If this is not true, then even DMA WAIT has to be added to the calculation and the floor APBX frequency increases further.

In 16-bit Mode:

```
Floor APBX freq = (DMA latency + 9) * sample rate.
For max DMA latency = (6 blocks) x (4 cycles per block) = 24 cycles and
max SPDIF sample rate = 96 kHz,
min APBX freq = 3.168 MHz.
```

In 32-bit Mode:

(A) Ideal Calculation:

```
min freq = [2*(DMA latency+4) + 7] * sample rate.
For max DMA latency = 24 cycles and max SPDIF sample rate = 96 kHz,
min APBX freq = 6.048 MHz.
```

(B) Simpler Calculation:

```
Floor APBX freq = 2*(latency + 9) * sample rate = twice that of 16-bit
mode.
For max latency = 24 cycles and max sample rate = 96 kHz,
min APBX freq = 6.336 MHz.
```

Option A is ideal as it allows a lower floor frequency; option B can be used to keep it simple and avoid confusion.

27.5. PIO Debug Mode of Operation

The block is connected only as a PIO device to the APBX bus. Even though it is designed to work with the DMA controller integrated in the APBX bridge, all transfers to and from the block are programmed I/O (PIO) read or write cycles. When the DMA is ready to write to the HW_SPDIF_DATA register, it does so with standard APB write cycles. There *are* four DMA related signals that connect the SPDIF transmitter to the DMA, *but* all data transfers are standard PIO cycles on the APB. The state of these four signals can be seen in the HW_SPDIF_DEBUG register.

Thus, it is possible to completely exercise the SPDIF block for diagnostic purposes, using only load and store instructions from the CPU without ever starting the DMA controller. This section describes how to interact with the block using PIO operations, and also defines the block's detailed behavior.

Whenever the HW_SPDIF_CTRL register is written to, by either the CPU or the DMA, it establishes the basic operation mode for the block. If the HW_SPDIF_CTRL register is written with a one in the RUN bit, then the operation begins and the SPDIF attempts to read the data block by toggling its PDMAREQ signal to the DMA.

Notice that the PDMAREQ signal is defined as a “toggle” signal. This changes state to signify either a request for another DMA word or a notification that the current command transfer has been ended by the SPDIF. Diagnostic software should poll these signals to determine when the SPDIF is ready for another DMA write, and can then supply data by storing a 32-bit word to the HW_SPDIF_DATA register, just as the DMA would do in normal operation.

To perform SPDIF transfers in PIO debug mode, diagnostic software should perform the following:

1. Program the HW_CLKCTRL_SPDIFCLKCTRL register correctly and wait for the PLL to lock.
2. Turn off the Soft Reset bit, HW_SPDIF_CTRL_SFTRST**,** and the Clock Gate bit, HW_SPDIF_CTRL_CLKGATE.
3. Properly configure the subcode information by writing the HW_SPDIF_FRAMECTRL register. NOTE: See IEC-60958 for proper coding of these fields.
4. Enable the SPDIF transmitter by setting the HW_SPDIF_CTRL_RUN register.
5. Wait for HW_SPDIF_DEBUG_DMA_PREQ status bit to toggle.
6. Write one sample of the left/right DATA block data to the HW_SPDIF_DATA register.
7. Repeat 5 and 6 until all samples have been written to HW_SPDIF_DATA.

27.6. Programmable Registers

The following registers provide control for programmable elements of the SPDIF module.

27.6.1. SPDIF Control Register Description

The SPDIF Control Register provides overall control of the SPDIF converter.

HW_SPDIF_CTRL 0x80054000
HW_SPDIF_CTRL_SET 0x80054004
HW_SPDIF_CTRL_CLR 0x80054008
HW_SPDIF_CTRL_TOG 0x8005400C

Table 900. HW_SPDIF_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSRVD1									DMAWAIT_COUNT					RSRVD0										WAIT_END_XFER	WORD_LENGTH	FIFO_UNDERFLOW_IRQ	FIFO_OVERFLOW_IRQ	FIFO_ERROR_IRQ_EN	RUN

Table 901. HW_SPDIF_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Setting this bit to one forces a reset to the entire block and then gates the clocks off. This bit must be set to zero for normal operation.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. WARNING: First set the CLKGATE bit in the HW_CLKCTRL_SPDIFCLKCTRL register to 1. Only then, set this bit to 1 to prevent any extra samples from being transmitted. When removing clock gating, follow the reverse order: First reset this CLKGATE bit to 0, and then reset the CLKGATE bit in the HW_CLKCTRL_SPDIFCLKCTRL register to 0 .
29:21	**RSRVD1**	RO	0x00	Reserved
20:16	**DMAWAIT_COUNT**	RW	0x00	DMA Request Delay Count. This bit field specifies the number of APBX clock cycles (0 to 31) to delay before each DMA request. This field acts as a throttle on the bandwidth consumed by the SPDIF block. This field can be loaded by the DMA.
15:6	**RSRVD0**	RO	0x000	Reserved
5	**WAIT_END_XFER**	RW	0x1	Set this bit to a one if the SPDIF Transmitter should wait until the internal FIFO is empty before halting transmission based on deassertion of RUN. Use in conjuntion with HW_SPDIF_STAT_END_XFER to determine transfer completion
4	**WORD_LENGTH**	RW	0x0	Set this bit to one to enable 16-bit mode. Set this bit to zero for 32-bit mode. In either case, the SPDIF frame allows transmission of only 24 bits. In 16-bit mode, eight zeros will be appended to the LSB's of the input sample; in 32-bit mode, the 24 MSB's of HW_SPDIF_WDATA will be transmitted.
3	**FIFO_UNDERFLOW_IRQ**	RW	0x0	This bit is set by hardware if the FIFO underflows during SPDIF transmission. It is reset in software by writing a zero to the bit position or by writing a one to the SCT clear address space.
2	**FIFO_OVERFLOW_IRQ**	RW	0x0	This bit is set by hardware if the FIFO overflows during SPDIF transmission. It is reset by software by writing a zero to the bit position or by writing a one to the SCT clear address space.
1	**FIFO_ERROR_IRQ_EN**	RW	0x0	Set this bit to one to enable a SPDIF interrupt request on FIFO overflow or underflow status conditions.
0	**RUN**	RW	0x0	Setting this bit to one causes the SPDIF to begin converting data. The actual conversion will begin when the SPDIF FIFO is filled (4 or 8 words written, depending upon sample word format, i.e., 16 or 32 bits).

DESCRIPTION:

The SPDIF Control Register contains the overall control for SPDIF sample formats, loopback mode, and interrupt controls.

EXAMPLE:

```
HW_SPDIF_CTRL.RUN = 1; // start SPDIF conversion
```

27.6.2. SPDIF Status Register Description

The SPDIF Status Register reflects overall status of the SPDIF converter.

HW_SPDIF_STAT 0x80054010
HW_SPDIF_STAT_SET 0x80054014
HW_SPDIF_STAT_CLR 0x80054018
HW_SPDIF_STAT_TOG 0x8005401C

Table 902. HW_SPDIF_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PRESENT	RSRVD1																														END_XFER

Table 903. HW_SPDIF_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**PRESENT**	RO	0x1	This bit is set to 1 in products in which SPDIF is present.
30:1	**RSRVD1**	RO	0x0	Reserved
0	**END_XFER**	RO	0x0	When set, indicates that the SPDIF module has completed transfer of all data, including data stored in internal FIFOs. Used in conjunction with HW_SPDIF_CTRL_WAIT_END_XFER.

DESCRIPTION:

The SPDIF Status Register provides the status of the SPDIF converter.

EXAMPLE:

```
unsigned TestBit = HW_SPDIF_STAT.PRESENT;
```

27.6.3. SPDIF Frame Control Register Description

The SPDIF Frame Control Register provides direct control of the control bits transmitted over an SPDIF frame.

HW_SPDIF_FRAMECTRL 0x80054020
HW_SPDIF_FRAMECTRL_SET 0x80054024
HW_SPDIF_FRAMECTRL_CLR 0x80054028
HW_SPDIF_FRAMECTRL_TOG 0x8005402C

Table 904. HW_SPDIF_FRAMECTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2														V_CONFIG	AUTO_MUTE	RSRVD1	USER_DATA	V	L	RSRVD0	CC							PRE	COPY	AUDIO	PRO

Table 905. HW_SPDIF_FRAMECTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:18	**RSRVD2**	RO	0x0	Reserved
17	**V_CONFIG**	RW	0x1	Defines SPDIF behavior when sending invalid frames. 0:Do NOT tag frame as invalid. 1: Tag frame as invalid.
16	**AUTO_MUTE**	RW	0x0	Auto-Mute Stream on stream-suspend detect.
15	**RSRVD1**	RO	0x0	Reserved
14	**USER_DATA**	RW	0x0	User data transmitted during each sub-frame. Consult IEC Standard for additional details.
13	**V**	RW	0x0	Indicates that a sub-frame's samples are invalid. If V=0, the sub-frame is indicated as valid, that is, correctly transmitted and received by the interface. If V=1, the subframe is indicated as invalid.
12	**L**	RW	0x0	Generation level is defined by the IEC standard, or as appropriate.
11	**RSRVD0**	RO	0x0	Reserved
10:4	**CC**	RW	0x0	Category code is defined by the IEC standard, or as appropriate.
3	**PRE**	RW	0x0	0: No Pre-Emphasis. 1: Pre-Emphasis is 50/15 usec.
2	**COPY**	RW	0x0	0: Copyright bit NOT asserted. 1: Copyright bit asserted.
1	**AUDIO**	RW	0x0	AUDIO=0:PCM Data;1. AUDIO=Non-PCM Data
0	**PRO**	RW	0x0	0: Consumer use of the channel. 1: Professional use of the channel.

DESCRIPTION:

The SPDIF Frame Control Register provides direct control of the control bits transmitted over an SPDIF frame.

EXAMPLE:

```
HW_SPDIF_FRAMECTRL.COPY=1 //SPDIF frame contains copyrighted material
```

27.6.4. SPDIF Sample Rate Register Description

The SPDIF Sample Rate Register controls the sample rate of the data stream played back from the circular buffer.

HW_SPDIF_SRR 0x80054030
HW_SPDIF_SRR_SET 0x80054034
HW_SPDIF_SRR_CLR 0x80054038

HW_SPDIF_SRR_TOG 0x8005403C

Table 906. HW_SPDIF_SRR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1	BASEMULT			RSRVD0								RATE																			

Table 907. HW_SPDIF_SRR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD1**	RO	0x0	Reserved
30:28	**BASEMULT**	RW	0x1	Base-Rate Multiplier. 1=Single-Rate(48 kHz). 2=Double-Rate (96 kHz).
27:20	**RSRVD0**	RO	0x0	Reserved
19:0	**RATE**	RW	0x00000	Sample-Rate Conversion Factor. The only valid entries are: 0x07D00, 0x0AC44, 0x0BB80 // 38K, 44.1K, 48K

DESCRIPTION:

The SPDIF Sample Rate Register provides a RATE field for specifying the sample rate conversion factor to use in outputting the current SPDIF stream.

EXAMPLE:

```
HW_SPDIF_SRR.B.RATE = 0x0AC44; // 44.1KHz
```

27.6.5. SPDIF Debug Register Description

The SPDIF Debug Register provides read-only access to various internal state information that may be useful for block debugging and validation.

HW_SPDIF_DEBUG 0x80054040
HW_SPDIF_DEBUG_SET 0x80054044
HW_SPDIF_DEBUG_CLR 0x80054048
HW_SPDIF_DEBUG_TOG 0x8005404C

Table 908. HW_SPDIF_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD1																														DMA_PREQ	FIFO_STATUS

Table 909. HW_SPDIF_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	**RSRVD1**	RO	0x00	Reserved
1	**DMA_PREQ**	RO	0x0	DMA request status. This read-only bit reflects the current state of the SPDIF's DMA request signal. DMA requests are issued any time the request signal toggles. This bit can be polled by software, in order to manually move samples to the SPDIF's FIFO from a memory buffer when the SPDIF's DMA channel is not used
0	**FIFO_STATUS**	RO	0x1	This bit is set when the FIFO has empty space. This reflects a DMA request being generated.

DESCRIPTION:

This is a read-only register used for checking FIFO status and PIO mode of operation.

EXAMPLE:

```
unsigned TestBit = HW_SPDIF_DEBUG.DMA_PREQ;
```

27.6.6. SPDIF Write Data Register Description

The SPDIF Write Data Register receives 32-bit data transfers from the DMA. It deposits the data into an internal FIFO and from there into the SPDIF stream. These 32-bit writes contain either one 32-bit sample or two 16-bit samples.

HW_SPDIF_DATA 0x80054050
HW_SPDIF_DATA_SET 0x80054054
HW_SPDIF_DATA_CLR 0x80054058
HW_SPDIF_DATA_TOG 0x8005405C

Table 910. HW_SPDIF_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
HIGH																LOW															

Table 911. HW_SPDIF_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**HIGH**	RW	0x0000	For 16-bit mode, this field contains the entire right channel sample. For 32-bit mode, this field contains the 16 MSBs of the 32-bit sample (either left or right).
15:0	**LOW**	RW	0x0000	For 16-bit mode, this field contains the entire left channel sample. For 32-bit mode, this field contains the 16 LSBs of the 32-bit sample (either left or right).

DESCRIPTION:

Writing a 32-bit value to the register corresponds to pushing that 32-bit value into the SPDIF FIFO. The DMA writes 32-bit values to this register. In 32-bit-per-sample mode, the DMA is writing either one full left sample or one full right sample for each

write to this register. For 16-bit mode, the DMA is writing a 16-bit left sample and a 16-bit right sample for each 32-bit write to this register.

EXAMPLE:

```
  HW_SPDIF_DATA = 0x12345678; // write 0x1234 to the right channel and 0x5678 to the left
channel in 16-bit mode
  HW_SPDIF_DATA = 0x12345678; // write 0x12345678 to either the left or right channel in 32-
bit per sample mode.
```

SPDIF XML Revision: 1.26

28. DIGITAL RADIO INTERFACE (DRI)

This chapter describes the digital radio interface included on the STMP36xx. Programmable registers are described in Section 28.4.

28.1. Overview

The STMP36xx implements a digital interface to a digital radio receiver. Details of the receiver are described in a separate data sheet.

The interface consists of two digital input signals that share pins with the analog line inputs. These pins can be used for either their analog functions or their digital functions in any given application, but not both.

Figure 125 shows a block diagram of the digital radio interface, and Figure 126 describes DRI synchronization and data recovery.

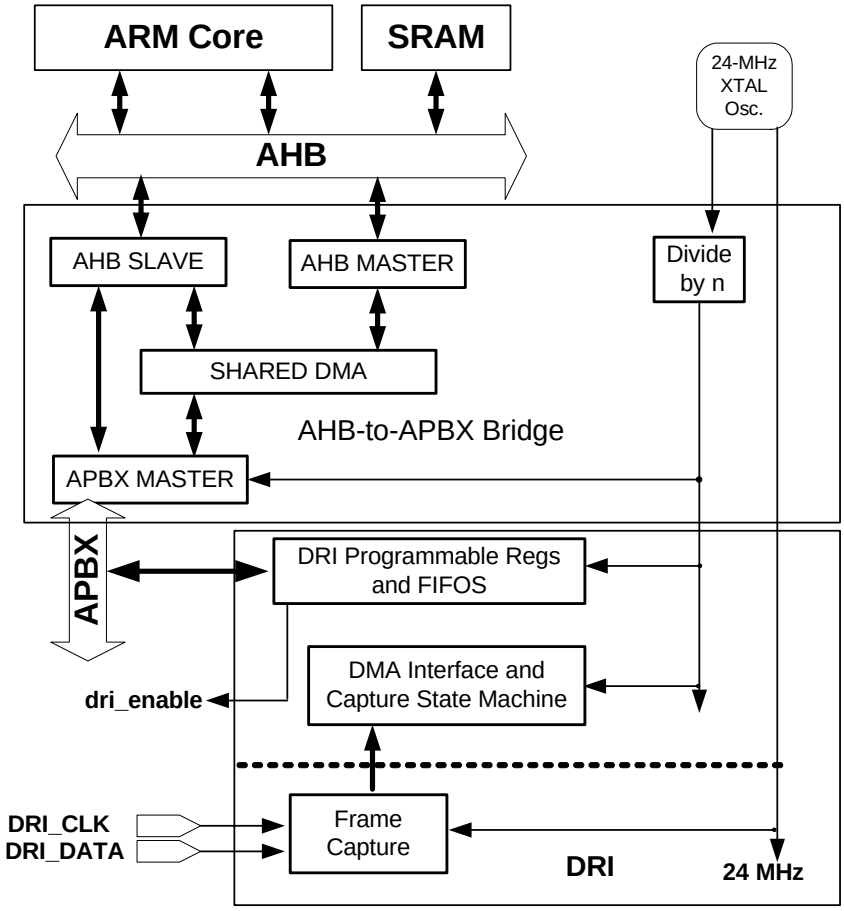


Figure 125. Digital Radio Interface (DRI) Block Diagram

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

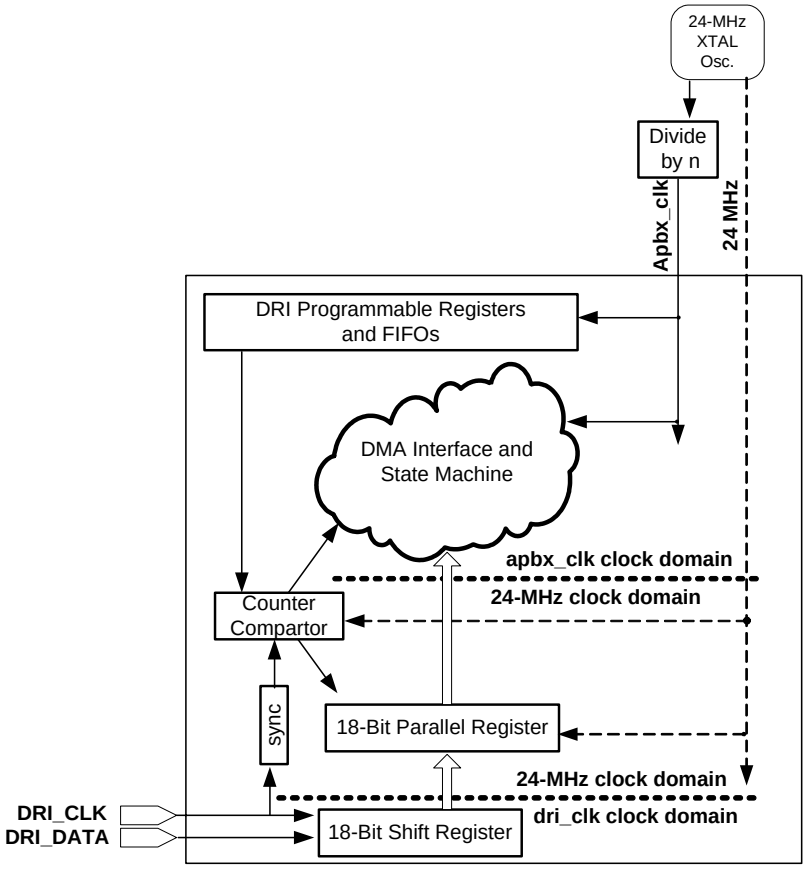


Figure 126. DRI Synchronization and Data Recovery

28.2. Frame Structure

The frame structure used on the digital radio interface is shown in Figure 127. The 8-clock gap can vary from as short as 8 dri_clks at 4 MHz (12 clocks at 6 MHz) to as long as allowed by the desired bandwidth. Dri_clk is expected to run at 4 MHz or 6 MHz on this interface, i.e., it is expected to have a period of 250 ns or 166 ns except for the gap interval.

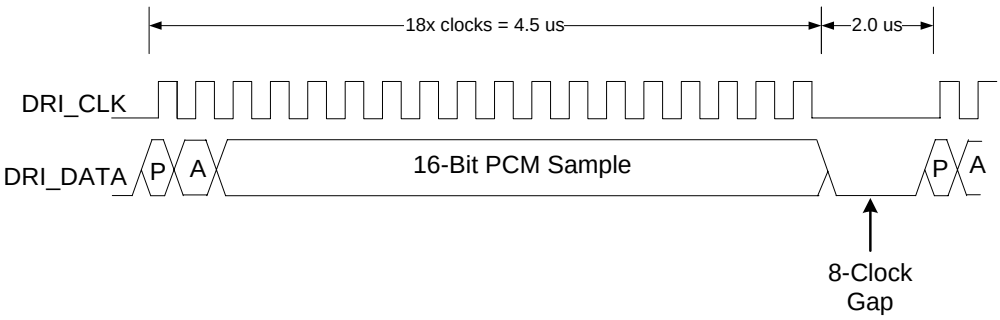


Figure 127. Digital Radio Interface (DRI) Framing

The gap is detected by synchronizing the dri_clk to 24.0 MHz. A counter is incremented for each 24-MHz clock. The rising edge of each synchronized dri_clk is used to reset the counter. If the counter ever reaches the threshold established in HW_DRI_TIMING_GAP_DETECTION_INTERVAL, then the contents of the 18-bit shift register are captured into the 24-MHz domain, and the apbx_clk based state machine is notified.

The bits within the 18-bit frame are shown in Table 912 and described in Table 913.

Table 912. Digital Radio Interface Frame

17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Pilot Peak	Attention	16-Bit PCM Sample															

Table 913. DRI Frame Bit Field Descriptions

BITS	LABEL	DEFINITION
17	**PILOT PEAK**	Set to one at the pilot peak. Set to zero for the following N frames, where N is 7 for 4 MHz and 11 for 6 MHz. This is the first bit shifted in from the frame.
16	**ATTENTION**	Set to one by receiver to request attention from software. Software uses the I^2C bus to determine the cause of the attention request. When set to one in a frame, this bit causes an attention interrupt to be set in the **HW_DRI_CTRL** register.
15:0	**PCM SAMPLE**	The 16 bit PCM sample value is pushed into the DMA FIFO and from there to a system memory buffer.

The PILOT_PEAK bit generally occurs on every eighth PCM sample when pilot synchronization has been achieved in the digital radio. The DRI hardware monitors this bit to ensure that it occurs on every eighth PCM sample. An interrupt is generated to the CPU if pilot synchronization is lost. In addition, HW_DRI_STATUS_PHASE_OFFSET indicates the phase shift of the last Pilot Peak received relative to the phase established immediately after reset. This allows software to make programmatic phase shifts without restarting the DMA.

The two external inputs, dri_clk and dri_data, are supplied as 1.8-V digital pins that are connected to the analog line inputs. When the dri_enable bit is set, these inputs become digital inputs, as shown in Figure 128.

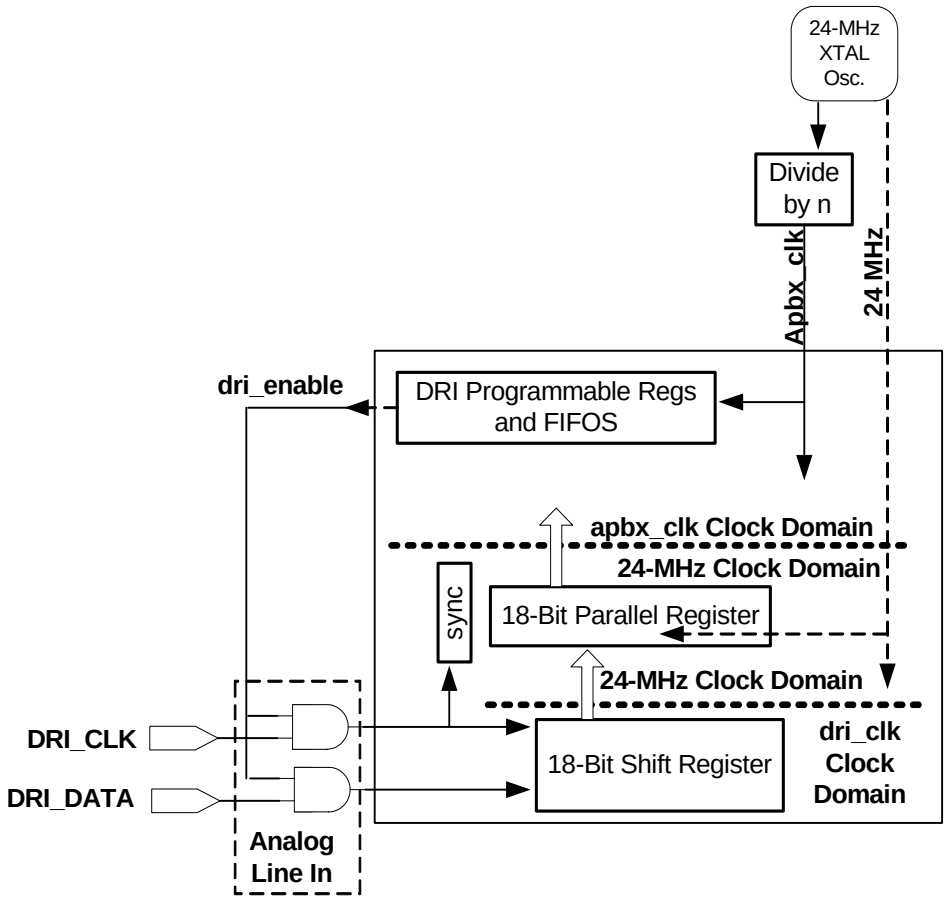


Figure 128. Digital Radio Interface (DRI) Digital Signals into Analog Line-In

28.3. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, "Correct Way to Soft Reset a Block" on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

28.4. Programmable Registers

The following registers describe the programming interface for the digital radio interface (DRI).

28.4.1. DRI Control Register Description

The DRI Control Register specifies the reset state and the interrupt controls for the DRI controller.

HW_DRI_CTRL 0x80074000
HW_DRI_CTRL_SET 0x80074004
HW_DRI_CTRL_CLR 0x80074008
HW_DRI_CTRL_TOG 0x8007400C

Table 914. HW_DRI_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	ENABLE_INPUTS	RSVD4		STOP_ON_OFLOW_ERROR	STOP_ON_PILOT_ERROR	RSVD3					DMA_DELAY_COUNT				REACQUIRE_PHASE	RSVD2			OVERFLOW_IRQ_EN	PILOT_SYNC_LOSS_IRQ_EN	ATTENTION_IRQ_EN	RSVD1					OVERFLOW_IRQ	PILOT_SYNC_LOSS_IRQ	ATTENTION_IRQ	RUN

Table 915. HW_DRI_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. RUN = 0x0 Allow DRI to operate normally. RESET = 0x1 Hold DRI in reset.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. RUN = 0x0 Allow DRI to operate normally. NO_CLKS = 0x1 Do not clock DRI gates in order to minimize power consumption.
29	**ENABLE_INPUTS**	RW	0x0	Set this bit to one to enable the input pads in a digital input mode instead of line-in left/right normal analog mode. ANALOG_LINE_IN = 0x0 Use line inputs in their analog mode. DRI_DIGITAL_IN = 0x1 Use line inputs in their special DRI digital mode.
28:27	**RSVD4**	RO	0x0	Always set this bit field to zero.
26	**STOP_ON_OFLOW_ERROR**	RW	0x0	This bit is set to one to cause the run bit to reset and the DMA transfers to stop on the detection of an overflow condition. IGNORE = 0x0 Ignore a DMA overflow condition and keep transferring samples to the DMA. STOP = 0x1 Stop DMA transfers when a DMA overflow condition is detected.

Table 915. HW_DRI_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25	**STOP_ON_PILOT_ERROR**	RW	0x0	This bit is set to one to cause the run bit to reset and the DMA transfers to stop on the detection of a loss of sync condition. IGNORE = 0x0 Ignore a loss of pilot sync condition and keep transferring samples to the DMA. STOP = 0x1 Stop DMA transfers when a loss of pilot sync is detected.
24:21	**RSVD3**	RO	0x0	Always set this bit field to zero.
20:16	**DMA_DELAY_COUNT**	RW	0x01	This bit field used to encode the number of idle cycles that must be placed between successive DMA requests. It no longer peforms this function. These bits are now spares.
15	**REACQUIRE_PHASE**	RW	0x0	This bit is set to one to cause the state machine to reacquire its phase alignment with the pilot peak marker in the eighteenth bit. This bit will be reset to zero by the hardware when the next pilot peak marker is received. NORMAL = 0x0 Normal operation with existing phase relationship. NEW_PHASE = 0x1 Reacquire new phase.
14:12	**RSVD2**	RO	0x0	Always set this bit field to zero.
11	**OVERFLOW_IRQ_EN**	RW	0x0	This bit is set to enable an overflow interrupt. DISABLED = 0x0 Interrupt Request Disabled. ENABLED = 0x1 Interrupt Request Enabled.
10	**PILOT_SYNC_LOSS_IRQ_EN**	RW	0x0	This bit is set to enable a pilot sync loss interrupt. DISABLED = 0x0 Interrupt Request Disabled. ENABLED = 0x1 Interrupt Request Enabled.
9	**ATTENTION_IRQ_EN**	RW	0x0	This bit is set to enable an attention interrupt from the DRI. DISABLED = 0x0 Interrupt Request Disabled. ENABLED = 0x1 Interrupt Request Enabled.
8:4	**RSVD1**	RO	0x0	Always set this bit field to zero.
3	**OVERFLOW_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the DRI controller. This bit is cleared by software by writing a one to its SCT clear address. A DMA FIFO overrun was detected, PCM samples have been lost. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
2	**PILOT_SYNC_LOSS_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the DRI controller. This bit is cleared by software by writing a one to its SCT clear address. This bit is set if the expected pilot peak bit is not seen at the expected eight-sample boundary. Firmware should consider resynchronizing its data framing in the DMA buffers. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

Table 915. HW_DRI_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**ATTENTION_IRQ**	RW	0x0	This bit is set to indicate that an interrupt is requested by the DRI controller. This bit is cleared by software by writing a one to its SCT clear address. This bit is set in response to the detection of an attention bit in a DRI frame. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
0	**RUN**	RW	0x0	Set this bit to one to enable the DRI Controller operation. This bit is automatically set by DMA commands that write to CTRL1 after the last PIO write of the DMA command. For soft DMA operation, software can set this bit to enable the controller. See note in HW_DRI_DEBUG1 register about when to manually set this bit. There are cases in which the DMA should be used to kick off the digtial radio interface. HALT = 0x0 No DRI command in progress. RUN = 0x1 Process a slave or master DRI command.

DESCRIPTION:

This register is either written by the DMA or the CPU, depending on the state of an DRI transaction.

EXAMPLE:

```
// turn off soft reset and clock gating
HW_DRI_CTRL_CLR(BM_DRI_CTRL_SFTRST| BM_DRI_CTRL_CLKGATE);
```

28.4.2. DRI Timing Register Description

The DRI Timing Register specifies the detailed timing used for the DRI controller.

HW_DRI_TIMING 0x80074010

Table 916. HW_DRI_TIMING

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2												PILOT_REP_RATE				RSVD1								GAP_DETECTION_INTERVAL							

Table 917. HW_DRI_TIMING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	**RSVD2**	RO	0x0	Always set this bit field to zero.
19:16	**PILOT_REP_RATE**	RW	0xC	Set this bit field to the number of DRI samples between pilot sample centers. Set to 8 for 4-MHz. Set to 0xC for 6-MHz operation.
15:8	**RSVD1**	RO	0x0	Always set this bit field to zero.
7:0	**GAP_DETECTION_INTERVAL**	RW	0x10	Set this bit field to the number of 24-MHz crystal clocks to count to detect a gap in DRI source clocks.

DESCRIPTION:

This register is either written by the DMA or the CPU depending on the state of an DRI transaction.

EXAMPLE:

```
// set the gap detection interval
HW_DRI_TIMING_WR(0x00000022);
```

28.4.3. DRI Status Register Description

The DRI controller reports status information in this register.

HW_DRI_STAT 0x80074020

Table 918. HW_DRI_STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DRI_PRESENT	RSVD3											PILOT_PHASE				RSVD2												OVERFLOW_IRQ_SUMMARY	PILOT_SYNC_LOSS_IRQ_SUMMARY	ATTENTION_IRQ_SUMMARY	RSVD1

Table 919. HW_DRI_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**DRI_PRESENT**	RO	0x1	This read-only bit indicates that the DRI interface is present when it reads back a one. This DRI function is not available on a device that returns a zero for this bit field. UNAVAILABLE = 0x0 DRI is not present in this product. AVAILABLE = 0x1 DRI is present in this product.
30:20	**RSVD3**	RO	0x0	Always set this bit field to zero.

Table 919. HW_DRI_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:16	**PILOT_PHASE**	RO	0x0	This bit indicates the phase of the last pilot peak marker received from the digital radio relative to a phase established when the DRI block was removed from soft reset state and saw its first pilot peak. This value will be zero until the first phase error is noticed.
15:4	**RSVD2**	RO	0x0	Always set this bit field to zero.
3	**OVERFLOW_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the DRI controller. It is the logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
2	**PILOT_SYNC_LOSS_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the DRI controller. It is the logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
1	**ATTENTION_IRQ_SUMMARY**	RO	0x0	This bit is set to indicate that an interrupt is requested by the DRI controller. It is the logical AND of the corresponding interrupt status bit and interrupt enable bit. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
0	**RSVD1**	RO	0x0	Always set this bit field to zero.

DESCRIPTION:

The status register provides read-only access to the function presence bits, as well as the pilot phase detector and the interrupt summaries.

EXAMPLE:

```
  if(HW_DRI_STAT.DRI_PRESENT != BV_DRI_STAT_SLAVE_BUSY__AVAILABLE)// then wait till it fin-
ishes
```

28.4.4. DRI Controller DMA Read Data Register Description

The DRI DMA Read Data Register is the target for both source and destination DMA transfers. This register is backed by an eight-deep FIFO.

HW_DRI_DATA 0x80074030

Table 920. HW_DRI_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 921. HW_DRI_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x00000000	The DMA channel reads from this address.

DESCRIPTION:

DMA reads are directed to this register.

EXAMPLE:

```
The DMA data register is used by the DMA to read or write data from the DRI controller as
mediated by the DRI controller`s DMA request signal.
```

28.4.5. DRI Device Debug Register 0 Description

The DRI Device Debug Register 0 provides a diagnostic view into the internal state machine and states of the DRI device.

HW_DRI_DEBUG0 0x80074040
HW_DRI_DEBUG0_SET 0x80074044
HW_DRI_DEBUG0_CLR 0x80074048
HW_DRI_DEBUG0_TOG 0x8007404C

Table 922. HW_DRI_DEBUG0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DMAREQ	DMACMDKICK	DRI_CLK_INPUT	DRI_DATA_INPUT	TEST_MODE	PILOT_REP_RATE	SPARE								FRAME																	

Table 923. HW_DRI_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**DMAREQ**	RO	0x0	Read-only view of the toggle state of the DMA request line.
30	**DMACMDKICK**	RO	0x0	Read-only view of the toggle state of the DMA request line.
29	**DRI_CLK_INPUT**	RO	0x0	Read-only view of the state of the DRI clock input signal from the analog logic.
28	**DRI_DATA_INPUT**	RO	0x0	Read-only view of the state of the DRI data input signal from the analog logic.
27	**TEST_MODE**	RW	0x0	Set to one to enable a special internal test mode that supplies pseudo DRI frames as DRI_CLK and DRI_DATA inputs. The integrated test source has been removed for synthesis. This is now a spare bit.
26	**PILOT_REP_RATE**	RW	0x0	This bit is set to one to select a 12-sample repeat cycle for pilot sync (6-MHz case) in the test mode. Set to zero for an 8-sample repeat cycle (4-MHz case). NOTE: This bit only affects a built in test generator not the operating mode of the DRI. The integrated test source has been removed for synthesis. This is now a spare bit. 8_AT_4MHZ = 0x0 At 4 MHz, there is 1 pilot per 8 samples. 12_AT_6MHZ = 0x1 At 6 MHz, there is 1 pilot per 12 samples.
25:18	**SPARE**	RW	0x00	Spares for patching hardware in metal.
17:0	**FRAME**	RO	0x0000	Current state of frame synchronizing register received from the digital radio receiver.

DESCRIPTION:

This register provides access to various internal states and controls that are used in diagnostic modes of operation.

EXAMPLE:

```
  while(HW_DRI_DEBUG0.DMAREQ == old_dma_req_value); // wait for next dma request toggle
  old_dma_req_value = HW_DRI_DEBUG0.DMAREQ; // remember the new state of the dma request tog-
gle
```

28.4.6. DRI Device Debug Register 1 Description

The DRI Device Debug Register 1 provides a diagnostic view into the swizzle Frame Register of the DRI device.

HW_DRI_DEBUG1 0x80074050
HW_DRI_DEBUG1_SET 0x80074054
HW_DRI_DEBUG1_CLR 0x80074058
HW_DRI_DEBUG1_TOG 0x8007405C

Table 924. HW_DRI_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
INVERT_PILOT	INVERT_ATTENTION	INVERT_DRI_DATA	INVERT_DRI_CLOCK	REVERSE_FRAME	RSVD1									SWIZZLED_FRAME																	

Table 925. HW_DRI_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**INVERT_PILOT**	RW	0x0	This bit is set to one to invert the frame register bit used for the pilot peak indicator. NORMAL = 0x0 Normal clock polarity. INVERTED = 0x1 Inverted clock polarity.
30	**INVERT_ATTENTION**	RW	0x0	This bit is set to one to invert the frame register bit used for the attention bit. NORMAL = 0x0 Normal clock polarity. INVERTED = 0x1 Inverted clock polarity.
29	**INVERT_DRI_DATA**	RW	0x0	This bit is set to one to invert the DRI_DATA prior to shifting into the shift register. NORMAL = 0x0 Normal clock polarity. INVERTED = 0x1 Inverted clock polarity.
28	**INVERT_DRI_CLOCK**	RW	0x0	This bit is set to one to invert the DRI_CLK edge used to shift data into the shift register. NORMAL = 0x0 Normal clock polarity. INVERTED = 0x1 Inverted clock polarity.
27	**REVERSE_FRAME**	RW	0x0	This bit is set to one to reverse the bit order of the 18-bit frames received from the digital radio. NORMAL = 0x0 Normal clock polarity. REVERSED = 0x1 Inverted clock polarity.

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

Table 925. HW_DRI_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26:18	**RSVD1**	RO	0x0	Always set this bit field to zero.
17:0	**SWIZZLED_FRAME**	RO	0x0000	Current state of frame synchronizing register received from the digital radio receiver as swizzled by the various insurance flip bits.

DESCRIPTION:

This register provides access to various internal states and controls that are used in diagnostic modes of operation. NOTE: When the INVERT_PILOT, INVERT_ATTENTION, INVERT_DRI_DATA, INVERT_DRI_CLOCK or REVERSE_FRAME bits are set to one, the DMA should not be used to kick off digital radio interface transfers. Instead, software should manually set the run bit after the DMA is initialized and the values in DEBUG1 have been established.

EXAMPLE:

```
fram = HW_DRI_DEBUG1.SWIZZELED_FRAME; // then pilot peak is set for this frame;
```

DRI XML Revision: 1.36

29. LOW-RESOLUTION ADC AND TOUCH-SCREEN INTERFACE

This chapter describes the low-resolution analog-to-digital converters and touch-screen interface included on the STMP36xx. It includes sections on scheduling conversions and delay channels. Programmable registers are described in Section 29.5.

29.1. Overview

The eight-channel low-resolution ADC (LRADC) block is used for voltage measurement. Figure 129 shows a block diagram of the LRADC. Six channels are available for general use.

- One channel is dedicated to measure the voltage on the BATT pin (LRADC7) and can be used to sense the amount of battery life remaining.
- One channel is dedicated to measuring the voltage on the VDDIO Rail (LRADC6). and is used to calibrate the voltage levels measured on the auxiliary channels.
- The other channels, LRADC0–LRADC5, measure the voltage on the six application-dependent LRADC pins. The auxiliary channels can be used for a variety of different uses, including a resistor-based wired remote control, temperature sensing, touch screen, etc.

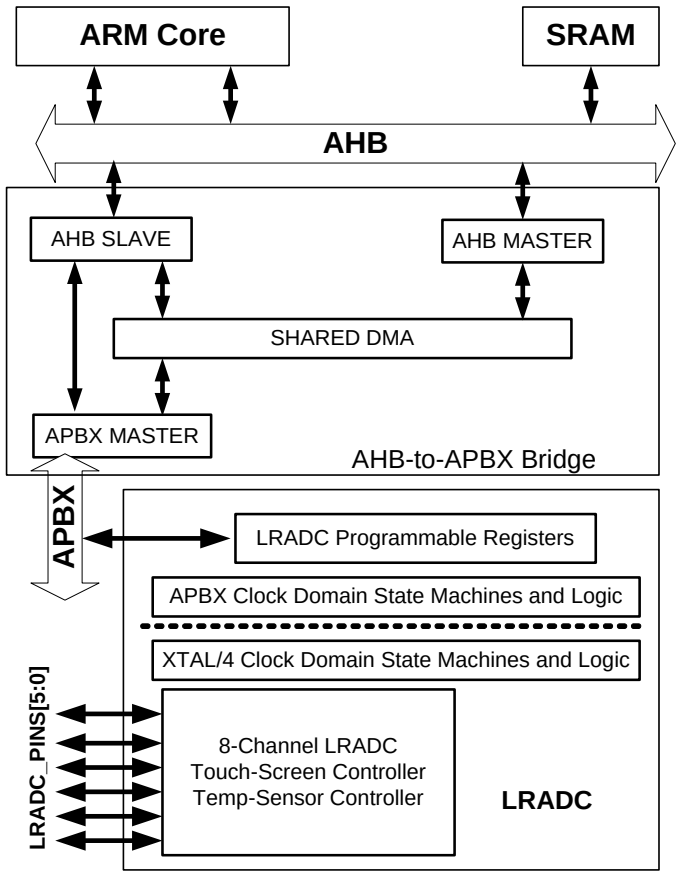


Figure 129. Low-Resolution ADC and Touch-Screen Interface Block Diagram

The LRADC has 12 bits of resolution and an absolute accuracy of 2%.

All channels sample on the same divided clock rate from the 24.0-MHz crystal clock.

The LRADC controller includes an integrated touch-screen controller with drive voltage generation for touch-screen coordinate measurement as well as a touch-detection interrupt circuit.

The LRADC controller also contains four delay control channels that can be used to automatically time and schedule control events within the LRADC.

All eight channels of the LRADC share a common successive approximation style analog-to-digital converter through a common analog mux front end (see Figure 130).

The BATT pin has a built-in 2:1 voltage divider on its analog multiplexor input, so that it can measure battery voltages that are at a higher potential than the VDDIO rail. All other channel inputs are restricted by the VDDIO rail voltage.

The touch-screen driver was designed to drive typical touch-screen impedances of 200–900 ohm (measured across X or Y terminals). However, it should work for higher impedance touch-screens as well. The touch-detect feature may not work reliably for touchscreen impedances greater than 20 kohm (40 kohm total impedance across X and Y dimensions). For higher impedance touch-screens, it may be necessary to use the LRADC to sample “xplus” to determine a touch event, rather than using the touch-detect feature.

The LRADC channels 0 and 1 have optional current source outputs to allow these channels to be used with an external thermistor for temperature sensing. The controls for these current sources are in LRADC_CTRL2. The current source values can be changed to allow significant temperature sensing range. The currents are derived using the on-chip 1% accurate bandgap voltage reference and the accurate external 620-ohm resistor. The bandgap voltage is accurate to 1% and 620-ohm external resistor should also be 1% accurate. With the addition of current mirror errors, the total error of the temperature sensor current sources should be typically within 3%. Most thermistors are no more than 5% accurate, so this level of current source accuracy is acceptable to almost all applications.

29.2. Scheduling Conversions

The APB-X clock domain logic schedules conversions on a per channel basis and handles interrupt processing back to the CPU. Each of the eight channels has its own interrupt request enable bit and its own interrupt request status bit.

There is a schedule request bit for each channel, HW_LRADCCSR0_SCHEDULE. Setting this bit causes the LRADC to schedule a conversion for that channel. Each channel schedule bit is sequentially checked and, if scheduled, causes a conversion. The schedule bit is cleared upon completion of a successive approximation conversion, and its corresponding interrupt request status bit is set. Thus, software controls how often a conversion is requested. As each scheduled channel is converted, its interrupt status bit is set and its schedule bit is reset. There is a mechanism to continuously reschedule a conversion for a particular channel.

With set/clear/toggle addressing modes, independent threads can request conversions without needing any information from unrelated threads using other channels. Setting a schedule bit can be performed in an atomic way. Setting a “gang” of four channel-schedule bits can also be performed atomically. The LRADC scheduler is round-robin. It snapshots all schedule bits at once, and then processes them in sequence until all are converted. It then monitors the schedule bits. If any schedule

bits are set, it snapshots them and starts a new conversion operation for all scheduled channels. Thus, one can set the schedule bits for four channels on the same clock edge. The channel with the largest channel number is converted last and has its interrupt status bit set last. If that channel is the only one of the four with an interrupt enable bit set, then it interrupts the ARM after all four channels have been converted, effectively ganging four channels together.

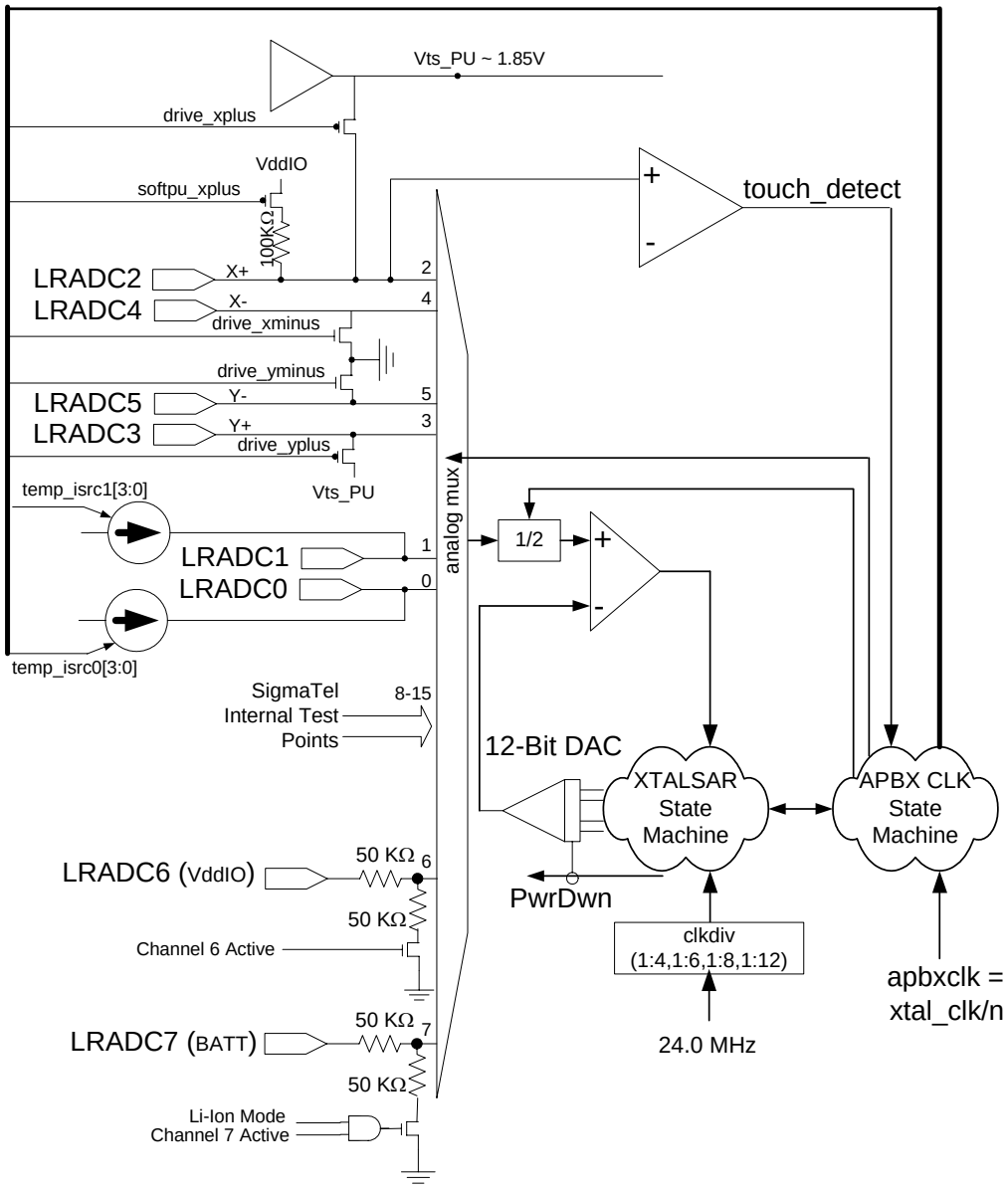


Figure 130. Low-Resolution ADC Successive Approximation Unit

29.3. Delay Channels

To minimize the interrupt load on the ARM processor, four delay channels are provided. Each has an 11-bit counter that increments at 2 kHz. A delay channel can be kicked off either by an ARM store instruction or at the completion of a delay channel time out. At time-out, each channel has the option of kicking off any combination of LRADC conversions, as well as any combination of delay channels.

Consider the case of a touch-screen that requires 4 x oversampling of its coordinate values. Further, suppose you wish to receive an oversampled X or Y coordinate approximately every 5 ms and that the oversampling should be spaced at 1-ms intervals.

- In the touch-screen, first select either X or Y drive, then setup the appropriate LRADC.
- In setting up the LRADC, clear the accumulator associated with it by setting the ACCUMULATE bit and set the NUM_SAMPLES field to 3 (4 samples before interrupt request).
- Next, setup two delay channels.
  - Delay Channel 1 is set to delay 1 ms (DELAY = 1, two ticks) and then kick the schedule bit for LRADC 4. Its LOOP_COUNT bit field is also set to 3, so that four kicks of LRADC 4 occur, each spaced by 1 ms.
  - Delay Channel 0 is set to delay 1 ms with LOOP_COUNT = 0, i.e., one time. Its TRIGER_DELAYS field is set to trigger Delay Channel 1 when it times out. The ISR routine kicks off Delay Channel 0 immediately before it does its return from interrupt. Another interrupt (LRADC4_IRQ) is asserted once the entire 4x oversample data capture is complete. A sample timeline for such a sequence is shown in Figure 131.

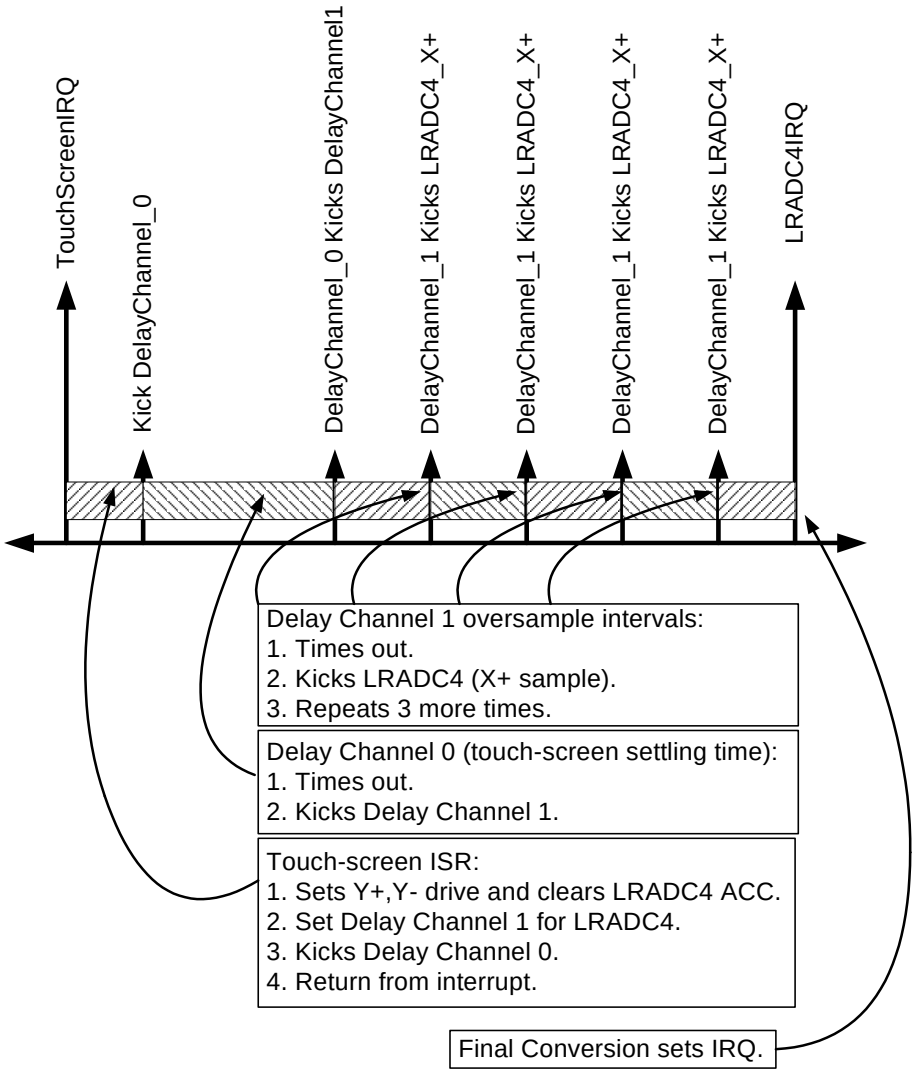


Figure 131. Using Delay Channels to Oversample a Touch-Screen

WARNING: The pad ESD protection limits maximum voltage on all LRADC inputs. The BATT LRADC is specifically designed to handle higher voltages, but LRADC1–LRADC7 inputs are limited to VDDIO.

29.4. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

29.5. Programmable Registers

The following programmable registers are available to software for controlling and using the low-resolution analog-to-digital converters.

29.5.1. LRADC Control Register 0 Description

The LRADC Control Register 0 provides overall control of the eight low-resolution analog-to-digital converters.

HW_LRADC_CTRL0 0x80050000
HW_LRADC_CTRL0_SET 0x80050004
HW_LRADC_CTRL0_CLR 0x80050008
HW_LRADC_CTRL0_TOG 0x8005000C

Table 926. HW_LRADC_CTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	RSRVD2								ONCHIP_GROUNDREF	TOUCH_DETECT_ENABLE	YMINUS_ENABLE	XMINUS_ENABLE	YPLUS_ENABLE	XPLUS_ENABLE	RSRVD1								SCHEDULE							

Table 927. HW_LRADC_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	When set to one, this bit causes a reset to the entire LRADC block. In addition, it turns off the converter clock and powers down the analog portion of the LRADC. Set this bit to zero for normal operation.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29:22	**RSRVD2**	RO	0x0	Reserved
21	**ONCHIP_GROUNDREF**	RW	0x0	Set this bit to one to use the on-chip ground as reference for conversions. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
20	**TOUCH_DETECT_ENABLE**	RW	0x0	Set this bit to one to enable the touch-panel touch detector. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
19	**YMINUS_ENABLE**	RW	0x0	Set this bit to one to enable the yminus pulldown on the LRADC5 pin. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.

Table 927. HW_LRADC_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
18	**XMINUS_ENABLE**	RW	0x0	Set this bit to one to enable the xminus pulldown on the LRADC4 pin. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
17	**YPLUS_ENABLE**	RW	0x0	Set this bit to one to enable the yplus pullup on the LRADC3 pin. Both HW_LRADC_CTRL3_FORCE_ANALOG_PWUP and HW_LRADC_CTRL3_PWD40UA_PWUP must be set to one for this switch to turn on. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
16	**XPLUS_ENABLE**	RW	0x0	Set this bit to one to enable the xplus pullup on the LRADC2 pin. Both HW_LRADC_CTRL3_FORCE_ANALOG_PWUP and HW_LRADC_CTRL3_PWD40UA_PWUP must be set to one for this switch to turn on. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
15:8	**RSRVD1**	RO	0x00	Reserved
7:0	**SCHEDULE**	RW	0x00	Setting a bit to one schedules the corresponding LRADC channel to be converted. When the conversion of a scheduled channel is completed, the corresponding schedule bit is reset by the hardware and the corresponding interrupt request is set to one. Thus, any thread can request a conversion asynchronously from any other thread.

DESCRIPTION:

The LRADC control register provides control over the shared eight-channel LRADC converter. It allows software to independently schedule conversion cycles on any number of any size subsets of the eight channels. In addition, it allows software to manage the interrupt reporting for the channel conversion sets.

EXAMPLE:

```
BW_LRADC_CTRL0_YPLUS_ENABLE(BV_LRADC_CTRL0_YPLUS_ENABLE__ON);
```

29.5.2. LRADC Control Register 1 Description

The LRADC Control Register 1 provides overall control of the eight low-resolution analog-to-digital converters.

HW_LRADC_CTRL1 0x80050010
HW_LRADC_CTRL1_SET 0x80050014
HW_LRADC_CTRL1_CLR 0x80050018
HW_LRADC_CTRL1_TOG 0x8005001C

Table 928. HW_LRADC_CTRL1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2							TOUCH_DETECT_IRQ_EN	LRADC7_IRQ_EN	LRADC6_IRQ_EN	LRADC5_IRQ_EN	LRADC4_IRQ_EN	LRADC3_IRQ_EN	LRADC2_IRQ_EN	LRADC1_IRQ_EN	LRADC0_IRQ_EN	RSRVD1							TOUCH_DETECT_IRQ	LRADC7_IRQ	LRADC6_IRQ	LRADC5_IRQ	LRADC4_IRQ	LRADC3_IRQ	LRADC2_IRQ	LRADC1_IRQ	LRADC0_IRQ

Table 929. HW_LRADC_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSRVD2**	RO	0x00	Reserved
24	**TOUCH_DETECT_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for the touch detector comparator. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
23	**LRADC7_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 7 (BATT) conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
22	**LRADC6_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 6 (VddIO) conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
21	**LRADC5_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 5 conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
20	**LRADC4_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 4 conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
19	**LRADC3_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 3 conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
18	**LRADC2_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 2 conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
17	**LRADC1_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 1 conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.

Table 929. HW_LRADC_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	**LRADC0_IRQ_EN**	RW	0x0	Set to one to enable an interrupt for channel 0 conversions. DISABLE = 0x0 Disable Interrupt request. ENABLE = 0x1 Enable Interrupt request.
15:9	**RSRVD1**	RO	0x00	Reserved
8	**TOUCH_DETECT_IRQ**	RW	0x0	This bit is set to one upon detection of a touch condition in the touch panel attached to LRADC2-LRADC5. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
7	**LRADC7_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 7 (BATT). It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
6	**LRADC6_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 6 (VDDIO). It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
5	**LRADC5_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 5. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
4	**LRADC4_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 4. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
3	**LRADC3_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 3. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.

Table 929. HW_LRADC_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	**LRADC2_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 2. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
1	**LRADC1_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 1. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
0	**LRADC0_IRQ**	RW	0x0	This bit is set to one upon completion of a scheduled conversion for channel 0. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.

DESCRIPTION:

The LRADC Control Register 1 provides control over the shared eight-channel LRADC converter. It allows software to independently schedule conversion cycles on any number of any size subsets of the eight channels. In addition, it allows software to manage the interrupt reporting for the channel conversion sets.

EXAMPLE:

```
if(HW_LRADC_CTRL1.TOUCH_DETECT_IRQ == BV_LRADC_CTRL1_TOUCH_DETECT_IRQ__PENDING){
// Then handle the interrupt.
HW_LRADC_CTRL1.TOUCH_DETECT_IRQ_EN = BV_LRADC_CTRL1_TOUCH_DETECT_IRQ_EN__DISABLE;
}
```

29.5.3. LRADC Control Register 2 Description

The LRADC Control Register 2 provides overall control of the eight low-resolution analog-to-digital converters.

HW_LRADC_CTRL2 0x80050020
HW_LRADC_CTRL2_SET 0x80050024
HW_LRADC_CTRL2_CLR 0x80050028
HW_LRADC_CTRL2_TOG 0x8005002C

Table 930. HW_LRADC_CTRL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DIVIDE_BY_TWO								LRADC6SELECT				LRADC7SELECT				RSRVD2						TEMP_SENSOR_IENABLE1	TEMP_SENSOR_IENABLE0	TEMP_ISRC1				TEMP_ISRC0			

Table 931. HW_LRADC_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**DIVIDE_BY_TWO**	RW	0x0	Each bit of this eight bit field corresponds to a channel of an LRADC. Setting the bit to one caused the A/D converter to use its analog divide-by-two circuit for the conversion of the corresponding channel.
23:20	**LRADC6SELECT**	RW	0x6	This bit field selects which analog mux input is used for conversion on LRADC channel 6 (VddIO). CHANNEL0 = 0x0 CHANNEL1 = 0x1 CHANNEL2 = 0x2 CHANNEL3 = 0x3 CHANNEL4 = 0x4 CHANNEL5 = 0x5 CHANNEL6 = 0x6 VDDIO CHANNEL7 = 0x7 BATTERY CHANNEL8 = 0x8 PMOS THIN CHANNEL9 = 0x9 NMOS THIN CHANNEL10 = 0xA NMOS THICK CHANNEL11 = 0xB PMOS THICK CHANNEL12 = 0xC CHANNEL13 = 0xD CHANNEL14 = 0xE CHANNEL15 = 0xF
19:16	**LRADC7SELECT**	RW	0x7	This bit field selects which analog mux input is used for conversion on LRADC channel 7 (BATT). CHANNEL0 = 0x0 CHANNEL1 = 0x1 CHANNEL2 = 0x2 CHANNEL3 = 0x3 CHANNEL4 = 0x4 CHANNEL5 = 0x5 CHANNEL6 = 0x6 VDDIO CHANNEL7 = 0x7 BATTERY CHANNEL8 = 0x8 PMOS THIN CHANNEL9 = 0x9 NMOS THIN CHANNEL10 = 0xA NMOS THICK CHANNEL11 = 0xB PMOS THICK CHANNEL12 = 0xC CHANNEL13 = 0xD CHANNEL14 = 0xE CHANNEL15 = 0xF
15:10	**RSRVD2**	RO	0x00	Reserved

Table 931. HW_LRADC_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9	**TEMP_SENSOR_IENABLE1**	RW	0x0	Set this bit to one to enable the current source onto LRADC1. WARNING: Temperature sensor current source requires VDDIO greater than 2.9 V for correct operation. DISABLE = 0x0 Disable Temperature Sensor Current Source. ENABLE = 0x1 Enable Temperature Sensor Current Source.
8	**TEMP_SENSOR_IENABLE0**	RW	0x0	Set this bit to one to enable the current source onto LRADC0. WARNING: Temperature sensor current source requires VDDIO greater than 2.9 V for correct operation. DISABLE = 0x0 Disable Temperature Sensor Current Source. ENABLE = 0x1 Enable Temperature Sensor Current Source.
7:4	**TEMP_ISRC1**	RW	0x0	This four-bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC1. 300 = 0xF 300uA. 280 = 0xE 280uA. 260 = 0xD 260uA. 240 = 0xC 240uA. 220 = 0xB 220uA. 200 = 0xA 200uA. 180 = 0x9 180uA. 160 = 0x8 160uA. 140 = 0x7 140uA. 120 = 0x6 120uA. 100 = 0x5 100uA. 80 = 0x4 80uA. 60 = 0x3 60uA. 40 = 0x2 40uA. 20 = 0x1 20uA. ZERO = 0x0 0uA.
3:0	**TEMP_ISRC0**	RW	0x0	This four-bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC0. 300 = 0xF 300uA. 280 = 0xE 280uA. 260 = 0xD 260uA. 240 = 0xC 240uA. 220 = 0xB 220uA. 200 = 0xA 200uA. 180 = 0x9 180uA. 160 = 0x8 160uA. 140 = 0x7 140uA. 120 = 0x6 120uA. 100 = 0x5 100uA. 80 = 0x4 80uA. 60 = 0x3 60uA. 40 = 0x2 40uA. 20 = 0x1 20uA. ZERO = 0x0 0uA.

DESCRIPTION:

The LRADC Control Register 2 provides control over the shared eight-channel LRADC converter. It allows software to independently schedule conversion cycles on any number of any size subsets of the eight channels. In addition, it allows software to manage the interrupt reporting for the channel conversion sets.

EXAMPLE:

```
BW_LRADC_CTRL2_TEMP_SENSOR_IENABLE1(BV_LRADC_CTRL2_TEMP_SENSOR_IENABLE1__DISABLE);
```

29.5.4. LRADC Control Register 3 Description

The LRADC Control Register 3 specifies the voltages at which a touch-detect interrupt is generated.

HW_LRADC_CTRL3 0x80050030
HW_LRADC_CTRL3_SET 0x80050034
HW_LRADC_CTRL3_CLR 0x80050038
HW_LRADC_CTRL3_TOG 0x8005003C

Table 932. HW_LRADC_CTRL3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD5						DISCARD		FORCE_ANALOG_PWUP	FORCE_ANALOG_PWDN	FORCE_PWD40UA_PWUP	FORCE_PWD40UA_PWDN	RSRVD4		VDD_FILTER		RSRVD3		ADD_CAP2INPUTS		RSRVD2		CYCLE_TIME		RSRVD1		HIGH_TIME		REMOVE_CFILT	SHORT_RFILT	DELAY_CLOCK	INVERT_CLOCK

Table 933. HW_LRADC_CTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD5**	RO	0x0	Reserved
25:24	**DISCARD**	RW	0x0	This bit field specifies the number of samples to discard whenever the LRADC analog is first powered up. 00= Discard first 3 samples. 01= Discard first sample. 10= Discard first 2 samples. 11= Discard first 3 samples. 1_SAMPLE = 0x1 Discard first sample before first capture. 2_SAMPLES = 0x2 Discard 2 samples before first capture. 3_SAMPLES = 0x3 Discard 3 samples before first capture.
23	**FORCE_ANALOG_PWUP**	RW	0x0	Set this bit to zero for normal operation. Setting it to one forces an analog power up, regardless of where the digital state machine may be. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
22	**FORCE_ANALOG_PWDN**	RW	0x0	Set this bit to zero for normal operation. Setting it to one forces an analog power down, regardless of where the digital state machine may be. ON = 0x0 Turn it on. OFF = 0x1 Turn it off.

Table 933. HW_LRADC_CTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21	**FORCE_PWD40UA_PWUP**	RW	0x0	Set to one to force a power up of the 40 microAmp bias current source. Set to zero for normal operation. This bit was added due to concerns that turning the 40uA bias current on and off at audible frequency rate, say 1KSPS, might cause audible tones to appear during playback. Lab experiments show that this concern was unfounded. Therefore, this bit should not be set without approval from SigmaTel. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
20	**FORCE_PWD40UA_PWDN**	RW	0x0	Set to one to force a power down of the 40 microAmp bias current source. Set to zero for normal operation. ON = 0x0 Turn it on. OFF = 0x1 Turn it off.
19:18	**RSRVD4**	RO	0x0	Reserved
17:16	**VDD_FILTER**	RW	0x0	This bit field controls a test function that adds additional series resistance to the LRADC VDD path to further filter the VDD noise. 00= 0 ohms 0dB 01= 100 ohms 5dB 10= 250 ohms 9.5dB 11= 500 ohms 13.5dB 0OHMS = 0x0 0 ohms additional VDD filter. 100OHMS = 0x1 100 ohms additional VDD filter. 250OHMS = 0x2 250 ohms additional VDD filter. 5000OHMS = 0x3 500 ohms additional VDD filter.
15:14	**RSRVD3**	RO	0x0	Reserved
13:12	**ADD_CAP2INPUTS**	RW	0x0	This bit field controls a test function that adds additional capacitance to the filter inputs. 00= 0 pf additional cap 01= 0.5 pf 10= 1.0 pf 11= 2.5 pf 0PF = 0x0 0 pf additional capacitance. 0_5PF = 0x1 0.5 pf additional capacitance. 1_0PF = 0x2 1.0 pf additional capacitance. 2_5PF = 0x3 2.5 pf additional capacitance.
11:10	**RSRVD2**	RO	0x0	Reserved
9:8	**CYCLE_TIME**	RW	0x0	Changes the LRADC clock frequency. Note: The sample rate is one-thirteenth of the frequency selected here. 00= 6 MHz 01= 4 MHz 10= 3 MHz 11= 2 MHz 6MHZ = 0x0 6-MHz clock. 4MHZ = 0x1 4-MHz clock. 3MHZ = 0x2 3-MHz clock. 2MHZ = 0x3 2-MHz clock.
7:6	**RSRVD1**	RO	0x0	Reserved

Table 933. HW_LRADC_CTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
5:4	**HIGH_TIME**	RW	0x0	Changes the duty cycle (time high) for the LRADC clock. 00= 41.66 ns 01= 83.33 ns 10= 125 ns 11= 250 ns 42NS = 0x0 Duty cycle high time to 41.66 ns. 83NS = 0x1 Duty cycle high time to 83.33 ns. 125NS = 0x2 Duty cycle high time to 125 ns. 250NS = 0x3 Duty cycle high time to 250 ns.
3	**REMOVE_CFILT**	RW	0x0	Changes filtering on the output of the D/A converter. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
2	**SHORT_RFILT**	RW	0x0	Changes filtering on the output of the D/A converter. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
1	**DELAY_CLOCK**	RW	0x0	Set this bit to one to delay the 24-MHz clock used in the LRADC even further away from the predominant rising edge used within the digital section. The delay inserted is approximately 400 ps. NORMAL = 0x0 Normal operation, that is no delay. DELAYED = 0x1 Delay the clock.
0	**INVERT_CLOCK**	RW	0x0	Set this bit field to one to invert the 24-MHz clock where it comes into the LRADC analog section. This moves it away from the predominant digital rising edge. Setting this bit to one causes the A/D converter to run from the negative edge of the divided clock, effectively shifting the conversion point away from the edge used by the DCDC converter. NORMAL = 0x0 Run the clock in normal (not inverted) mode. INVERT = 0x1 Invert the clock.

DESCRIPTION:

The LRADC touch detect control and status register controls the voltage at which a touch detection interrupt is generated. This register also contains the interrupt request status bit and enable bit for the touch detection interrupt request to the CPU's IRQ interrupt input.

EXAMPLE:

```
BW_LRADC_CTRL3_HIGH_TIME(BV_LRADC_CTRL3_HIGH_TIME__83NS);
BW_LRADC_CTRL3_INVERT_CLOCK(BV_LRADC_CTRL3_INVERT_CLOCK__NORMAL);
```

29.5.5. LRADC Status Register Description

The LRADC Status Register returns various read-only status bit field values.

HW_LRADC_STATUS 0x80050040
HW_LRADC_STATUS_SET 0x80050044
HW_LRADC_STATUS_CLR 0x80050048
HW_LRADC_STATUS_TOG 0x8005004C

Table 934. HW_LRADC_STATUS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3					TEMP1_PRESENT	TEMP0_PRESENT	TOUCH_PANEL_PRESENT	CHANNEL7_PRESENT	CHANNEL6_PRESENT	CHANNEL5_PRESENT	CHANNEL4_PRESENT	CHANNEL3_PRESENT	CHANNEL2_PRESENT	CHANNEL1_PRESENT	CHANNEL0_PRESENT	RSRVD2															TOUCH_DETECT_RAW

Table 935. HW_LRADC_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	**RSRVD3**	RO	0x0	Reserved
26	**TEMP1_PRESENT**	RO	0x1	This read-only bit returns a one when the temperature sensor 1 current source is present on the chip.
25	**TEMP0_PRESENT**	RO	0x1	This read-only bit returns a one when the temperature sensor 0 current source is present on the chip.
24	**TOUCH_PANEL_PRESENT**	RO	0x1	This read-only bit returns a one when the touch panel controller function is present on the chip.
23	**CHANNEL7_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 7 converter function is present on the chip.
22	**CHANNEL6_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 6 converter function is present on the chip.
21	**CHANNEL5_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 5 converter function is present on the chip.
20	**CHANNEL4_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 4 converter function is present on the chip.
19	**CHANNEL3_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 3 converter function is present on the chip.
18	**CHANNEL2_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 2 converter function is present on the chip.
17	**CHANNEL1_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 1 converter function is present on the chip.
16	**CHANNEL0_PRESENT**	RO	0x1	This read-only bit returns a one when the LRADC channel 0 converter function is present on the chip.
15:1	**RSRVD2**	RO	0x0	Reserved
0	**TOUCH_DETECT_RAW**	RO	0x0	This read-only bit shows the status of the Touch Detect Comparator in the analog section. OPEN = 0x0 No contact, i.e., open connection. HIT = 0x1 Someone is touching the panel.

DESCRIPTION:

The status register returns the value of a number of status bit fields.

EXAMPLE:

```
if(HW_LRADC_STATUS.TOUCH_DETECT_RAW == BV_LRADC_STATUS_TOUCH_DETECT_RAW__HIT){
// Then something is touching the screen.
}
```

29.5.6. LRADC 0 Result Register Description

The LRADC 0 Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel 0.

HW_LRADC_CH0 0x80050050
HW_LRADC_CH0_SET 0x80050054
HW_LRADC_CH0_CLR 0x80050058
HW_LRADC_CH0_TOG 0x8005005C

Table 936. HW_LRADC_CH0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 937. HW_LRADC_CH0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18-bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 0 Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(0).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(0, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC0_IRQ != BV_LRADC_CTRL1_LRADC0_IRQ__PENDING)
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(0).VALUE / 5;
```

29.5.7. LRADC 1 Result Register Description

The LRADC 1 Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel 1.

HW_LRADC_CH1 0x80050060
HW_LRADC_CH1_SET 0x80050064
HW_LRADC_CH1_CLR 0x80050068
HW_LRADC_CH1_TOG 0x8005006C

Table 938. HW_LRADC_CH1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 939. HW_LRADC_CH1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This read-only bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18-bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.

Table 939. HW_LRADC_CH1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 1 Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(1).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(1, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC1_IRQ != BV_LRADC_CTRL1_LRADC1_IRQ__PENDING)
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(1).VALUE / 5;
```

29.5.8. LRADC 2 Result Register Description

The LRADC 2 Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel 2.

HW_LRADC_CH2 0x80050070
HW_LRADC_CH2_SET 0x80050074
HW_LRADC_CH2_CLR 0x80050078
HW_LRADC_CH2_TOG 0x8005007C

Table 940. HW_LRADC_CH2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 941. HW_LRADC_CH2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This read-only bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18-bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 2 Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(2).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(2, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC2_IRQ != BV_LRADC_CTRL1_LRADC2_IRQ__PENDING)
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(2).VALUE / 5;
```

29.5.9. LRADC 3 Result Register Description

The LRADC 3 Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel 3.

HW_LRADC_CH3 0x80050080
HW_LRADC_CH3_SET 0x80050084
HW_LRADC_CH3_CLR 0x80050088
HW_LRADC_CH3_TOG 0x8005008C

Table 942. HW_LRADC_CH3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 943. HW_LRADC_CH3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This read-only bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18-bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 3 Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(3).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(3, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC3_IRQ != BV_LRADC_CTRL1_LRADC3_IRQ__PENDING)
```

```
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(3).VALUE / 5;
```

29.5.10. LRADC 4 Result Register Description

The LRADC 4 Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel 4.

HW_LRADC_CH4 0x80050090
HW_LRADC_CH4_SET 0x80050094
HW_LRADC_CH4_CLR 0x80050098
HW_LRADC_CH4_TOG 0x8005009C

Table 944. HW_LRADC_CH4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 945. HW_LRADC_CH4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This read-only bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18-bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 4 Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(4).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(4, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC4_IRQ != BV_LRADC_CTRL1_LRADC4_IRQ__PENDING)
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(4).VALUE / 5;
```

29.5.11. LRADC 5 Result Register Description

The LRADC 5 Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel five.

HW_LRADC_CH5 0x800500A0
HW_LRADC_CH5_SET 0x800500A4
HW_LRADC_CH5_CLR 0x800500A8
HW_LRADC_CH5_TOG 0x800500AC

Table 946. HW_LRADC_CH5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 947. HW_LRADC_CH5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This read-only bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18-bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.

Table 947. HW_LRADC_CH5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 5 Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(5).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(5, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC5_IRQ != BV_LRADC_CTRL1_LRADC5_IRQ__PENDING)
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(5).VALUE / 5;
```

29.5.12. LRADC 6 (VddIO) Result Register Description

The LRADC 6 (VddIO) Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel 6 (VddIO).

HW_LRADC_CH6 0x800500B0
HW_LRADC_CH6_SET 0x800500B4
HW_LRADC_CH6_CLR 0x800500B8
HW_LRADC_CH6_TOG 0x800500BC

Table 948. HW_LRADC_CH6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE																	

Table 949. HW_LRADC_CH6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This read-only bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**RSRVD2**	RO	0x0	Reserved
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18-bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 6 (VddIO) Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(6).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(6, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC6_IRQ != BV_LRADC_CTRL1_LRADC6_IRQ__PENDING)
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(6).VALUE / 5;
```

29.5.13. LRADC 7 (BATT) Result Register Description

The LRADC 7 (BATT) Result Register returns the 12-bit result for low-resolution analog-to-digital converter channel 7 (BATT).

HW_LRADC_CH7 0x800500C0
HW_LRADC_CH7_SET 0x800500C4
HW_LRADC_CH7_CLR 0x800500C8
HW_LRADC_CH7_TOG 0x800500CC

Table 950. HW_LRADC_CH7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TOGGLE	TESTMODE_TOGGLE	ACCUMULATE	NUM_SAMPLES					RSRVD1								VALUE															

Table 951. HW_LRADC_CH7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**TOGGLE**	RW	0x0	This read-only bit toggles at every completed conversion, so that software can detect a missed or duplicated sample.
30	**TESTMODE_TOGGLE**	RO	0x0	This read-only bit toggles at every completed conversion of interest in test mode so software can synchornize to the desired sample. When the test mode count is loaded with a value of 7, this will toggle every eighth conversion on channel 7. If testmode operation for channel 5 and or 6 are set then the sample rate will be lower for channel 7.
29	**ACCUMULATE**	RW	0x0	Set this bit to one to add successive samples to the 18 bit accumulator.
28:24	**NUM_SAMPLES**	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23:18	**RSRVD1**	RO	0x000	Reserved
17:0	**VALUE**	RW	0x0000	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enabled, this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256 K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

DESCRIPTION:

The LRADC 7 (BATT) Result Register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N

times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC delay channels.

EXAMPLE:

```
if (HW_LRADC_CHn(7).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;
HW_LRADC_CHn_WR(7, (BF_LRADC_CHn_ACCUMULATE(1) | // Enable accumulation mode.
BF_LRADC_CHn_NUM_SAMPLES(5) | // Set samples to five.
BF_LRADC_CHn_VALUE(0) ) ); // Clear accumulator.
// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)
while (HW_LRADC_CTRL1.LRADC7_IRQ != BV_LRADC_CTRL1_LRADC7_IRQ__PENDING)
{
// Wait for interrupt.
}
channelAverage = HW_LRADC_CHn(7).VALUE / 5;
```

29.5.14. LRADC Scheduling Delay 0 Register Description

The LRADC Scheduling Delay 0 Register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more scheduling delay channels.

HW_LRADC_DELAY0 0x800500D0
HW_LRADC_DELAY0_SET 0x800500D4
HW_LRADC_DELAY0_CLR 0x800500D8
HW_LRADC_DELAY0_TOG 0x800500DC

Table 952. HW_LRADC_DELAY0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_DELAYS				LOOP_COUNT					DELAY										

Table 953. HW_LRADC_DELAY0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**TRIGGER_LRADCS**	RW	0x00	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The hardware accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23:21	**RSRVD2**	RO	0x0	Reserved
20	**KICK**	RW	0x0	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.

Table 953. HW_LRADC_DELAY0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:16	**TRIGGER_DELAYS**	RW	0x0	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15:11	**LOOP_COUNT**	RW	0x00	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels. Note setting the loop count to 0x01 will yield two conversions.
10:0	**DELAY**	RW	0x000	This 11-bit field counts down to zero. At zero, it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single even. This counter operates on a 2-kHz clock derived from the crystal clock.

DESCRIPTION:

The RADC Scheduling Delay 0 Register provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2-kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE:

```
HW_LRADC_DELAYn_WR(0, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05) | // LRADC channel 0 and 2
BF_LRADC_DELAYn_KICK(1) | // Start the Delay channel
BF_LRADC_DELAYn_TRIGGER_DELAYS(0x1) | // restart delay channel 0 each time
BF_LRADC_DELAYn_DELAY(0x0E45) ) ); // delay 3653 periods of 2 kHz clock
// ... do other things until the triggered LRADC channels report an interrupt.
```

29.5.15. LRADC Scheduling Delay 1 Register Description

The LRADC Scheduling Delay 1 Register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more scheduling delay channels .

HW_LRADC_DELAY1 0x800500E0
HW_LRADC_DELAY1_SET 0x800500E4
HW_LRADC_DELAY1_CLR 0x800500E8
HW_LRADC_DELAY1_TOG 0x800500EC

Table 954. HW_LRADC_DELAY1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_DELAYS				LOOP_COUNT					DELAY										

Table 955. HW_LRADC_DELAY1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**TRIGGER_LRADCS**	RW	0x00	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23:21	**RSRVD2**	RO	0x0	Reserved
20	**KICK**	RW	0x0	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19:16	**TRIGGER_DELAYS**	RW	0x0	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15:11	**LOOP_COUNT**	RW	0x00	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels.
10:0	**DELAY**	RW	0x000	This 11-bit field counts down to zero. At zero, it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single even. This counter operates on a 2-kHz clock derived from the crystal clock.

DESCRIPTION:

The LRADC Scheduling Delay 1 Register provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2-kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE:

```
HW_LRADC_DELAYn_WR(1, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05) | // LRADC channel 0 and 2
BF_LRADC_DELAYn_KICK(1) | // Start the Delay channel
BF_LRADC_DELAYn_TRIGGER_DELAYS(0x2) | // restart delay channel 1 each time
BF_LRADC_DELAYn_DELAY(0x0E45) ) ); // delay 3653 periods of 2 kHz clock
// ... do other things until the triggered LRADC channels report an interrupt.
```

29.5.16. LRADC Scheduling Delay 2 Register Description

The LRADC Scheduling Delay 2 Register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more scheduling delay channels .

HW_LRADC_DELAY2 0x800500F0
HW_LRADC_DELAY2_SET 0x800500F4
HW_LRADC_DELAY2_CLR 0x800500F8
HW_LRADC_DELAY2_TOG 0x800500FC

Table 956. HW_LRADC_DELAY2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_DELAYS				LOOP_COUNT					DELAY										

Table 957. HW_LRADC_DELAY2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**TRIGGER_LRADCS**	RW	0x00	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23:21	**RSRVD2**	RO	0x0	Reserved

Table 957. HW_LRADC_DELAY2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	**KICK**	RW	0x0	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19:16	**TRIGGER_DELAYS**	RW	0x0	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15:11	**LOOP_COUNT**	RW	0x00	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels.
10:0	**DELAY**	RW	0x000	This 11-bit field counts down to zero. At zero, it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single even. This counter operates on a 2-kHz clock derived from the crystal clock.

DESCRIPTION:

The LRADC Scheduling Delay 2 Register provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2-kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE:

```
HW_LRADC_DELAYn_WR(2, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05) | // LRADC channel 0 and 2
BF_LRADC_DELAYn_KICK(1) | // Start the Delay channel
BF_LRADC_DELAYn_TRIGGER_DELAYS(0x4) | // restart delay channel 2 each time
BF_LRADC_DELAYn_DELAY(0x0E45) ) ); // delay 3653 periods of 2 kHz clock
// ... do other things until the triggered LRADC channels report an interrupt.
```

29.5.17. LRADC Scheduling Delay 3 Register Description

The LRADC Scheduling Delay 3 Register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more scheduling delay channels .

HW_LRADC_DELAY3 0x80050100
HW_LRADC_DELAY3_SET 0x80050104
HW_LRADC_DELAY3_CLR 0x80050108
HW_LRADC_DELAY3_TOG 0x8005010C

Table 958. HW_LRADC_DELAY3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_DELAYS				LOOP_COUNT					DELAY										

Table 959. HW_LRADC_DELAY3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**TRIGGER_LRADCS**	RW	0x00	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23:21	**RSRVD2**	RO	0x0	Reserved
20	**KICK**	RW	0x0	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19:16	**TRIGGER_DELAYS**	RW	0x0	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15:11	**LOOP_COUNT**	RW	0x00	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels. ERRATA: TA1 and TA2 silicon revisions do not correctly support the LOOP_COUNT field, do not use.
10:0	**DELAY**	RW	0x000	This 11-bit field counts down to zero. At zero, it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single even. This counter operates on a 2-kHz clock derived from crystal clock.

DESCRIPTION:

The LRADC Scheduling Delay 3 Register provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2-kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE:

```
HW_LRADC_DELAYn_WR(3, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05) | // LRADC channel 0 and 2
BF_LRADC_DELAYn_KICK(1) | // Start the Delay channel
BF_LRADC_DELAYn_TRIGGER_DELAYS(0x8) | // restart delay channel 3 each time
BF_LRADC_DELAYn_DELAY(0x0E45) ) ); // delay 3653 periods of 2 kHz clock
// ... do other things until the triggered LRADC channels report an interrupt.
```

29.5.18. LRADC Debug Register 0 Description

The LRADC Debug Register 0 provides read-only access to various internal states and other debug information.

HW_LRADC_DEBUG0 0x80050110
HW_LRADC_DEBUG0_SET 0x80050114
HW_LRADC_DEBUG0_CLR 0x80050118
HW_LRADC_DEBUG0_TOG 0x8005011C

Table 960. HW_LRADC_DEBUG0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
READONLY																RSRVD1				STATE											

Table 961. HW_LRADC_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**READONLY**	RO	0x4321	LRADC internal state machine current state.
15:12	**RSRVD1**	RO	0x0	Reserved
11:0	**STATE**	RO	0x0	LRADC internal state machine current state.

DESCRIPTION:

The LRADC Debug Register 0 contains read-only diagnostic information regarding the internal state machine. This only used in debugging.

EXAMPLE:

```
if (HW_LRADC_DEBUG0.STATE == 0X33) {} // some action based on this state.
```

29.5.19. LRADC Debug Register 1 Description

The LRADC Debug Register 1 provides read-only access to various internal states and other debug information.

HW_LRADC_DEBUG1 0x80050120
HW_LRADC_DEBUG1_SET 0x80050124
HW_LRADC_DEBUG1_CLR 0x80050128
HW_LRADC_DEBUG1_TOG 0x8005012C

Table 962. HW_LRADC_DEBUG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3								REQUEST								RSRVD2			TESTMODE_COUNT					RSRVD1					TESTMODE6	TESTMODE5	TESTMODE

Table 963. HW_LRADC_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD3**	RO	0x0	Reserved
23:16	**REQUEST**	RO	0x0	LRADC internal request register.
15:13	**RSRVD2**	RO	0x0	Reserved
12:8	**TESTMODE_COUNT**	RW	0x0	When in test mode, the value in this register will be loaded in to a counter which is decremented upon each Channel 7 conversion. When that counter decrements to zero, the HW_LRADC_CH7_TESTMODE_TOGGLE field will be toggled, indicating that the conversion value of interest is available in the HW_LRADC_CH7_VALUE bit field.
7:3	**RSRVD1**	RO	0x0	Reserved
2	**TESTMODE6**	RW	0x0	Force dummy conversion cycles on channel 6 during test mode. NORMAL = 0x0 Normal operation. TEST = 0x1 Put it in test mode, i.e., continuously sample channel 6.
1	**TESTMODE5**	RW	0x0	Force dummy conversion cycles on channel 5 during test mode. NORMAL = 0x0 Normal operation. TEST = 0x1 Put it in test mode, i.e., continuously sample channel 5.
0	**TESTMODE**	RW	0x0	Place the LRADC in a special test mode in which the analog section is free-running at its clock rate. LRADC_CH7 result is continuously updated every N conversions from the analog source selected in CTRL2, where N is determined by TESTMODE_COUNT. NORMAL = 0x0 Normal operation. TEST = 0x1 Put it in test mode, i.e., continuously sample channel 7.

DESCRIPTION:

The LRADC Debug Register 1 provides read-only diagnostic information and control over the test modes of LRADC channels 5, 6, and 7. This is only used in debugging the LRADC.

EXAMPLE:

```
BW_LRADC_DEBUG1_TESTMODE(BV_LRADC_DEBUG1_TESTMODE__TEST);
```

29.5.20. LRADC Battery Conversion Register Description

The LRADC Battery Conversion Register provides access to the battery voltage scale multiplier.

HW_LRADC_CONVERSION 0x80050130
HW_LRADC_CONVERSION_SET 0x80050134
HW_LRADC_CONVERSION_CLR 0x80050138
HW_LRADC_CONVERSION_TOG 0x8005013C

Table 964. HW_LRADC_CONVERSION

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3											AUTOMATIC	RSRVD2		SCALE_FACTOR		RSRVD1						SCALED_BATT_VOLTAGE									

Table 965. HW_LRADC_CONVERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	**RSRVD3**	RO	0x0	Reserved
20	**AUTOMATIC**	RW	0x0	Control the automatic update mode of the BATT_VAL bit field in the HW_POWER_MONITOR register. DISABLE = 0x0 No automatic update of the scaled value. ENABLE = 0x1 Automatically compute the scaled battery voltage each time an LRADC Channel 7 (BATT) conversion takes place.
19:18	**RSRVD2**	RO	0x0	Reserved
17:16	**SCALE_FACTOR**	RW	0x0	Scale factors of 29/512, 29/256 or 29/128 are selected here. NIMH = 0x0 Single NiMH Battery operation, 29/512. DUAL_NIMH = 0x1 Two NiMH Battery operation, 29/256. LI_ION = 0x2 Lithium Ion Battery operation, 29/128. ALT_LI_ION = 0x3 Lithium Ion Battery operation, 29/128.
15:10	**RSRVD1**	RO	0x0	Reserved
9:0	**SCALED_BATT_VOLTAGE**	RW	0x80	LRADC Battery Voltage Divided by approximately 17.708. The actual scale factor is (battery voltage) times 29 divided by 512, 256, or 128.

DESCRIPTION:

The LRADC Battery Conversion Register controls the voltage scaling multiplier that is used to multiply the LRADC battery voltage by 29 divided by 512 for NiMH, battery voltage times 29 divided by 256 for dual NiMH and battery voltage times 29 divided by 128 for Lithium Ion batteries.

EXAMPLE:

```
HW_LRADC_CONVERSION.AUTOMATIC = 1;
```

LRADC XML Revision: 1.49

30. MEMORY COPY DEVICE

This chapter describes the memory copy device included on the STMP36xx. along with programming examples. Programmable registers are described in Section 30.4.

30.1. Overview

The memory copy or MEMCPY APB device provides a path from a source DMA channel to a destination DMA channel, allowing blocks of data located on any slave on the AHB to be copied to any slave on the AHB. In particular, it can be used to copy data from SDRAM to on-chip SRAM or SRAM to SDRAM without CPU involvement. It is also used to copy blocks of data and instructions from on-chip ROM to on-chip RAM. It is also used by the overlay/paging software to copy pages from SDRAM to on-chip RAM when page faults are detected. Figure 132 shows a block diagram of the memory copy device included on the STMP36xx.

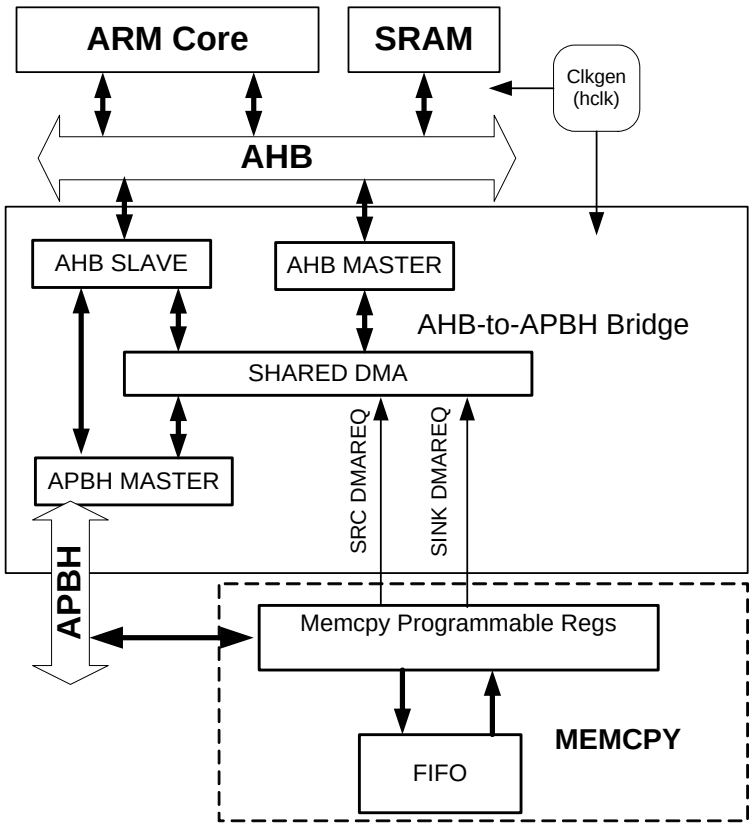


Figure 132. Memory Copy Device Block Diagram

The MEMCPY device is removed from soft reset and has its clocks enabled. Then, both a source and destination DMA are initialized. Either the source or destination DMA command structure contains one PIO register value to write to the MEMCPY device. As a result, the MEMCPY device copies all of the bytes from the source to the destination. Scatter and/or gather operations can be implemented by chaining multiple source DMA commands or multiple destination DMA commands together.

A memory-fill operation can be created by chaining a source DMA descriptor back on itself, so that the MEMCPY device copies the same source values over and over to the destination.

If the user wants to program APBH DMA Channel 2 (the MEMCPY source channel) to utilize a different count value than APBH DMA Channel 3 (the MEMCPY destination channel), the count value for channel 2 needs to be a multiple of 4 bytes.

If the user programs the Channel 2 and Channel 3 count values to be the same value, then any byte count value can be used.

Not following these guidelines can lead to expected behavior.

30.2. Programming Examples

30.2.1. Block Copy

```
//////////////////////////////////////////////////////
// the two descriptors need to be placed in non-cached memory
//////////////////////////////////////////////////////

static reg32_t  source_descriptor[4];
static reg32_t  destination_descriptor[3];

//////////////////////////////////////////////////////
//  block copy routine using memcpy device
//////////////////////////////////////////////////////
unsigned block_copy(const void* source, void* destination, unsigned bytes)
{
    const unsigned TIMEOUT = 100;
    unsigned retries;

    // Setup source descriptor.
    source_descriptor[0] = 0;
    source_descriptor[1] = (BF_APBH_CHn_CMD_XFER_COUNT(bytes) |
                            BF_APBH_CHn_CMD_CMDWORDS(1) |
                            BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ));
    source_descriptor[2] = (reg32_t) source;
    source_descriptor[3] = BF_MEMCPY_CTRL_XFER_SIZE(bytes);

    // Setup destination descriptor.
    destination_descriptor[0] = 0;
    destination_descriptor[1] = (BF_APBH_CHn_CMD_XFER_COUNT(bytes) |
                                 BF_APBH_CHn_CMD_SEMAPHORE(1) |
                                 BV_FLD(APBH_CHn_CMD, COMMAND, DMA_WRITE));
    destination_descriptor[2] = (reg32_t) destination;

    // Reset both source and destination channels.
    HW_APBH_CTRL0_SET(BF_APBH_CTRL0_RESET_CHANNEL((1 << 3) | (1 << 2)));

    // Setup source and destination descriptor pointers.
    BF_WRn(APBH_CHn_NXTCMDAR, 2, CMD_ADDR, (reg32_t) source_descriptor);
    BF_WRn(APBH_CHn_NXTCMDAR, 3, CMD_ADDR, (reg32_t) destination_descriptor);

    // Start both source and destination channels by incrementing semaphore.
    BF_WRn(APBH_CHn_SEMA, 2, INCREMENT_SEMA, 1);
    BF_WRn(APBH_CHn_SEMA, 3, INCREMENT_SEMA, 1);

    // Poll for decrement of destination channel's semaphore.
    for (retries = 0; retries < TIMEOUT; retries++)
        if (!BF_RDn(APBH_CHn_SEMA, 3, PHORE))
            break;

    // Return false if timed out waiting for semaphore.
    if (retries == TIMEOUT)
        return 0;

    // Otherwise, return true; the block has been copied.
    return 1;
}
```

30.2.2. ROM Dot Data Copying and BSS Fill

The following sample code shows how the MEMCPY device can be used to copy the dot data section from ROM to RAM and how it can then be used to zero the BSS section.

```
////////////////////////////////////////////////////////
// the descriptors and DMA data buffer need to be placed
// in non-cached memory
////////////////////////////////////////////////////////

unsigned long fill_buffer[32] = {
                        0x00000000, 0x00000000, 0x00000000, 0x00000000,
                        0x00000000, 0x00000000, 0x00000000, 0x00000000
};

const reg32_t  source_descriptor_bss_fill[3] =
{
        (reg32_t) source_descriptor_bss_fill, // repeat this one over and over
        (reg32_t) (BF_APBH_CHn_CMD_XFER_COUNT(32) |
                   BF_APBH_CHn_CMD_CHAIN(1)       |// repeat
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ)),
        (reg32_t) fill_buffer;
};


const reg32_t  source_descriptor_dot_data[3] =
{
        (reg32_t) source_descriptor_bss_fill,
        (reg32_t) (BF_APBH_CHn_CMD_XFER_COUNT(bytes) |
                   BF_APBH_CHn_CMD_CHAIN(1)          |
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ)),
        (reg32_t) &dot_data_start_address
};

const reg32_t  destination_descriptor_bss_fill[4] =
{
        (reg32_t) 0x0,
        (reg32_t) (BF_APBH_CHn_CMD_XFER_COUNT(bytes) |
                   BF_APBH_CHn_CMD_SEMAPHORE(1)      |
                   BF_APBH_CHn_CMD_CMDWORDS(1)       |
                   BF_APBH_CHn_CMD_CHAIN(0)          |
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_WRITE)),
        (reg32_t) &bss_target_address_in_ram,
        (reg32_t) BF_MEMCPY_CTRL_XFER_SIZE(bytes)
};
const reg32_t  destination_descriptor_dot_data[4] =
{
        (reg32_t) destination_descriptor_bss_fill,
        (reg32_t) (BF_APBH_CHn_CMD_XFER_COUNT(bytes) |
                   BF_APBH_CHn_CMD_CMDWORDS(1)       |
                   BF_APBH_CHn_CMD_CHAIN(1)          |
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_WRITE)),
        (reg32_t) &dot_data_target_address_in_ram,
        (reg32_t) BF_MEMCPY_CTRL_XFER_SIZE(bytes)
};

////////////////////////////////////////////////////////
// ROM routine for copying dot data section from ROM to RAM
// and for zero filling the bss section.
////////////////////////////////////////////////////////
void ROM_dot_data_copy()
{
    unsigned retries;

    // remove soft reset from the memcpy device and start the clock
    HW_MEMCPY_CTRL_CLR(BM_MEMCPY_CTRL_SFTRST | BM_MEMCPY_CTRL_CLKGATE);

    // Reset both source and destination channels.
    HW_APBH_CTRL0_SET(BF_APBH_CTRL0_RESET_CHANNEL((1 << 3) | (1 << 2)));

    // Setup source and destination descriptor pointers.
    BF_WRn(APBH_CHn_NXTCMDAR, 2, CMD_ADDR, (reg32_t) source_descriptor_dot_data);
    BF_WRn(APBH_CHn_NXTCMDAR, 3, CMD_ADDR, (reg32_t) destination_descriptor_dot_data);
```

```
    // Start both source and destination channels by incrementing semaphore.
    BF_WRn(APBH_CHn_SEMA, 2, INCREMENT_SEMA, 1);
    BF_WRn(APBH_CHn_SEMA, 3, INCREMENT_SEMA, 1);

    // return and do other things while the various memory blocks are copied

}
int ROM_wait4_dot_data_copy()

    // Poll for decrement of destination channel's semaphore.
    for (retries = 0; retries < TIMEOUT; retries++)
        if (!BF_RDn(APBH_CHn_SEMA, 3, PHORE))
            break;

    // Return false if timed out waiting for semaphore.
    if (retries == TIMEOUT)
        return 0;

    // Otherwise, return true; the block has been copied.
    return 1;
}
```

30.3. Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Section 33.4.10, “Correct Way to Soft Reset a Block” on page 805 for additional information on using the SFTRST and CLKGATE bit fields.

30.4. Programmable Registers

The following registers are available for programmer access and control of the memory copy device.

30.4.1. Memory Copy Device Control and Status Register Description

The MEMCPY Control and Status Register specifies the reset state, and the interrupt and control information for the memory copy device.

HW_MEMCPY_CTRL 0x80014000
HW_MEMCPY_CTRL_SET 0x80014004
HW_MEMCPY_CTRL_CLR 0x80014008
HW_MEMCPY_CTRL_TOG 0x8001400C

Table 966. HW_MEMCPY_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SFTRST	CLKGATE	PRESENT	RSRVD1												BURST	XFER_SIZE															

Table 967. HW_MEMCPY_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**SFTRST**	RW	0x1	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. RUN = 0x0 Allow MEMCPY to operate normally. RESET = 0x1 Hold MEMCPY in reset.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block. RUN = 0x0 Allow MEMCPY to operate normally. NO_CLKS = 0x1 Do not clock MEMCPY gates in order to minimize power consumption.
29	**PRESENT**	RO	0x1	This read-only bit indicates that the memcpy function is present when it reads back a one. The memcpy function is not available on a device that returns a zero for this bit field. UNAVAILABLE = 0x0 MEMCPY is not present in this product. AVAILABLE = 0x1 MEMCPY is present in this product.
28:17	**RSRVD1**	RO	0x0	These bits always read back zero.
16	**BURST**	RW	0x0	When set, the MEMCPY performs 4-beat burst instead of the 1-beat burst.
15:0	**XFER_SIZE**	RW	0x0000	A write to this register sets the number of bytes to transfer. A read from this register reflects the number of bytes remaining for the DMA Read Channel to read from the MEMCPY device.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

30.4.2. MEMCPY Device DMA Read and Write Data Register Description

The MEMCPY Device DMA Read and Write Data Register is the target, for both source and destination DMA transfers. This register is backed by an eight-deep FIFO.

HW_MEMCPY_DATA 0x80014010
HW_MEMCPY_DATA_SET 0x80014014
HW_MEMCPY_DATA_CLR 0x80014018
HW_MEMCPY_DATA_TOG 0x8001401C

Table 968. HW_MEMCPY_DATA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DATA																															

Table 969. HW_MEMCPY_DATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	**DATA**	RW	0x00000000	The source DMA channel writes to this address. The destination DMA channel reads from this address.

DESCRIPTION:

Empty Description.

30.4.3. MEMCPY Device Debug Register Description

The MEMCPY Device Debug Register provides a diagnostic view into the internal state machine and states of the MEMCPY device.

HW_MEMCPY_DEBUG 0x80014020
HW_MEMCPY_DEBUG_SET 0x80014024
HW_MEMCPY_DEBUG_CLR 0x80014028
HW_MEMCPY_DEBUG_TOG 0x8001402C

Table 970. HW_MEMCPY_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD2	DST_END_CMD	DST_KICK	DST_DMA_REQ	RSVD1		SRC_KICK	SRC_DMA_REQ	RSVD0																				WRITE_STATE		READ_STATE	

Table 971. HW_MEMCPY_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSVD2**	RO	0x0	Reserved.
30	**DST_END_CMD**	RO	0x0	This bit reflects the state of the destination channel end command signal.
29	**DST_KICK**	RO	0x0	This bit reflects the state of the destination channel kick signal.
28	**DST_DMA_REQ**	RO	0x0	This bit reflects the state of the destination channel DMA request signal.
27:26	**RSVD1**	RO	0x0	Reserved.
25	**SRC_KICK**	RO	0x0	This bit reflects the state of the source channel kick signal.
24	**SRC_DMA_REQ**	RO	0x0	This bit reflects the state of the source channel DMA request signal.
23:4	**RSVD0**	RO	0x0	Reserved.
3:2	**WRITE_STATE**	RO	0x0	These bits reflect the state of the MEMCPY state machine.
1:0	**READ_STATE**	RO	0x0	These bits reflect the state of the MEMCPY state machine.

DESCRIPTION:

Empty Description.

MEMCPY XML Revision: 1.21

31. POWER SUPPLY

This chapter describes the power supply subsystem provided on the STMP36xx. It includes sections on the DC-DC converters, linear regulators, PSWITCH pin functions, battery monitor and charger, and silicon speed sensor. Programmable registers are described in Section 31.8.

31.1. Overview

The STMP36xx integrates a comprehensive power supply subsystem, including the following features.

- Two integrated DC-DC converters support 1-cell, 2-cell, and Li-Ion batteries
- Two linear regulators supply power directly from 5V.
- Linear battery charger for NiMH and Li-Ion cells.
- Battery voltage and brownout monitor.
- Reset controller.
- System monitors for temperature and speed.
- Brownout detect for VDD, I/O and 5V supplies
- Generates USB-OTG 5V from Li-Ion battery (using PWM).
- Support for on-the-fly transitioning between 5V and battery power.
- Integrated FET switch to gate power to peripheral devices.

The STMP36xx power supply is designed to offer maximum flexibility and performance, while minimizing external component requirements. Figure 133 shows a functional block diagram of the power supply components including switching converters (DC-DC#1 and #2), two linear regulators, battery charge support, as well as battery monitoring, supply brownout detection, and silicon process/temperature sensors. This figure can be used to understand which register and status bits relate to which subsystems, but it is not intended to be a complete architecture description.

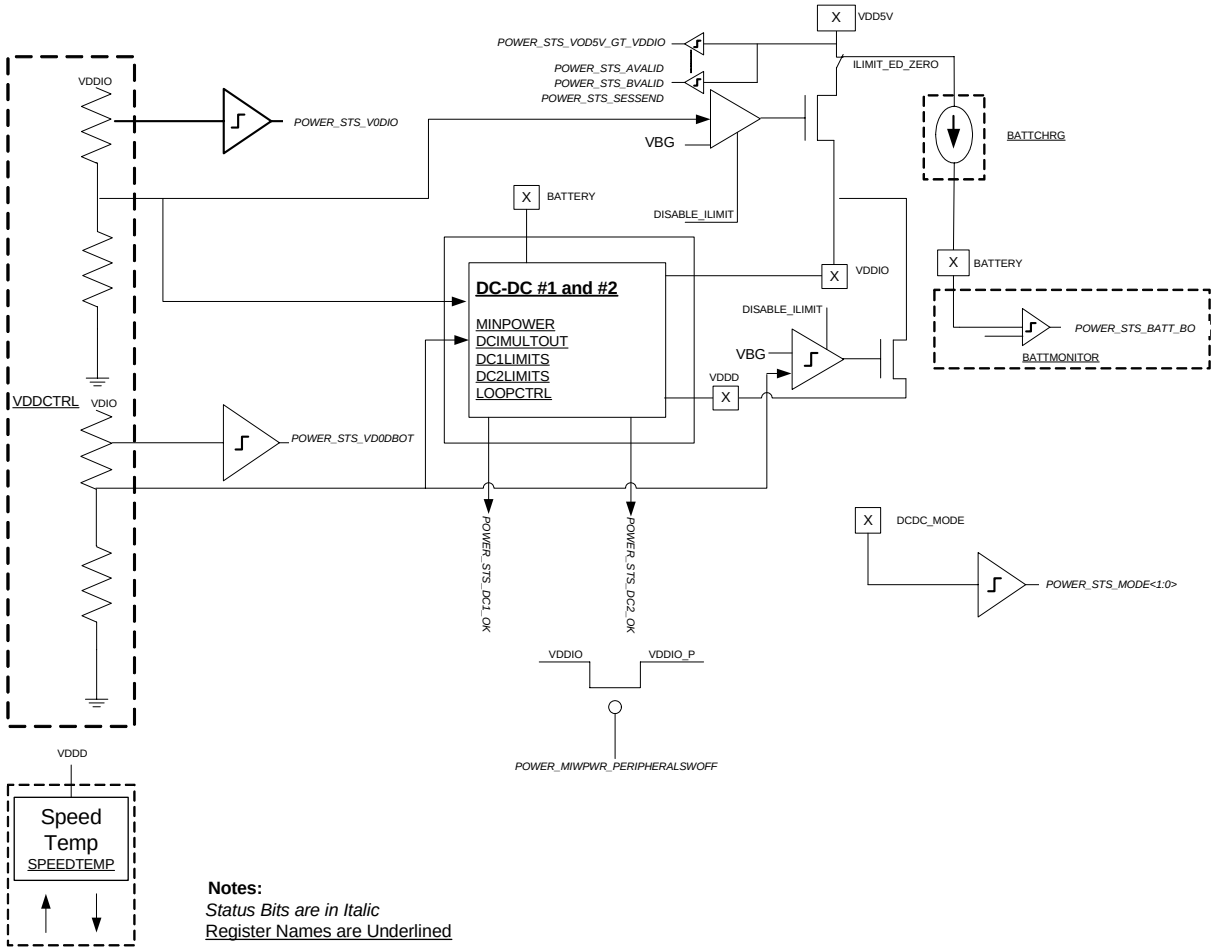


Figure 133. Power Supply Block Diagram

31.2. DC-DC Converters

The DC-DC converters efficiently scale battery voltage to the required supply voltages. The DC-DC converters include several advanced features:

- Flexible battery support
- Single Inductor modes for 1-cell and low-power Li-Ion
- Dual-inductor modes for 2-cell and high-power Li-Ion (HDD player)
- Programmable output voltages
- Programmable brownout detection thresholds
- Pulse Frequency Modulation (PFM) mode for low-current load operation

31.2.1. DC-DC Operating Modes

The DC-DC converter's operation is set up using a combination of hardware and software configuration. The basic operation, including battery type and inductor configuration, is set by hardware configuration during product design. Operating parameters, such as output voltage, are programmable after power-up.

The DC-DC mode pin determines which battery and inductor configuration is used. Table 972 lists the various DC-DC battery modes.

Table 972. DC-DC Battery Modes

DC-DC MODE PIN	MODE	BATT	VDD	VIO	INDUCTOR	MAX POWER	COMMENTS
Open	3	1-Cell 0.9V to 1.6V	Boost	Boost	1	200mW @ 1.0V	Low cost and area
270K	2	2-Cell 1.8 to 3.2V	Buck	Boost	2	400mW @ 1.8V	VIO must be higher than max battery voltage
120K	1	Li-Ion 2.9V to 4.2V	Buck	Buck/Boost	1	600mW @ 3.1V	Low cost and area
0-Ohm	0	Li-Ion 2.9V to 4.2V	Buck	Buck/Boost (169BGA) Buck (100QFP)	2	1600mW @ 3.1V	High power capability, battery > 3.1 to startup

The STMP36xx DC-DC converter has several advanced modes. It offers buck/boost modes that maintain the VIO voltage even as the battery voltage drops below it. This allows hard drive operation at 3.3 V while a Li-Ion battery discharges to 3.0 V, providing 10% better battery life than buck-only designs that must shut down when the battery reaches 3.3 V. The STMP36xx also offers modes for 1-cell and Li-Ion batteries that can supply both the VDD and VIO rails with a single inductor. The single-inductor operation allows the lowest cost and area for products that do not require high peak power.

31.2.2. DC-DC Operation

The STMP36xx DC-DC converter enables a low-power system and features programmable output voltages and control modes. Most products adjust VDD dynamically to provide the minimum voltage required for proper system operation. VIO is typically set once during system initialization and not changed during operation, because most LCD displays are not compatible with dynamic voltage adjustment.

31.2.2.1. Brownout/Error Detection

The power subsystem has several mechanisms active by default that safely return the device to the off state if any one of the following errors or brownouts occur:

- The crystal oscillator frequency is detected below a certain threshold—This threshold is process- and voltage-sensitive, but will always be between 100 kHz and 2 MHz. This feature can be disabled in the DCDC_CTRL field in the HW_RTC_PERSISTENT0 register.
- The battery voltage falls below the battery brownout level (field BRWNOUT_LVL in HW_POWER_BATTMONITOR)—This feature is disabled by clearing PWDN_BATTBRNOUT in the same register
- 5 V is detected, then removed—This feature is disabled by clearing HW_POWER_5VCTRL_PWDN_5VBRNOUT.

All three mechanisms are active by default to ensure that the device always has a valid transition to a known state in case the power source is unexpectedly removed before software has completed system configuration. Software can disable the func-

tionality of PWDN_5VBRNOUT and PWDN_BATTBRNOUT after system configuration is complete, as shown in Figure 134. System configuration generally includes setting up brownout detection thresholds on the supply voltages, battery, etc. to obtain the desired system operation as the battery or power source is depleted or removed.

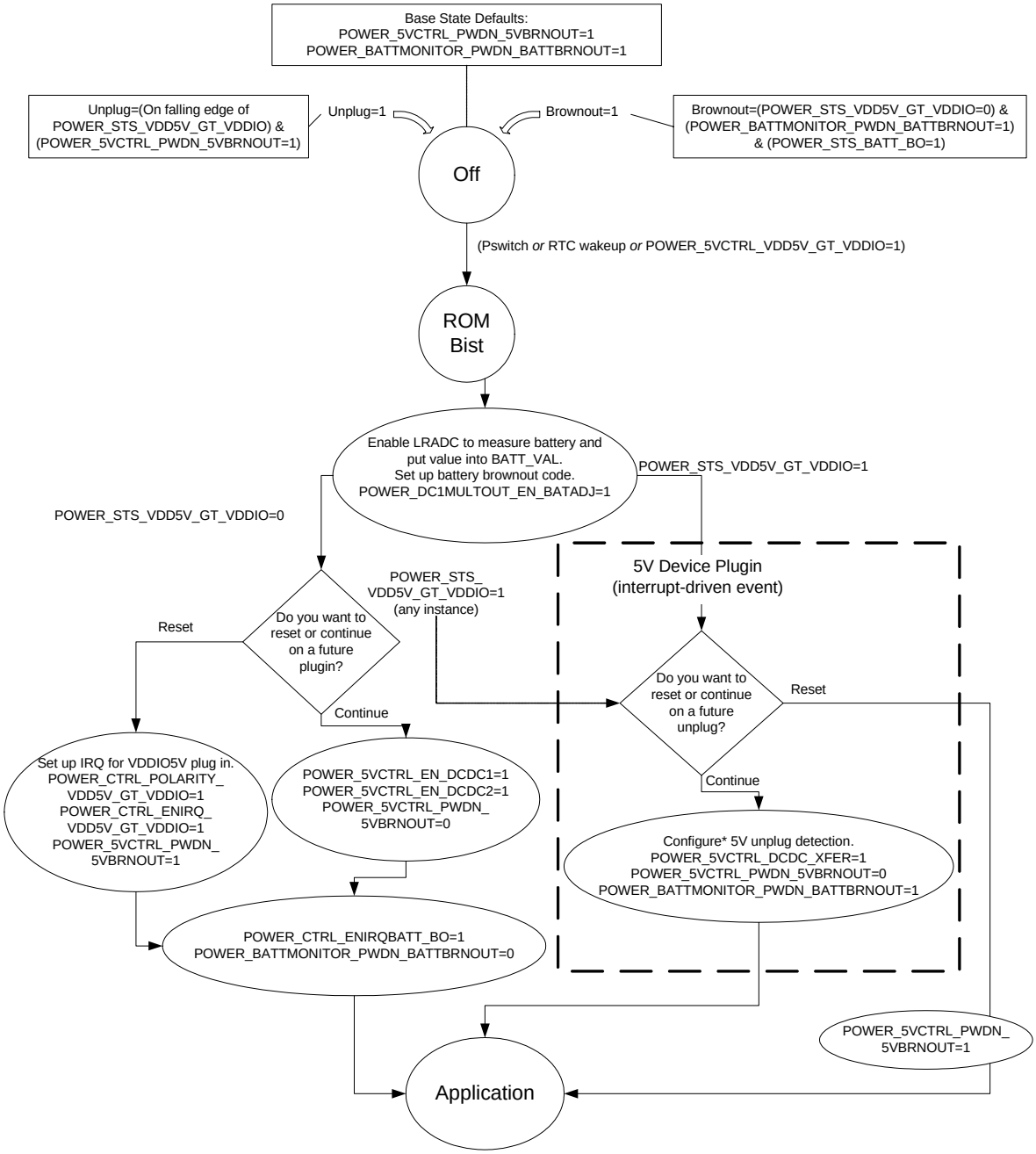
Typically, each voltage output is set to some voltage margin above the minimum operating level via VDDD_TRG and VDDIO_TRG in HW_POWER_VDDCTRL. The brownout detection threshold is also set (VDDD_BO and VDDIO_BO) above the minimum operation level, but below the rail's operating level. If the voltage drops to the brownout detector's level, then it optionally triggers a CPU Fast Interrupt (FIQ). The CPU can then alleviate the problem and/or shut down the system elegantly. See Chapter 2, "Characteristics and Specifications" on page 39 for suggested voltage settings for the supply and brownout targets for different operating frequencies.

To eliminate false detection, the brownout circuit filters transient noise above 1 MHz. Any system with an STMP36xx should include at least 10 µF of decoupling capacitance on all power rails. The capacitors should be arranged to filter supply noise in the 1-MHz and higher frequencies. See Figure 134.

31.2.2.2. DC-DC Extended Battery Life Features

The DC-DC converter has several other power-reducing programmable modes useful in maximizing battery life:

- **Li-Ion Buck/Boost**—Both Li-Ion battery configurations support buck/boost operation, which means that a VDDIO voltage can be supported that is higher than the input Li-Ion battery voltage. This is important to maximize battery life in all applications, but is crucial in hard drives that have large transient current requirements. In the Li-Ion configuration mode 0, the DC-DC converter will only power up as a buck converter to prevent over-discharge of the battery, and boost operation is enabled after powerup by setting EN_BOOST high and increasing POSLIMIT_BUCK to 0x30 in HW_POWER_DC2LIMITS. However, mode 0 buck/boost requires the 169-pin BGA package option, so that the necessary connections can be made to the power transistors. The single inductor Li-Ion configuration is automatically configured for buck/boost and can even powerup with a Li-Ion battery less than 3.0 V in either 100-pin or 169-pin options.
- **Transient Loading Optimizations**—Several new incremental improvements have been made to the control architecture of the switching converters. At this time, it is recommended the following bits be set via software in HW_POWER_LOOPCTRL to obtain maximum efficiency and minimum supply ripple: EN_CMP_HYST, TRAN_NOHYST, EN_DC2_RCSCALE, and EN_RCSCALE**.** EN_DC1_RCSCALE should be set only in alkaline battery configuration, mode 3, or in Li-Ion configurations when large decoupling inductors and capacitors (at least 22-uF capacitors and 15-uH inductors) are used. Also, converter configurations that use one inductor to produce two outputs (modes 3 and 1) should set EN_BATADJ and TOGGLE_DIF after programming FUNCV as described in the HW_POWER_DC1MULTOUT register definition. Lastly, single alkaline/NiMH configurations (mode 3) should also set EN_PFETOFF.



Note: See Section 31.3.2.1.*

Figure 134. Brownout Detection Flowchart

- **Pulse Frequency Modulation (PFM)**—PFM, also known as pulse-skipping mode, is used to reduce power consumed by the DC-DC converter when the voltage outputs are lightly-loaded at a cost of higher transient noise. Each DC-DC converter can be separately placed in PFM mode via the EN_DC2_PFM

and EN_DC1_PFM bits in HW_POWER_MINPWR. When using PFM mode, it is also necessary to set HYST_SIGN in HW_POWER_LOOPCTRL**.** When PFM mode is not enabled, HYST_SIGN should be set low**.**

- **DC-DC Switching Frequency**—The standard DC-DC switching frequency is 1.5 MHz, which provides a good mix of efficiency and power output. The frequency can be reduced to 750 kHz to reduce operating current in some light load situations via DC1_HALFCLK and DC2_HALFCLK in HW_POWER_MINPWR.
- **DC-DC Converter Power Down**—If the system is to operate from linear regulators or an external power supply, then the internal DC-DC converters can be powered down via DC1_STOPCLK and DC2_STOPCLK in HW_POWER_MINPWR. These bits are not intended to power down the whole system. Use the HW_POWER_RESET bits to power the system off.

31.3. Linear Regulators

The STMP36xx integrates two linear regulators that are typically used when the system is powered from a 5-V supply. Both of these regulators have an output impedance of approximately one ohm. One regulator generates VIO from the VDD5V pin, and the other generates VDD from the VIO supply. Therefore, all of the current is supplied by the VDD5V->VIO regulator. In normal system operation, the battery voltage is relatively stable. However, the VDD5V voltage can dynamically change as the product is plugged into a USB port or other 5-V supply. The STMP36xx is programmable to provide a variety of behaviors when the VDD5V supply becomes valid or invalid, as well as to support operation via USB or external power.

31.3.1. USB Compliance Features

Upon connection of 5 V to the powered-down device, the linear regulators will automatically power up the device. To meet USB inrush specifications, linear regulators have a current limit which is <100mA, nominally 50mA, and is active by default. This current limit is disabled in the ROM via the POWER_5VCTRL_DISABLE_ILIMIT after the supplies have reached their target values. System designers must understand that the current limit during 5-V power-up places restrictions on application current consumption until it is disabled. Specifically, after connection to 5 V, if the system draws more current than the current limit allows, the startup sequence does not complete and the ROM code does not execute.

Further, USB OTG implies that B-devices must draw very little current from 5 V. This requirement can be met by setting POWER_5VCTRL_ILIMIT_EQ_ZERO when the OTG application is active. The comparators required for OTG can be enabled by POWER_5VCTRL_OTG_PWRUP_CMPS**.** It is also possible to change the threshold of the Vbus valid comparator via POWER_5VCTRL_VBUSVALID_TRSH**.**

If very low power operation is required, as in USB suspend, then the circuits required to elegantly switch to the DC-DC converter may have to be powered off. In those cases, the system will have to fully power down after VDD5V becomes invalid. It can auto restart with DC-DC converter if HW_RTC_PERSIST0_AUTORESTART is set.

31.3.2. 5V to Battery Power Interaction

The STMP36xx supports several different options related to the interaction of the switching converters with the linear regulators. The two primary options are a reset on 5-V insertion/removal or a handoff to the DC-DC converters that is invisible to the end-user of the application. Figure 135 includes these two options as the two system architecture decision boxes in the figure.

31.3.2.1. Battery Power to 5-V Power

By default, the DC-DC converters turn off when VDD5V becomes valid and the system does not reset. If the system is operating from the DC-DC converter and using more than 50 mA, then if VDD5V becomes valid, the DC-DC will turn off and the linear regulators may not be able to supply the required current. The VDD and VIO rails will droop and the system will brownout and shut down. To avoid this issue, the LINREG_OFFSET, EN_DCDC1 (in all modes) and EN_DCDC2 (if using mode 0 or 2) bits should be set in anticipation of VDD5V becoming present. The EN_DCDC1 and EN_DCDC2 two bits will cause the DC-DC converters to remain on even after 5 V is connected and, thus, guarantee a stable supply voltage until the system is configured for removal of 5 V. The LINREG_OFFSET causes the DC-DC converters to regulate a higher target voltage than the linear regulators to prevent unwanted interaction between the two power supplies. LINREG_OFFSET affects the decode of the voltage targets as described in the VDDCTRL register. After the system is configured for removal of 5 V, EN_DCDC1 and EN_DCDC2 can be set low and DISABLE_ILIMIT set high in HW_POWER_5VCTRL to allow the linear regulators to supply the system power.

31.3.2.2. 5-V Power to Battery Power

Configuring the system for a 5-V-to-battery power handoff requires setup code to monitor the battery voltage as well as detect the removal of 5 V.

Monitoring the battery voltage is performed by the LRADC. Typically, this involves programming the LRADC registers to periodically monitor the battery voltage as described in Chapter 29, “Low-Resolution ADC and Touch-Screen Interface” on page 705. The measured battery voltage should be written into the HW_POWER_BATTMONITOR register field BATT_VAL using the AUTOMATIC field in the HW_LRADC_CONVERSION register. Also**,** configuring battery brownout should be performed so that the system behaves as desired when 5 V is no longer present and the battery is low.

The recommended method to detect removal of 5V requires setting VBUSVALID_5VDETECT and programming the detection threshold VBUSVALID_TRSH to 0x1 in HW_POWER_5VCTRL**.** Next, in order to minimize linear regulator and DC-DC converter interaction, it is necessary to set LINREG_OFFSET**.** Finally, set DCDC_XFER and clear PWDN_5VBRNOUT in the HW_POWER_5VCTRL register. This sequence is important because it is safe to disable the powerdown-on-unplug functionality of the device only after the system is completely ready for a transition to battery power.

31.3.2.3. 5-V Power and Battery Power

It is also possible to operate from both the 5-V and the DC-DC converters. This may be desirable under heavier load conditions than the 5 V can support. This functionality is enabled by setting EN_DCDC1**/2** (to enable the switching converters when 5v is present) and using either LINREGOFFSET or DISABLE_ILIMIT in

HW_POWER_5VCTRL to determine which path is the dominant power source. Battery charge can also be enabled in Li-Ion configurations to provide additional power efficiency when connected to 5 V, because the buck switching converters will efficiently convert the battery voltage to the desired VDDD and VDDIO voltages.

It is also possible to improve power efficiency in Li-Ion modes when connected to 5 V by enabling battery charge and also enabling the DC-DC converters by setting EN_DCDC1/2.

31.3.3. Power-Up Sequence

The DC-DC converters control the power-up and reset of the STMP36xx. The power-up sequence begins when the battery is connected to the BATT pin of the device (or a 5-V source is connected to the VDD5V pin). Either the BATT pin or VDD5V provides power through VOD_XTAL to the DC-DC startup circuitry, the crystal oscillator, and the real-time clock. This means that the crystal oscillator can be running, if desired, whenever a battery is connected to BATT pin. This feature allows the real-time clock to operate when the chip is in the off state. The crystal oscillator/RTC is the only power drain on the battery in this state and consumes only a very small amount of power. During this time, the VDD supply is held at ground, while the VIO rail is either shorted to the battery (modes 3 or 2) or held at ground (Li-Ion modes). This is the off state that continues until the system power up begins.

Power-up can be started with one of several events:

- PSWITCH pin > 0.9 * Vbat (1-cell) or > 0.5 * Vbat (2-cell and Li-Ion) for 100 ms
- VDD5V power pin > 4.25 V for 100 ms
- Real-time clock alarm wakeup

When a power-up event has occurred, if VDD5V is valid, then the on-chip linear regulators charge the VDD and VIO rails to their default voltages. If VDD5V is not valid, then the DC-DC supply the VDD and VIO rails. When the voltage rails have reached their target values, the digital logic reset is deasserted and the CPU begins executing code. If the power supplies do not reach the target values by the time PSWITCH is deasserted, the system returns to the off state.

The power-up time is dependent on the VDD/VIO load and battery or VDD5V voltage, but should be less than 100 ms. The VDD/VIO load should be minimal during power up to ensure proper startup of the DC-DC converters.

There is an integrated 5KΩ resistor that can be switched in between the VDDXTAL pin and the PSWITCH pin. If enabled (**HW_RTC_PERSIST0_AUTORESTART**), then the device immediately begins the power-up sequence after power down.

31.3.4. Power-Down Sequence

Power-down is also controlled by the DC-DC converters. When the DC-DC converters detect a power-down event, they return the player to the off state described above. The power-down sequence is started when one of these events occurs:

- HW_POWER_RESET_PWD bit set while the register is unlocked.
- PSWITCH pin has a fast (<15-ns) falling edge
- Watchdog timer expires while enabled

The HW_POWER_RESET_PWD_OFF bit disables all power-down paths except for the watchdog timer when it is set.

The lower 16 bits of the HW_POWER_RESET register can only be written if the value 0x3E77 is placed in the unlock field.

An external capacitor on the PSWITCH can be used to prevent unwanted power down due to falling edges. This can also be disabled in register HW_POWER_RESET_PWD_OFF.

31.3.4.1. Powered-Down State

While the chip is powered down, the VddD rail is pulled down to ground. The VDDIO rail is either shorted to ground (Li-Ion) or to the battery (1- or 2-cell). The crystal oscillator and the RTC can continue to operate by drawing power from the BATT pin. See Chapter 19, "Real-Time Clock, Alarm, Watchdog, and Persistent Bits" on page 497 for more information about operating an XTAL and RTC in the powered-down state.

To support peripherals that need to be completely powered down in the off state, the 1- and 2-cell operating modes integrate a peripheral power switch. This switch separates the VIO rail, which will be at the battery voltage in the off state, from peripherals that need to have their supply grounded. This switch is opened during the power-down state and active pulldowns are turned on to hold the peripheral power supply at ground, which ensures that any devices connected to this rail receive a valid power-on reset.

31.3.5. Reset Sequence

A reset event can be triggered by unlocking the HW_POWER_RESET register and setting the HW_POWER_RESET_RST_DIG bit. This reset only affects the digital logic, although the digital logic also includes most of the registers that control the analog portions of the chip. The persistent bits within the real-time clock block and the power module control bits are not reset using this method. The DC-DC converters and/or linear regulators continue to maintain the power supply rails during the reset.

31.3.6. Power Up, Power Down, and Reset Flow Chart

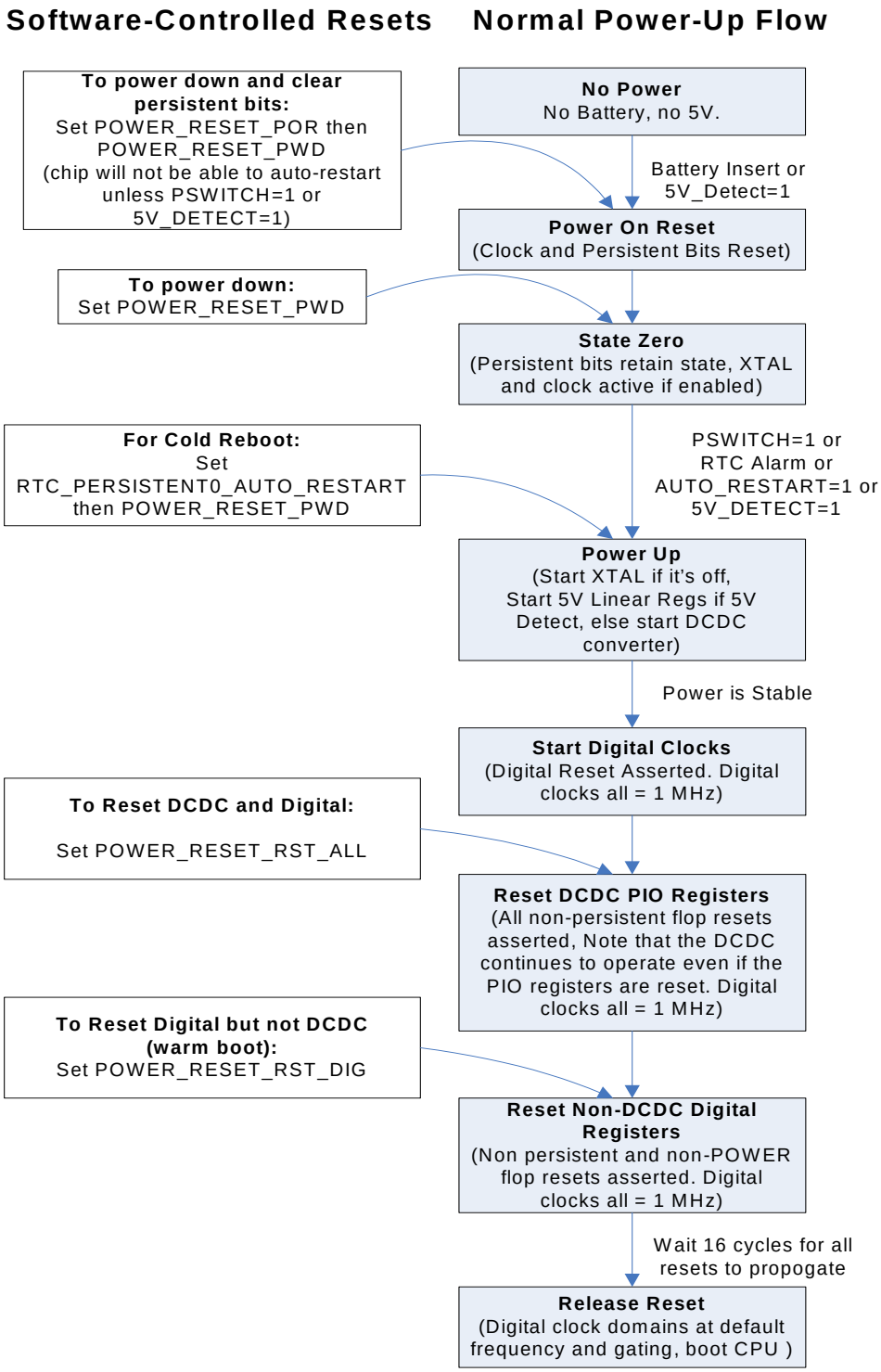


Figure 135. Power Up, Power Down, and Reset Flow Chart

31.4. PSWITCH Pin Functions

The PSWITCH pin has several functions whose operation is determined by the STMP36xx-based product's hardware and software design.

31.4.1. Power On

When the PSWITCH pin voltage is higher than approximately 0.9*Vbat for 1-cell or Vbat/2 for 2-cell or Li-Ion modes for >100 ms, the DC-DC converter begins its startup routine. This is the primary method of starting the system. All products based on the STMP36xx must have a mechanism of bringing PSWITCH high.

31.4.2. Power Down

If the PSWITCH pin voltage has a falling edge faster than 15 ns, then this sends a power-down request to the DC-DC converter. The fast-falling-edge power-down may be blocked by the HW_POWER_RESET_PWD_OFF function. The fast-falling edge can also be prevented by placing an RC filter on the PSWITCH pin. Most STMP36xx-based systems do not use the PSWITCH fast-falling-edge power-down and include the RC filter to prevent it from occurring accidentally.

31.4.3. Software Functions/Recovery Mode

When the PSWITCH pin voltage is pulled up to at least 0.9 * Vbat (1-cell) or Vbat/2 (2-cell and Li-Ion), the lower HW_DIGCTL_STATUS bit is set. Software can poll this bit and perform a function as desired by the product designer. Example functions include a play/pause/power-down button, delay for startup, etc.

When the PSWITCH pin is connected to VDDIO through a current limiting resistor, the upper HW_DIGCTL_STATUS bit is also set. If this bit is set for more than five seconds during ROM boot, the system executes the SigmaTel USB Firmware Recovery function. If the product designer does not wish to use SigmaTel USB Firmware Recovery, the product can be designed to not assert a voltage higher than the battery on the PSWITCH pin.

Refer to the SigmaTel STMP36xx reference schematics for example configurations of the PSWITCH pin.

31.5. Battery Monitor

The power control system includes a battery monitor. The battery monitor has two functions: battery brownout detection and battery voltage feedback to the DC-DC converter.

If the battery voltage drops below the programmable brownout, then a fast interrupt (FIQ) can be generated for the CPU. Software typically uses the LRADC to monitor the battery voltage and shut down elegantly while there is a minimal operating margin. But, if an unexpected event (such as a battery removal) occurs, then the system needs to be placed immediately in the off state to ensure that it can restart properly. The brownout is controlled in the HW_POWER_BATTMONITOR register. The IRQ must also be enabled in the interrupt collector.

To enable optimum performance over the battery range in modes 3 and 1, the DC-DC converter needs to be provided with the battery voltage, which is measured by the battery pin LRADC. Normally, LRADC channel 7 is dedicated to periodically measuring the battery voltage with a period in the millisecond range for most applications. The voltage is automatically placed into the BATT_VAL field of the

HW_POWER_BATTMONITOR register via the HW_LRADC_CONVERSION register. If necessary, software can turn off the automatic battery voltage update and set the BATT_VAL field manually.

31.6. Battery Charger

Some products in the STMP36xx family integrate charging for Li-Ion and NiMH batteries from a 5-V source connected to the VDD5V pin. The battery charger is essentially a linear regulator that has current and voltage limits.

Charge current is software programmable within the HW_BATTCHARGE register. The charger supports 0.1C for NiMH batteries, which results in a 12-hour charge time. Li-Ion batteries can be charged at the lower of 1C, 785 mA, or the VDD5V current limit. USB charging is typically limited to 500 mA or less to meet compliance requirements. Typical charge times for a Li-Ion battery are 1.5 to 3 hours with >70% of the charge delivered in the first hour.

The battery charge voltage limit is determined by the battery type. If the DC-DC converter is configured for 1-cell mode, then the charge voltage is limited to 1.75 V. If the DC-DC converter is configured for Li-Ion mode, then the charge voltage is limited to 4.2 V.

The Li-Ion charge is typically stopped after a certain time limit OR when the charging current drops below 10% of the charge current setting. The HW_BATTCHARGE register includes controls for the maximum charge current AND for the stop charge current. While the charger is delivering current greater than the stop charge limit, the HW_POWER_STS_CHRGSTS bit will be high. This bit should be polled (a low rate of 1 second or greater is ok) during charge. When the bit goes low, the charging is complete. It would be good practice to check that this bit is low for two consecutive checks, as the DC-DC switching might cause a spurious “low” result. Once this bit goes low, the charger can either be stopped immediately or stopped after a “top-off” time limit. Although the charger will avoid exceeding the charge voltage limit on the battery, it is NOT recommended to leave the charger active indefinitely. It should be turned off when the charge is complete.

NiMH charging does not use the “stop charge current” feature. NiMH charging should be stopped after 12 hours (at a 0.1C rate) regardless of the output current.

One can programatically monitor the battery voltage using the LRADC. The charger has its own (very robust) voltage limiting that operates independently of the LRADC. But monitoring the battery voltage during the charge might be helpful for reporting the charge progress.

The battery charger is capable of generating a large amount of heat within the STMP36xx, especially at currents above 400 mA. The dissipated power can be estimated as: (5V – battery_volt) * current. At max current (785 mA) and a 3-V battery, the charger can dissipate 1.57 W, raising the die temp as much as 80 C°. To ensure that the system operates correctly, the die temperature sensor should be monitored every 100 ms. If the die temperature exceeds 115 C° (the max value for the chip temp sensor), then the battery charge current must be reduced. The LRADC can also be used to monitor the battery temperature or chip temperature. There is an integrated current source for the external temperature sensor that can be configured and enabled via HW_LRADC_CTRL2 register.

31.7. Silicon Speed Sensor

The STMP36xx integrates two silicon speed sensors to measure the performance characteristics of an individual die at its ambient temperature and process parametrics. Each sensor consists of a ring oscillator and a frequency counter. The ring oscillator runs on the VDD power rail. Therefore, its frequency tracks the silicon performance as it changes in response to changes in operating voltage and temperature. The crystal oscillator is directly used as the precision time base for measuring the frequency of a ring oscillator. The ring oscillator is normally disabled. There is a 7-bit counter connected to the ring oscillator that performs the frequency measurement. See the HW_POWER_SPEED_TEMP register.

Thus, the counter holds the number of cycles the ring oscillator was able to generate during one crystal clock period. The natural frequency of the ring oscillator strongly tracks the silicon process parametrics, i.e., faster silicon processes yield ring oscillators that run faster and thereby yield larger count values. The natural frequency tracks junction temperature effects on silicon speed as well.

The information given by the speed sensor can be used with the silicon temperature and process parameters, which can also be monitored by system software. SigmaTel can provide a power management application note and firmware that takes full advantage of the on-chip monitoring functions to enable minimum-voltage operation.

31.8. DC-DC Programmable Registers

31.8.1. Power Control Register Description

The Power Control Register contains control bits specific to the digital section.

HW_POWER_CTRL 0x80044000
HW_POWER_CTRL_SET 0x80044004
HW_POWER_CTRL_CLR 0x80044008
HW_POWER_CTRL_TOG 0x8004400C

Table 973. HW_POWER_CTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2	CLKGATE	RSRVD1																					BATT_BO_IRQ	ENIRQBATT_BO	VDDIO_BO_IRQ	ENIRQVDDIO_BO	VDDD_BO_IRQ	ENIRQVDDD_BO	POLARITY_VDD5V_GT_VDDIO	VDD5V_GT_VDDIO_IRQ	ENIRQVDD5V_GT_VDDIO

Table 974. HW_POWER_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD2**	RO	0x0	Empty Description.
30	**CLKGATE**	RW	0x1	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block. This bit has no effect on the RTC analog section.
29:9	**RSRVD1**	RO	0x0	Empty Description.
8	**BATT_BO_IRQ**	RW	0x0	Interrupt Status for BATT_BO. It is reset by software by writing a zero to the bit position or by writing a one to the SCT clear address space.
7	**ENIRQBATT_BO**	RW	0x0	Enable interrupt for battery brownout.
6	**VDDIO_BO_IRQ**	RW	0x0	Interrupt Status for VDDIO_BO. It is reset by software by writing a zero to the bit position or by writing a one to the SCT clear address space.
5	**ENIRQVDDIO_BO**	RW	0x0	Enable interrupt for VDDIO brownout.
4	**VDDD_BO_IRQ**	RW	0x0	Interrupt Status for VDDD_BO. It is reset by software by writing a zero to the bit position or by writing a one to the SCT clear address space.
3	**ENIRQVDDD_BO**	RW	0x0	Enable interrupt for VDDD brownout.
2	**POLARITY_VDD5V_GT_VDDIO**	RW	0x1	Set to 1 to check for 5V connected. Set to 0 to check for 5V disconnected.
1	**VDD5V_GT_VDDIO_IRQ**	RW	0x0	Interrupt status for VDD5V_GT_VDDIO signal. Interrupt polarity is set using POLARITY_VDD5V_GT_VDDIO. It is reset by software by writing a zero to the bit position or by writing a one to the SCT clear address space.
0	**ENIRQVDD5V_GT_VDDIO**	RW	0x0	Enable interrupt for 5V detect.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.2. DC-DC 5V Control Register Description

This register contains the configuration options of the power management sub-system that are available when external 5V is applied.

HW_POWER_5VCTRL 0x80044010
HW_POWER_5VCTRL_SET 0x80044014
HW_POWER_5VCTRL_CLR 0x80044018
HW_POWER_5VCTRL_TOG 0x8004401C

Table 975. HW_POWER_5VCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2										PWDN_5VBRNOUT	PWDN_IOBRNOUT	DISABLE_ILIMIT	DCDC_XFER	EN_BATT_PULLDN	VBUSVALID_5VDETECT	RSRVD1						VBUSVALID_TRSH		USB_SUSPEND_I	VBUSVALID_TO_B	ILIMIT_EQ_ZERO	OTG_PWRUP_CMPS	EN_DCDC2	PWD_VDDD_LINREG	EN_DCDC1	LINREG_OFFSET

Table 976. HW_POWER_5VCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:22	**RSRVD2**	RO	0x0	Empty Description.
21	**PWDN_5VBRNOUT**	RW	0x1	The purpose of this bit is to power down the device if 5V is removed before the system is completely initialized. Clear this bit to disable automatic hardware powerdown AFTER the system is configured for 5v removal. The removal of 5v is detected via the VDD5V_GT_VDDIO status signal and is latched internally. Due to this latch, it is not recommended to reset this bit high after it has been cleared. This bit should not be set if DCDC_XFER is set.
20	**PWDN_IOBRNOUT**	RW	0x0	It is not recommended to use this bit, as the functionality to better supported via PWDN_5VBRNOUT. This bit enables automatic hardware power-down of the system when VDDIO crosses the VDDIO brownout threshold. Setting this bit assumes that the 5V is present and enables the VDDIO brownout comparator automatically. (Only set this bit AFTER 5V has been detected; otherwise incorrect behavior will result.) This bit should not be set if DCDC_XFER is set.
19	**DISABLE_ILIMIT**	RW	0x0	Disable the current limit in the linear regulators. The current limit defaults to enabled so that the system can meet the USB in rush current specification of 100mA + 50uC. Note that this bit does not affect the battery charger current.
18	**DCDC_XFER**	RW	0x0	Enable automatic transition to switching DC-DC converters when VDD5V is removed. The LRADC must be operational and the BATT_VAL field must be written with the battery voltage using 8-mV step-size. When using one of the single-inductor/two-output configurations, it is also important to set the EN_BATADJ field. Also, this bit should NOT be set if the PWDN_IOBRNOUT is set.

Table 976. HW_POWER_5VCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17	**EN_BATT_PULLDN**	RW	0x0	Add very small current source (less than 5uA) to drain the battery pin. This maybe useful to detect the presence of an external battery when operating from an external 5-V supply. This information could be important if the DCDC_XFER functionality is enabled.
16	**VBUSVALID_5VDETECT**	RW	0x0	Enable the VBus Valid comparator and use it as detection circuit for 5V in the switching converters. Default is for the switching converter to use the VDD5V_GT_VDDIO status bit to determine the presence of 5V in the system. The VBus Valid comparator provides a more accurate and adjustable threshold to determine the presence of 5V in the system.
15:10	**RSRVD1**	RO	0x0	Empty Description.
9:8	**VBUSVALID_TRSH**	RW	0x0	Set the threshold for the VBus Valid comparator, 00=4.5V 01: 4.3V 10: 2.5V 11: 4.75V
7	**USB_SUSPEND_I**	RW	0x0	Turn off switching DC-DC converter bias current. This bit will prevent the switching DC-DC converters from working correctly and should only be used as needed to minimize system power to meet the USB suspend current specification.
6	**VBUSVALID_TO_B**	RW	0x0	This bit muxes the Bvalid comparator to the VBus Valid comparator and is used for test purposes only.
5	**ILIMIT_EQ_ZERO**	RW	0x0	The amount of current the device will consume from the 5V rail is minimized. The VDDIO linear regulator current limit is set to zero mA. Also, the source of current for the crystal oscillator and RTC is switched to the battery. Note that this functionality does not affect battery charge.
4	**OTG_PWRUP_CMPS**	RW	0x0	VBus Valid comparators are enabled.
3	**EN_DCDC2**	RW	0x0	Enables the switching DC-DC converter #2 when 5V is present. Use of the DC-DC converters and battery charge together may reduce system current requirements from the 5-V supply.
2	**PWD_VDDD_LINREG**	RW	0x0	Disable VDDD linear regulator. The bit can be used with EN_DCDC1 to force VDDD to be generated from the battery supply using DC-DC#1 even when external 5V is connected.

Table 976. HW_POWER_5VCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	**EN_DCDC1**	RW	0x0	Enables the switching DC-DC converter#1 when 5V is present.
0	**LINREG_OFFSET**	RW	0x0	Reverses the offset in the DC-DC converters so that the switching DC-DC converter regulates to a higher voltage than the linear regulators. The default is that the linear regulators regulate to one setting higher than the switching converters as described in VDDCTRL. Setting this bit is useful in DC-DC handoff to prevent the linear regulators from fighting with the switching DC-DC converters, because the regulators can only pull up the supplies. Note that this bit does not affect battery charge.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.3. DC-DC Minimum Power and Miscellaneous Control Register Description

This register controls options to drop the power used by the switching DC-DC converters. These bits should only be modified with guidance from SigmaTel.

HW_POWER_MINPWR 0x80044020
HW_POWER_MINPWR_SET 0x80044024
HW_POWER_MINPWR_CLR 0x80044028
HW_POWER_MINPWR_TOG 0x8004402C

Table 977. HW_POWER_MINPWR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD2								TEST_DISCHRG_VBUS	TEST_CHRG_VBUS	DC2_TST	DC1_TST	PERIPHERALSWOFF	TOGGLE_DIF	DISABLE_VDDIOSTEP	DISABLE_VDDSTEP	RSRVD1						SEL_PLLDIV16CLK	PWD_VDDIOBO	LESSANA_I	DC1_HALFFETS	DC2_STOPCLK	DC1_STOPCLK	EN_DC2_PFM	EN_DC1_PFM	DC2_HALFCLK	DC1_HALFCLK

Table 978. HW_POWER_MINPWR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**RSRVD2**	RO	0x0	Empty Description.
23	**TEST_DISCHRG_VBUS**	RW	0x0	Test function for OTG Disharge VBus. Only for test purposes.

Table 978. HW_POWER_MINPWR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	**TEST_CHRG_VBUS**	RW	0x0	Test function for OTG Charge VBus. Only for test purposes.
21	**DC2_TST**	RW	0x0	Reserved.
20	**DC1_TST**	RW	0x0	Reserved
19	**PERIPHERALSWOFF**	RW	0x0	Turn off peripheral switch and force peripheral supply to ground. This switch is used in single or series AA/AAA configuration. The peripheral switch is always active when 5V is present.
18	**TOGGLE_DIF**	RW	0x0	Set high to enable supply stepping to change only after the differential control loop has toggled as well. This should eliminate any chance of large transients when supply voltage changes are made in single-inductor dual-output DC-DC converter modes 01 or 11. In modes 10 or 00, this bit has no effect.
17	**DISABLE_VDDIOSTEP**	RW	0x0	DC-DC#1 or #2 steps the target VDDD voltage one programming step at a time to minimize transient noise. This feaure can be disabled by setting this bit. In DC-DC#1 PFM, this bit will prevent VDDIO from being toggled as well as VDDD. This PFM feature may be necessary if VDDIO is lightly loaded relative to VDDD.
16	**DISABLE_VDDSTEP**	RW	0x0	DC-DC#1 steps the target VDDD voltage one programming step at a time to minimize transient noise. This feaure can be disabled by setting this bit.
15:10	**RSRVD1**	RO	0x0	Empty Description.
9	**SEL_PLLDIV16CLK**	RW	0x0	This bit selects the source of the clock used for the DC-DC converter and the LRADC. The default is to use the 24-MHz clock. Setting this bit selects the PLL clock divided by 16 as a clock source for both the DC-DC converter and the LRADC.
8	**PWD_VDDIOBO**	RW	0x0	Power-Down the VDDIO Brownout Comparator. This should only be done when it is acceptable to lose the ability to detect a VDDIO brownout. Default is VDDIO Brownout Comparator enabled.
7	**LESSANA_I**	RW	0x0	Reduce DC-DC analog bias current 20%. This bit is intended to reduce power in low-performance operating modes, such as USB suspend.
6	**DC1_HALFFETS**	RW	0x0	Disable half the power transistors in DC-DC#1. This maybe be useful in low-power conditions when the increased resistance of the power FETs is acceptable.
5	**DC2_STOPCLK**	RW	0x0	Stop the clock to internal logic of switching converter DC-DC#1. This bit will take effect only after the switching FETs are off, due to battery configuration, PFM mode, or internal linear regulator operation.
4	**DC1_STOPCLK**	RW	0x0	Stop the clock to internal logic of switching converter DC-DC#1. This bit will take effect only after the switching FETs are off, due to battery configuration, PFM mode, or internal linear regulator operation.

Table 978. HW_POWER_MINPWR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	**EN_DC2_PFM**	RW	0x0	Forces DC-DC#2 to operate in a Pulse Frequency Modulation (PFM) mode. Intended to allow minimum system power in very low-power configurations when increased ripple on the supplies is acceptable. Also, HYST_SIGN in HW_POWER_LOOPCTRL should be set high when using PFM mode.
2	**EN_DC1_PFM**	RW	0x0	Forces DC-DC#1 to operate in a Pulse Frequency Modulation mode. Intended to allow minimum system power in very low-power configurations when increased ripple on the supplies is acceptable. Also, HYST_SIGN in HW_POWER_LOOPCTRL should be set high when using PFM mode.
1	**DC2_HALFCLK**	RW	0x0	Slow down DC-DC#2 clock from 1.5 MHz to 750 kHz. This maybe be useful to improve efficiency at light loads or improve EMI radiation, although peak-to-peak voltage on the supplies will increase.
0	**DC1_HALFCLK**	RW	0x0	Slow down DC-DC#1 clock from 1.5 MHz to 750 kHz. This maybe be useful to improve efficiency at light loads or improve EMI radiation, although peak-to-peak voltage on the supplies will increase.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.4. Battery Charge Control Register Description

This register cotrols the battery charge features for both NiMH slow charge and Li-Ion charge.

HW_POWER_BATTCHRG 0x80044030
HW_POWER_BATTCHRG_SET 0x80044034
HW_POWER_BATTCHRG_CLR 0x80044038
HW_POWER_BATTCHRG_TOG 0x8004403C

Table 979. HW_POWER_BATTCHRG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3												CHRG_STS_OFF	LIION_4P1	USE_EXTERN_R	PWD_BATTCHRG	RSRVD2				STOP_ILIMIT				RSRVD1				BATTCHRG_I			

Table 980. HW_POWER_BATTCHRG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	**RSRVD3**	RO	0x0	Empty Description.
19	**CHRG_STS_OFF**	RW	0x0	Setting this bit disables the CHRGSTS status bit. Disabling CHRGSTS should only be done when the switching converters are enabled during battery charge if noise from the switching converters causes CHRGSTS to toggle excessively.
18	**LIION_4P1**	RW	0x0	Default is 4.2-V final charging voltage
17	**USE_EXTERN_R**	RW	0x0	Set to 1 to use INTERNAL resistor to generate the battery charge current. Default uses EXTERNALLY generated precision bias current.
16	**PWD_BATTCHRG**	RW	0x1	Power-down the battery charge circuitry. This should only be set low when 5V is present
15:12	**RSRVD2**	RO	0x0	Empty Description.
11:8	**STOP_ILIMIT**	RW	0x0	Current threshold at which Li-Ion battery charge stops. The current represented by each bits is as follows: (100 mA, 50 mA, 20 mA, 10 mA) = (bit 3, bit 2, bit 1, bit 0) It is recommended to set this value to 10% of the charge current.
7:6	**RSRVD1**	RO	0x0	Empty Description.
5:0	**BATTCHRG_I**	RW	0x00	Magnitude of the battery charge current, the current represented by each bits is as follows: (400 mA, 200 mA, 100 mA, 50 mA, 20 mA, 10 mA) = (bit 5,bit 4, bit 3, bit 2, bit 1, bit 0)

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.5. VDDD and VDDIO Supply Targets and Brownouts Control Register Description

This register controls the voltage targets and brownout targets for the VDDD and VDDIO supplies generated from the switching DC-DC converters and integrated linear regulators. Both VDDD and VDDIO brownout comparators default enabled. When the VDDD and VDDIO are generated using the internal switching converters, the following guideline is recommended: Alterations to VDDD_TRG and VDDIO_TRG should not be done at the same time. After making a voltage target change, another one should not be programmed until the DC1_OK and DC2_OK flags return to the high state.

HW_POWER_VDDCTRL 0x80044040

Table 981. HW_POWER_VDDCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4			VDDIO_BO					RSRVD3			VDDIO_TRG					RSRVD2			VDDD_BO					RSRVD1			VDDD_TRG				

Table 982. HW_POWER_VDDCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	**RSRVD4**	RO	0x0	Empty Description.
28:24	**VDDIO_BO**	RW	0x0C	Voltage level of the brownout for the VDDIO supply. The step size is 64 mV with 0x00 = 2.049 V, 0x1F = 4.034 V, and the reset value is 2.817 V. The decode is identical when using linear regulators or the switching DC-DC converters and is not affected by LINREG_OFFSET. These values represent a mean value, but individual parts will have slight part-to-part variations with a sigma of around 12 mV from either switching converters or linear regulators.
23:21	**RSRVD3**	RO	0x0	Empty Description.
20:16	**VDDIO_TRG**	RW	0x10	Voltage level of the VDDIO supply. The step size of this field is 64 mV. When using the switching converters, 0x00 = 2.049 V, 0x1F = 4.034 V, and the reset value = 3.073 V. Due to impedance between the PCB and the internal dcdc regulation circuit, the measured supply voltage off chip will be typically higher than the programmed voltage due to IR drop. Thus, the magnitude of this drop will depend on board layout as well as application current requirements. When using the linear regulators, this field is interpreted with a one-step offset, such that 0x00 = 2.113 V, 0x1E-0x1F = 4.034 V, and the reset value of x10=3.137 V. However, it should be noted the linear regulators have an output impedance of near one ohm, so the supply voltages will also sag depending on the magnitude of current being drawn from the linear regulators. These target voltage values represent a mean value, but individual parts will have slight part-to-part variations with a sigma of around 12 mV from either switching converters or linear regulators. Note that the offset between switching converters and linear regulators can be reversed using LINREG_OFFSET.
15:13	**RSRVD2**	RO	0x0	Empty Description.

Table 982. HW_POWER_VDDCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12:8	**VDDD_BO**	RW	0x10	Voltage level of the brownout for the VDDD supply. 0x00 - 0x08 = 1.280 V. 0x09 - 0x1D = 32m V / per step above 0x08. 0x1E = 2.108 V, 0x1F = 2.20 V, and the reset value is 1.536 V. The decode is identical when using linear regulators or the switching DC-DC converters and is not affected by LINREG_OFFSET. These values represent a mean value, but individual parts will have slight part-to-part variations with a sigma of around 6 mV from either switching converters or linear regulators.
7:5	**RSRVD1**	RO	0x0	Empty Description.
4:0	**VDDD_TRG**	RW	0x16	Voltage level of the VDDD supply. The step size of this field is 32 mV. When using the switching converters, 0x00 - 0x08 = 1.280 V. 0x09 - 0x1D = 32 mV / per step above 0x08. 0x1E = 2.108 V, 0x1f = 2.20 V, and the reset value of x16=1.728 V. Due to impedance between the PCB and the internal dcdc regulation circuit, the measured supply voltage off chip will be typically higher than the programmed voltage due to IR drop. Thus, the magnitude of this drop will depend on board layout as well as application current requirements. When using the linear regulators, this field is interpreted with a one step offset such that 0x00 - 0x08 = 1.309 V. 0x09 - 0x1c = 32 mV / per step above 0x08. 0x1d=2.108 V 0x1E-0x1F = 2.20 V, and the reset value is 1.760 V. However, it should be noted the linear regulators have an output impedance of near one ohm, so the supply voltages will also sag depending on the magnitude of current being drawn from the linear regulator. These values represent a mean value, but individual parts will have slight part-to-part variations with a sigma of around 6mV from either switching converters or linear regulators. Note that the offset between switching converters and linear regulators can be reversed using LINREG_OFFSET. Programming VDDD_TRG to a value above 2.0 Volts is not intended for customer applications.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.6. DC-DC#1 MultiOutput Converter Modes Control Register Description

This register contains controls that may need to be adjusted when using DC-DC converter configurations that support two outputs using a single inductor. For single output/single inductor modes, this register should be left in its default state.

HW_POWER_DC1MULTOUT 0x80044050

Table 983. HW_POWER_DC1MULTOUT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3							FUNCV									RSRVD2							EN_BATADJ	RSRVD1				ADJTN			

Table 984. HW_POWER_DC1MULTOUT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSRVD3**	RO	0x0	Empty Description.
24:16	**FUNCV**	RW	0x000	This field should be programmed before EN_BATADJ and should be updated whenever the target values for the supplies are changed. For mode 01, single-inductor Li-Ion, write (VDDIO - VDD)/8e-03, and for mode 11, NiMH, write (VDDIO*VDD)/((VDDIO-VDD)*8e-03)). The values of VDDIO and VDDD are in volts, and the final result can be rounded. Note that this field cannot be programmed above x1FF. For example, for a VDDIO target voltage of 3.07 V in mode 11, FUNCV will not reach x1FF until VDDD > 1.74 V. If it desired to use a higher value of VDDD, set FUNCV=x1FF and begin to use ADJ_TN after consulting Sigmatel.
15:9	**RSRVD2**	RO	0x00	Empty Description.
8	**EN_BATADJ**	RW	0x0	This bit enables DC-DC#1 to improve efficiency and minimize ripple in the double output / single inductor configurations. Both of these features require that the LRADC be enabled and constantly monitoring the battery voltage. The battery voltage should be written to the BATT_VAL field in the BATT_MONITOR register using 8-mV step size. The FUNCV field in this register must also be written correctly when powering from the DC-DC converter in modes 01 or 11. It is not recommended to use the bit at the same time as EN_DC1_PFM
7:4	**RSRVD1**	RO	0x0	Empty Description.
3:0	**ADJTN**	RW	0x0	Two's complement number that can be used to adjust the duty cycle of VDDIO when using modes 01 or 11. This can be used to optimize efficiency performance of the double output boost converter after the FUNCV input is limited at 0x1ff.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.7. DC-DC#1 Duty Cycle Limits Control Register Description

This register defines the upper and lower duty cycle limits of DC-DC#1. These values depend on details of switching converter implementation and should not be changed without guidance from SigmaTel.

HW_POWER_DC1LIMITS 0x80044060

Table 985. HW_POWER_DC1LIMITS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4							EN_PFETOFF	RSRVD3	POSLIMIT_BOOST							RSRVD2	POSLIMIT_BUCK							RSRVD1	NEGLIMIT						

Table 986. HW_POWER_DC1LIMITS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSRVD4**	RO	0x0	Empty Description.
24	**EN_PFETOFF**	RW	0x0	Enables DC-DC#1 VDDD PFET to turn off when battery is greater than VDDD target voltage in double output boost mode. This bit will allow VDDD to be less than the battery voltage even though DC-DC#1 is configured as a boost converter. This bit should only be used in double output boost configuration, mode 3.
23	**RSRVD3**	RO	0x0	Empty Description.
22:16	**POSLIMIT_BOOST**	RW	0x02	Upper limit duty cycle limit in DC-DC#1 in boost mode.
15	**RSRVD2**	RO	0x0	Empty Description.
14:8	**POSLIMIT_BUCK**	RW	0x1E	Upper limit duty cycle limit in DC-DC#1 in buck mode.
7	**RSRVD1**	RO	0x0	Empty Description.
6:0	**NEGLIMIT**	RW	0x5F	Negative duty cycle limit of DC-DC#1.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.8. DC-DC#2 Duty Cycle Limits Control Register Description

This register defines the upper and lower duty cycle limits of DC-DC#2. These values depend on details of switching converter implementation and should not be changed without guidance from SigmaTel.

HW_POWER_DC2LIMITS 0x80044070

Table 987. HW_POWER_DC2LIMITS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD4							EN_BOOST	RSRVD3	POSLIMIT_BOOST							RSRVD2	POSLIMIT_BUCK							RSRVD1	NEGLIMIT						

Table 988. HW_POWER_DC2LIMITS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	**RSRVD4**	RO	0x0	Empty Description.
24	**EN_BOOST**	RW	0x0	Enables buck-boost operation in mode 00 Li-Ion configuration. This is intended to allow VDDIO to support hard drive current loads with Li-Ion battery voltages down below 3.0 V. Both ends of the inductor must be connected to different pairs of integrated switching FETs to permit this functionality. POSLIMIT_BUCK will need to be increased to allow the control loop to operate in the boost region.
23	**RSRVD3**	RO	0x0	Empty Description.
22:16	**POSLIMIT_BOOST**	RW	0x02	Upper limit duty cycle limit in DC-DC#2 in boost mode.
15	**RSRVD2**	RO	0x0	Empty Description.
14:8	**POSLIMIT_BUCK**	RW	0x1E	Upper limit duty cycle limit in DC-DC#2 in buck mode
7	**RSRVD1**	RO	0x0	Empty Description.
6:0	**NEGLIMIT**	RW	0x5F	Negative duty cycle limit of DC-DC#2.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.9. Converter Loop Behavior Control Register Description

This register defines the control loop parameters available for DC-DC#1 and DC-DC#2.

HW_POWER_LOOPCTRL 0x80044080
HW_POWER_LOOPCTRL_SET 0x80044084
HW_POWER_LOOPCTRL_CLR 0x80044088
HW_POWER_LOOPCTRL_TOG 0x8004408C

Table 989. HW_POWER_LOOPCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD5	TRAN_NOHYST	HYST_SIGN	EN_CMP_HYST	EN_DC2_RCSCALE	EN_DC1_RCSCALE	RC_SIGN	EN_RCSCALE	RSRVD4	DC2_FF			DC2_R				RSRVD3		DC2_C		RSRVD2	DC1_FF			DC1_R				RSRVD1		DC1_C	

Table 990. HW_POWER_LOOPCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**RSRVD5**	RO	0x0	Empty Description.
30	**TRAN_NOHYST**	RW	0x0	Disables hysteresis when DC1_OK or DC2_OK is low. This will speed voltage transitions.
29	**HYST_SIGN**	RW	0x0	Invert the sign of the hysteresis in DC-DC analog comparators. This bit should set when using PFM mode.
28	**EN_CMP_HYST**	RW	0x0	Enable hysteresis in switching converter analog comparators. This feature will improve transient supply ripple and efficiency.
27	**EN_DC2_RCSCALE**	RW	0x0	Enables digital DC-DC#2 control loop to respond faster to heavy transient loads
26	**EN_DC1_RCSCALE**	RW	0x0	Enables digital DC-DC#1 control loop to respond faster to heavy transient loads. It is not recommended to set this bit in Li-Ion applications unless using 15-uH inductors and capacitors larger than 22 uF. There are no restrictions when using this bit in alkaline battery applications.
25	**RC_SIGN**	RW	0x0	Invert the sign of the heavy load detection in the DC-DC analog circuits
24	**EN_RCSCALE**	RW	0x0	Enable analog circuit of DC-DC converter to detect heavy load conditions. This field should be set high when either EN_DC1_RCSCALE or EN_DC2_RCSCALE is set.
23	**RSRVD4**	RO	0x0	Empty Description.
22:20	**DC2_FF**	RW	0x0	Two's complement feed-forward step in duty cycle in DC-DC#2. Each time this field makes a transition from 0x0, the loop filter of the DC-DC converter is stepped once by a value proportional to the change. This can be used to force a certain control loop behavior, such as improving response under known heavy load transients.
19:16	**DC2_R**	RW	0x2	Magnitude of proportional control parameter in DC-DC#2 control loop.

Table 990. HW_POWER_LOOPCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:14	**RSRVD3**	RO	0x0	Empty Description.
13:12	**DC2_C**	RW	0x0	Ratio of integral control parameter to proportional control parameter in DC-DC#2. 00: Nominal, 01: Increase ratio, 10: Decrease ratio, 11: Lowest ratio. This setting is used to optimize efficiency and loop response.
11	**RSRVD2**	RO	0x0	Empty Description.
10:8	**DC1_FF**	RW	0x0	Two's complement feed forward step in duty cycle in DC-DC#1. Each time this field makes a transition from 0x0, the loop filter of the DC-DC converter is stepped once by a value proportional to the change. This can be used to force a certain control loop behavior, such as improving response under known heavy load transients.
7:4	**DC1_R**	RW	0x2	Magnitude of proportional control parameter in DC-DC#1 control loop.
3:2	**RSRVD1**	RO	0x0	Empty Description.
1:0	**DC1_C**	RW	0x0	Ratio of integral control parameter to proportional control parameter in DC-DC#1. 00: Nominal, 01: Increase ratio, 10: Decrease ratio, 11: Lowest ratio. This setting is used to optimize efficiency and loop response

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.10. Power Subsystem Status Register Description

HW_POWER_STS 0x80044090

Table 991. HW_POWER_STS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
BATT_CHRG_PRESENT	RSRVD5									MODE		RSRVD4			BATT_BO	RSRVD3	CHRGSTS	DC2_OK	DC1_OK	RSRVD2		VDDIO_BO	VDDD_BO	RSRVD1			VDD5V_GT_VDDIO	AVALID	BVALID	VBUSVALID	SESSEND

Table 992. HW_POWER_STS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	**BATT_CHRG_PRESENT**	RO	0x1	0= Battery charge circuit is not present in this product.
30:22	**RSRVD5**	RO	0x0	Empty Description.
21:20	**MODE**	RO	0x0	Battery configuration of system, 00= Li-Ion dual-converter, 01 = Li-Ion single-inductor, 10 = Series AA or AAA , 11 = Single AA or AAA
19:17	**RSRVD4**	RO	0x0	Empty Description.
16	**BATT_BO**	RO	0x0	Output of battery brownout comparator.
15	**RSRVD3**	RO	0x0	Empty Description.
14	**CHRGSTS**	RO	0x0	Battery charging status. High during Li-Ion battery charge until the charging current falls below the STOP_ILIMIT threshold.
13	**DC2_OK**	RO	0x0	High when DC-DC#2 control loop has stabilized after a voltage target change.
12	**DC1_OK**	RO	0x0	High when DC-DC#1 control loop has stabilized after a voltage target change.
11:10	**RSRVD2**	RO	0x0	Empty Description.
9	**VDDIO_BO**	RO	0x0	Output of VDDIO brownout comparator. High when a brownout is detected. This comparator defaults powered up, but can be powered down via the POWER_MINPWR register.
8	**VDDD_BO**	RO	0x0	Output of VDDD brownout comparator. High when a brownout is detected. It is not possible to power-down this comparator.
7:5	**RSRVD1**	RO	0x0	Empty Description.
4	**VDD5V_GT_VDDIO**	RO	0x0	Indicates the voltage on the VDD5V pin is higher than VDDIO by a Vt voltage, nominally 500 mV. However, if PWDN_IOBRNOUT is set, this bit is high when the VDD5V pin is higher than the level indicated by VDDIO_BO, the brownout level of VDDIO.
3	**AVALID**	RW	0x0	Indicates VBus is valid for a A-peripheral, high if VBus greater than 2.0, low if VBus less than 0.8, otherwise unknown.
2	**BVALID**	RW	0x0	Indicates VBus is valid for a B-peripheral, high if VBus greater than 4.0, low if VBus less than 0.8, otherwise unknown.
1	**VBUSVALID**	RW	0x0	VBus Valid for USB OTG. See POWER_5VCTRL to enable and set threshold for comparison.
0	**SESSEND**	RW	0x0	Session End for USB OTG. 0 if VBus is greater than 0.8 V, 1 if VBus is less than 0.2 V, otherwise unknown. See POWER_5VCTRL to enable comparators.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.11. Temperature and Transistor Speed Control and Status Register Description

This register contains the setup and controls needed to measure die temperature and silicon speed.

HW_POWER_SPEEDTEMP 0x800440a0
HW_POWER_SPEEDTEMP_SET 0x800440a4
HW_POWER_SPEEDTEMP_CLR 0x800440a8
HW_POWER_SPEEDTEMP_TOG 0x800440aC

Table 993. HW_POWER_SPEEDTEMP

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SPEED_STS1								SPEED_STS2								RSRVD2				TEMP_STS				RSRVD1		SPEED_CTRL		TEMP_CTRL			

Table 994. HW_POWER_SPEEDTEMP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	**SPEED_STS1**	RO	0x0	Result from first speed sensor. This result is only valid when SPEEDCTRL=0b11; otherwise this field contains debug information from the switching DC-DC converters.
23:16	**SPEED_STS2**	RO	0x0	Result from second speed sensor. This circuit is simply a duplicate of speed sensor 1, and is included for redundancy. This result is only valid when SPEEDCTRL=0b11; otherwise this field contains debug information from the switching DC-DC converters.
15:12	**RSRVD2**	RO	0x0	Empty Description.

Table 994. HW_POWER_SPEEDTEMP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11:8	**TEMP_STS**	RO	0x0	Die temperature result: 0000: Temp less than -40 0001: Temp greater than -40 and less than or equal to -30 0010: Temp greater than -30 and less than or equal to -20 0011: Temp greater than -20 and less than or equal to -10 0100: Temp greater than -10 and less than or equal to 0 0101: Temp greater than 0 and less than or equal to 15 0110: Temp greater than 15 and less than or equal to 25 0111: Temp greater than 25 and less than or equal to 35 1000: Temp greater than 35 and less than or equal to 45 1001: Temp greater than 45 and less than or equal to 55 1010: Temp greater than 55 and less than or equal to 70 1011: Temp greater than 70 and less than or equal to 85 1100: Temp greater than 85 and less than or equal to 95 1101: Temp greater than 95 and less than or equal to 105 1110: Temp greater than 105 and less than or equal to 115 1111: Temp greater than 115 temp and less than 130 Answer invalid when temp above 130
7:6	**RSRVD1**	RO	0x0	Empty Description.
5:4	**SPEED_CTRL**	RW	0x0	Speed Control bits. 00: Speed sensor off, 0b01: Speed sensor enabled, 11: Enable speed sensor measurement. Every time a measurement is taken, the sequence of 0x00 ; 01 ; 11 must be repeated. This sequence should proceed no faster than 1.5 MHz to ensure proper operation.
3:0	**TEMP_CTRL**	RW	0x0	Control bits to enable temperature sensor: Bit 3: 1=PWD, 0=Operational Bit 2: 1=Shift warmer mode, 0=Shift cooler mode Bits 1:0: 00=No shift, 01=Shift by 1, 10=Shift by 1+2, 11=Shift by 1+2+3 (shift unit is 3 degrees C) Bits 2,1,0 are used only to cancel offsets in the system and should not be set at this time.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.12. Battery Level Monitor Register Description

This register provides brownout controls and monitors the battery voltage.

HW_POWER_BATTMONITOR 0x800440b0

Table 995. HW_POWER_BATTMONITOR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD3						BATT_VAL										RSRVD2						PWDN_BATTBRNOUT	BRWNOUT_PWD	RSRVD1				BRWNOUT_LVL			

Table 996. HW_POWER_BATTMONITOR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	**RSRVD3**	RO	0x0	Empty Description.
25:16	**BATT_VAL**	RW	0x0	Software should be configured to place the battery voltage in this register measured with an 8-mV LSB resolution. This value is used by the DC-DC converters and must be correct to ensure correct operation of the converters
15:10	**RSRVD2**	RO	0x0	Empty Description.
9	**PWDN_BATTBRNOUT**	RW	0x1	Powers down the device after the DC-DC converters complete startup if a battery brownout occurs before the system is completely initialized. This function is only active when 5V is not present. Additionally, software should clear this bit and disable this function after a battery brownout interrupt is enabled.
8	**BRWNOUT_PWD**	RW	0x0	Power-down bit for battery brownout detector.
7:4	**RSRVD1**	RO	0x0	Empty Description.
3:0	**BRWNOUT_LVL**	RW	0x0	The default setting of the brownout settings decode to a voltage as follows: Single-alkaline/NiMH = 0.8 V Dual-alkaline/NiMH = 1.6 V Li-Ion = 2.8 V The voltage level can be calculated for other values by the following equation: Single alkaline NiMH brownout voltage = 0.8V + 0.02 * BRWNOUT_LVL Dual alkaline NiMH brownout voltage = 1.6V + 0.04 * BRWNOUT_LVL Li-Ion brownout voltage = 2.8V + 0.04 * BRWNOUT_LVL

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.13. Power Module Reset Register Description

This register allows software to put the chip into the off state.

HW_POWER_RESET 0x800440c0
HW_POWER_RESET_SET 0x800440C4
HW_POWER_RESET_CLR 0x800440C8
HW_POWER_RESET_TOG 0x800440CC

Table 997. HW_POWER_RESET

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
UNLOCK																RSRVD1											PWD_OFF	POR	PWD	RST_DIG	RST_ALL

Table 998. HW_POWER_RESET Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	**UNLOCK**	RW	0x0	Write 0x3E77 to unlock this register and allow other bits to be changed. NOTE: This register must be unlocked on a write-by-write basis, so the UNLOCK bitfield must contain the correct key value during all writes to this register in order to update any other bitfield values in the register. KEY = 0x3E77 Key needed to unlock HW_POWER_RESET register.
15:5	**RSRVD1**	RO	0x0000	Empty Description.
4	**PWD_OFF**	RW	0x0	Optional bit to disable all paths to power off the chip except the watchdog timer. Setting this bit will be useful for preventing fast falling edges on the PSWITCH pin from resetting the chip. It may also be useful increasing system tolerance of noisy EMI environments.
3	**POR**	RW	0x0	Clears the persistent bits. Similar to removing and re-inserting the battery.
2	**PWD**	RW	0x0	Powers down the chip.
1	**RST_DIG**	RW	0x0	Resets all non-power module digital registers and reboots the CPU. It is necessary to clear EN_BATADJ and PWDN_5VBRNOUT before asserting this bit.
0	**RST_ALL**	RW	0x0	Resets all digital registers, including the power module, and reboots the CPU. It is not recommended to use RST_ALL, instead RST_DIG should be used to reset and reboot the device.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

31.8.14. Power Module Debug Register Description

Debug Register.

HW_POWER_DEBUG 0x800440d0
HW_POWER_DEBUG_SET 0x800440d4
HW_POWER_DEBUG_CLR 0x800440d8
HW_POWER_DEBUG_TOG 0x800440dC

Table 999. HW_POWER_DEBUG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSRVD0																											ENCTRLVBUS	VBUSVALIDPIOLOCK	AVALIDPIOLOCK	BVALIDPIOLOCK	SESSENDPIOLOCK

Table 1000. HW_POWER_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:5	**RSRVD0**	RO	0x0	Empty Description.
4	**ENCTRLVBUS**	RW	0x0	Use the ChrgVbus and DischrgVbus control from the USB controller (ARC) instead of the Power Control Register.
3	**VBUSVALIDPIOLOCK**	RW	0x0	Empty Description.
2	**AVALIDPIOLOCK**	RW	0x0	Empty Description.
1	**BVALIDPIOLOCK**	RW	0x0	Empty Description.
0	**SESSENDPIOLOCK**	RW	0x0	Empty Description.

DESCRIPTION:

Empty Description.

EXAMPLE:

```
Empty Example.
```

POWER XML Revision: 1.67

31.9. DC-DC Converter Efficiency

The graphs in this section show typical efficiency plots vs. the battery voltage for the DC-DC converters configured in different modes. These graphs show measurements made with typical devices at room temperature with specific static loads. Therefore, these graphs should not be interpreted as specifications, but as estimates of typical efficiencies under nominal conditions. It should be noted that the 24-MHz XTAL bias current is counted as a loss term in the efficiency calculation, and thus the light load efficiency curves are somewhat pessimistic.

31.9.1. DC-DC1 Mode 3 Efficiency

Figure 136 and Figure 137 show the efficiency of the DC-DC converter #1 in mode 3 versus battery voltage for several values of output power. The test conditions for the two graphs are identical except for the load to which the power is delivered. Because the power FET that connects the inductor to the DCDC_VDDIO output has a higher resistance than the FET that connects the inductor to the DCDC_VDDD output, the efficiency of the DC-DC converter is improved when the same power is delivered to the low-voltage output, VDDD, relative to the high-voltage output, VDDIO.

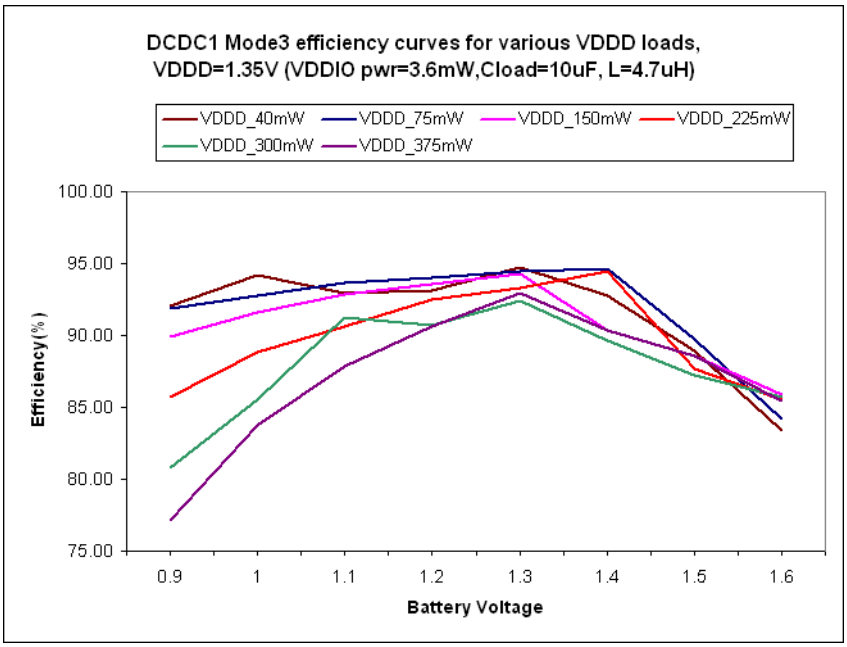


Figure 136. Efficiency of DC-DC #1 in Mode 3 for Various VDDD Loads

It should be noted that the boost configuration of DC-DC #1 could be the least efficient heavy load configuration for two reasons. First, Boost Mode power conversion only transfers power to the load during one phase of the converters charge/discharge cycle. Therefore, conservation of charge requires an inductor current that is higher than the load current, so I^2R losses in the power FETs that switch the inductor are increased relative to a Buck Mode configuration where current is transferred to the load during the entire charge/discharge cycle. Secondly, since mode 3 generates multiple outputs using one external inductor, the inductor current must increase

to support multiple load currents. Thus, I^2R losses are also increased when using the multiple output configurations relative to the configurations that have one inductor per output voltage. Therefore, mode 3 is primarily intended for lower power applications that use a single battery and single inductor to achieve an ultra small form factor.

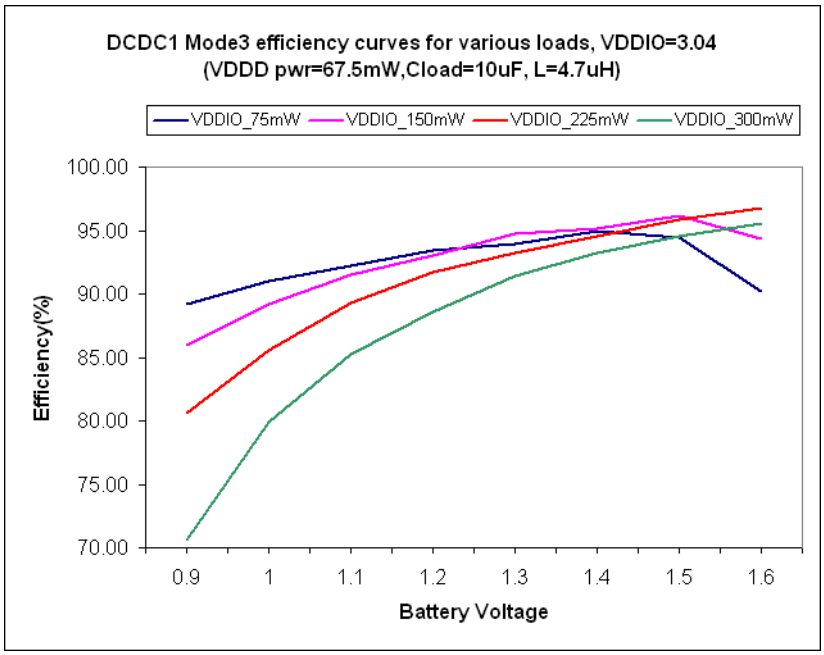


Figure 137. Efficiency of DC-DC #1 in Mode 3 for Various VDDIO Loads

31.9.2. DC-DC1 Mode 1 Efficiency

Figure 138 and Figure 139 show the efficiency of the DC-DC converter #1 in mode 1 versus battery voltage for several values of output power. The test conditions for the two graphs are identical except for the load to which the power is delivered. Because the power FET that connects the inductor to the DCDC_VDDIO output has a higher resistance than the FET that connects the inductor to the DCDC_VDDD output, the efficiency of the DC-DC converter is improved when the same power is delivered to the low-voltage output, VDDD, relative to the high-voltage output, VDDIO.

Mode 1 supports higher output power than mode 3 because mode 1 operates with a Li-Ion battery range. Thus, the DC-DC converter generally operates in a buck configuration, which causes inductor current to be reduced and thus I^2R losses in the FETs are higher. However, operating in buck mode shows decreased efficiency as the battery voltage rises, as there is capacitive loss driving the power FETs that increases as the battery voltage increases.

Additional efficiency measurements have shown that the use of a higher inductor value than 4.7 uH can improve efficiency a few percent under light load conditions.

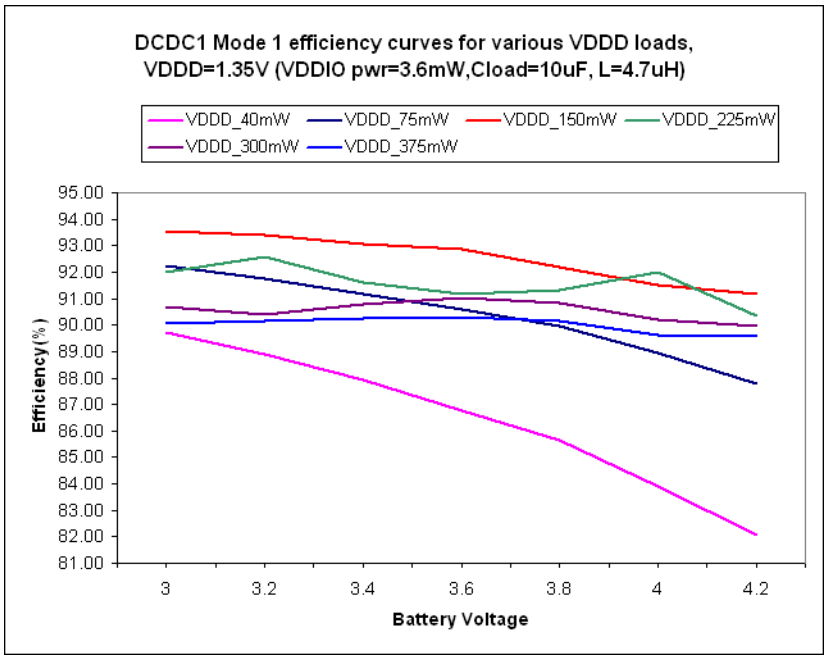


Figure 138. Efficiency of DC-DC #1 in Mode 1 for Various VDDD Loads

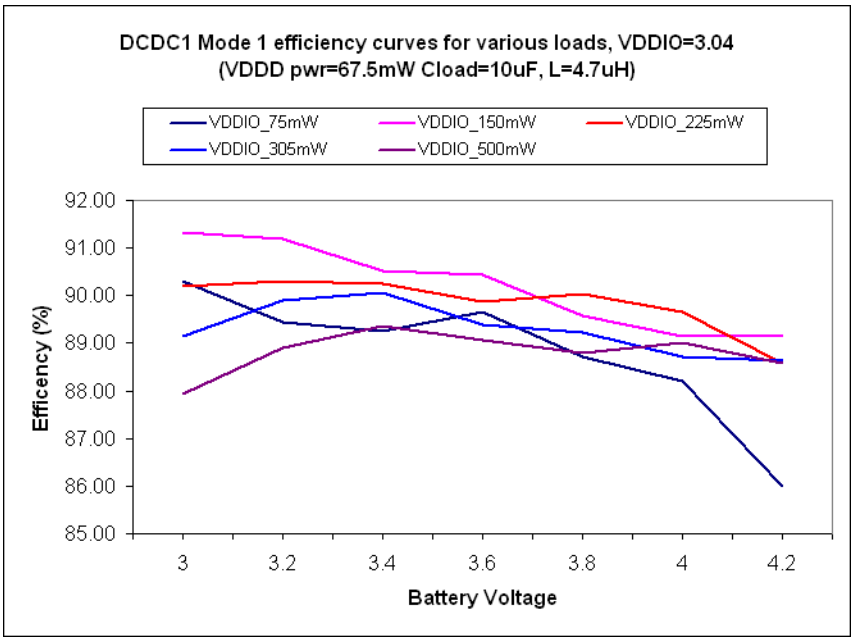


Figure 139. Efficiency of DC-DC #1 in Mode 1 for Various VDDIO Loads

31.9.3. DC-DC1 Mode 2/Mode 0 Efficiency

Figure 140 shows the efficiency for DC-DC #1 configured as a single-channel buck converter. Since DC-DC #1 is configured identically in modes 2 and 0, this graph is applicable in both modes.

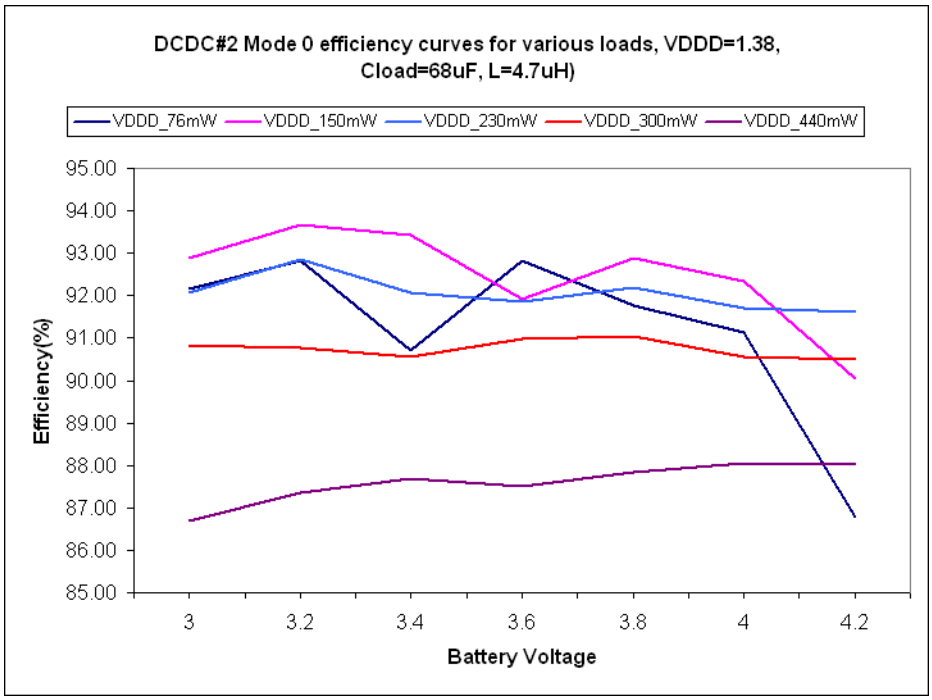


Figure 140. Efficiency of DC-DC #1 in Mode 2/Mode 0 for Various VDDD Loads

31.9.4. DC-DC2 Mode 2 Efficiency

Figure 141 shows the efficiency of DC-DC #2 for various values of output power. Since DC-DC #2 has a PFET with less on-resistance than the comparable DC-DC #1 PFET, and this configuration is a single-load converter, this mode is preferred for higher power VDDIO loads over mode 3 or 1.

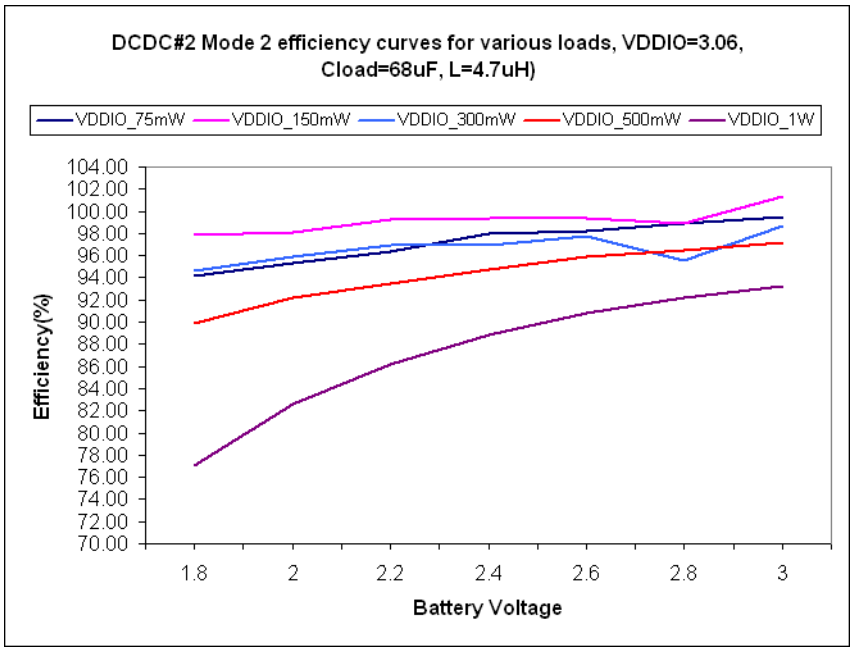


Figure 141. Efficiency of DC-DC #2 in Mode 2 for Various Current Loads

31.9.5. DC-DC2 Mode 0 Efficiency

Figure 142 shows the efficiency of DC-DC #2 for various values of output power. This converter configuration is intended for high power output applications, even as the Li-Ion battery voltage drops. For this reason, this converter is a buck-boost and can sustain a constant output voltage as the battery voltage decreases, even under heavy load conditions.

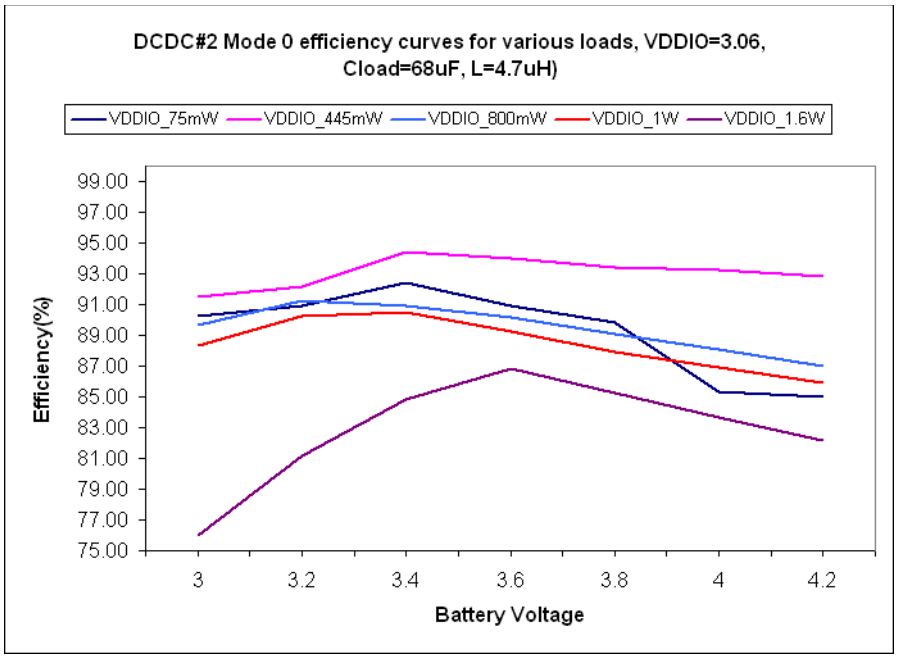


Figure 142. Efficiency of DC-DC #2 in Mode 0 for Various Current Loads

31.9.6. Max Power Out in Boost Modes

For a Boost Mode converter, the power delivered to the load can be estimated using the following equation:

$$P_{OUT} = \frac{V_{OUT}(V_{BAT}(1-D) - V_{OUT}(1-D)^2)}{R_P + R_L + D(R_N - R_P)}$$

In this equation, “D” represents the duty cycle of DC-DC converter, Vout represents the output voltage of the converter, Rn and Rp represent the on-resistance of the power FETs appropriate for the particular DC-DC converter configuration, and Rl represents the on-resistance of the DC-DC inductor.

For multiple boost output mode 3, this equation is a reasonable approximation if it is assumed that all the power is transferred to one of the loads. This equation leads to the following expression for the duty cycle that gives to the maximum power transferred to the load:

$$D_{MAX} = \frac{R_P - R_N\sqrt{1 + \frac{V_{BAT}}{V_{OUT}} \times \frac{(R_P - R_N)}{R_N}}}{R_P - R_N}$$

Substituting Dmax into the equation for Pout gives a maximum power that can be theoretically delivered to the load. Note that this equation was derived using only V=Ldi/dt and conservation of charge, and thus represents a theoretical maximum that neglects many significant sources of loss including frequency dependent core losses, bond wire ringing, switch non-overlap times, etc.

From an application point of view, it is probably not desirable to operate the converter at the maximum power output, since I^2R losses will also be maximized.

Therefore, it is recommended to constrain system power requirements to loads that can be supplied with efficiency ≥ 70% as shown in the graphs in Figure 136 Figure 137 and Figure 141.

31.9.7. Max Power Out in Buck Modes

Unlike the boost case, the maximum output power from a Buck Mode configuration is limited simply by voltage difference between the load and the battery divided by the PFET on-resistance.

$$P_{OUT} = \frac{V_{BAT} - V_{OUT}}{R_P + R_L}$$

Thus, a buck-mode configuration can better supply large currents to the load than boost-mode configurations.

32. BOOT MODES

This chapter describes the boot modes implemented on the STMP36xx. It includes sections on mode selection, the boot loader, and preparing bootable images.

32.1. Overview

The masked (on-chip) ROM-resident boot loader is responsible for booting from a subset of the peripherals attached to the STMP36xx. Peripherals supported by boot modes in current versions of the ROM include:

- NAND flash
- ATA
- I^2C EEPROM
- NOR flash
- Special JTAG debug spin loops
- SigmaTel-only factory test modes

Special terms used in this chapter include the following:

- **Authentication Signature**—A 4-byte value attached to each 508-byte block of boot commands. It is used to authenticate the contents of the 508-byte block of commands.
- **Cold Boot**—The first chip power-up after external power has been supplied to the chip.
- **Warm Boot**—Chip power-up after cold boot.
- **Recovery Mode**—A fail-safe ROM boot mode that uses the SigmaTel USB Boot Class.
- **POST (Power On Self Test)**—Tests the SRAM and provides repair vectors that may be used to repair the RAM. The repair vectors are placed into persistent bits.
- **Repair**—Repairs the SRAM by copying RAM repair vectors from persistent bits into the RAM repair registers.

32.2. Mode Selection

32.2.1. Boot Pins and Boot Modes

Table 1001 shows the pins used to determine the boot mode.

Table 1001. Boot Pins

100-PIN TQFP PIN#	169-PIN FPBGA PIN#	PIN LABEL	PINCTRL BANK:BIT	BOOT FUNCTION	BIT NAME
11	22	GPMI_D07	0:7	Enable RAM Repair	EREPAIR
10	20	GPMI_D06	0:6	Test Boot Mode Bit 2	TM2
9	18	GPMI_D05	0:5	Test Boot Mode Bit 1	TM1
8	16	GPMI_D04	0:4	Test Boot Mode Bit 0	TM0
7	14	GPMI_D03	0:3	Boot Mode Bit 3	BM3
6	12	GPMI_D02	0:2	Boot Mode Bit 2	BM2
3	8	GPMI_D01	0:1	Boot Mode Bit 1	BM1
2	6	GPMI_D00	0:0	Boot Mode Bit 0	BM0

These pads are powered during the initial loader sequence. The pads are enabled as GPIOs for sensing, and then disabled. However, the pads remain powered. The TBMx pins are not powered or configured as GPIO unless BMx=0XF.

The boot modes are shown in Table 1002.

Table 1002. Boot Modes

TBM2	TBM1	TBM0	BM3	BM2	BM1	BM0	PORT	BOOT MODE
x	x	x	0	0	0	0	USB	STMP Boot Class
x	x	x	0	0	0	1	I^2C	I^2C Master
x	x	x	0	0	1	0	-	Reserved
x	x	x	0	0	1	1	GPMI	ATA
x	x	x	0	1	0	0	GPMI	NAND 3.3 Volt
x	x	x	0	1	0	1	EMI	NOR
x	x	x	0	1	1	0	DEBUG0	Startup waits for JTAG debugger connection.
x	x	x	0	1	1	1	DEBUG1	Loader waits for JTAG debugger connection
x	x	x	1	0	0	0	-	Reserved
x	x	x	…	…	…	…	…	Reserved
x	x	x	1	1	1	0	-	Reserved
0	0	0	1	1	1	1	GPIO	Tester Loader (SIGMATEL USE ONLY)
0	0	1	1	1	1	1	GPIO	Burn in (SIGMATEL USE ONLY)
0	1	0	1	1	1	1	GPIO	ROM CRC (SIGMATEL USE ONLY)
0	1	1	1	1	1	1	GPIO	RAM Test (SIGMATEL USE ONLY)
1	0	0	1	1	1	1	GPIO	BIST (SIGMATEL USE ONLY)
1	0	1	1	1	1	1	UART	UART (SIGMATEL USE ONLY) (CURRENTLY NOT SUPPORTED) Will use standard SigmaTel boot images.
1	1	0	1	1	1	1	-	Reserved
1	1	1	1	1	1	1	-	Reserved

32.2.1.1. Persistent Bits

Persistent bits are used to control certain features in the ROM, as shown Table 1003. For more information on the persistent bits, see Chapter 19, "Real-Time Clock, Alarm, Watchdog, and Persistent Bits" on page 497.

Table 1003. Persistent Bits Used by the ROM

PERSISTENT BIT	BIT FIELD NAME	FUNCTION
HW_RTC_PERSISTENT2[31]	WBOOT	Warm boot.
HW_RTC_PERSISTENT0[31]	SDBOOT	SDRAM boot. If set, then the loader ignores the boot mode set by the boot pins. It will power up the SDRAM and jump to sdram_boot_strap().
HW_RTC_PERSISTENT0[24,23]	SDRAM_CS_HI, SDRAM_CS_LO	Two-bit binary encoding of the chip select the boot ROM should use when addressing the SDRAM. Valid values are 0, 1, 2, and 3.
HW_RTC_PERSISTENT0[22–19]	SDRAM_NDX_3,2,1,0	Four-bit binary encoding of the table index of SDRAM controller programming parameters. This table is stored internally in the ROM. A value of b'111 is an indication to the loader to disregard its internal table and use the values stored in the ram_patch[] single-entry table located in the SRAM section.bss.sdram.
HW_RTC_PERSISTENT0[18]	ETM_ENABLE	If set, the ROM enables the embedded trace macrocell (ETM) block.

Table 1004 shows how the persistent bits and the EREPAIR pin are used in the ROM.

Table 1004. RAM POST and Repair

TEST MODE	FPOST	FREPAIR	EREPAIR	USB	WBOOT	RUN POST	RUN REPAIR	DESCRIPTION
0	1	X	X	X	X	1	1	Perform POST and Repair every time. Don't copy repair vectors to persistent bits or set the warm boot.
0	0	0	0	X	X	0	0	No POST or Repair.
0	0	0	1	0	X	1	1	Perform POST and Repair, copy repair vectors to persistent bits, set Warm Boot persistent bit.
0	0	0	1	1	0	1	1	Perform POST and Repair, copy repair vectors to persistent bits, set Warm Boot persistent bit.
0	0	0	1	1	1	0	1	No POST. Copy repair vectors from persistent bits to SRAM.
0	0	1	X	0	X	1	1	Perform POST and Repair, copy repair vectors to persistent bits, set Warm Boot persistent bit.
0	0	1	X	1	0	1	1	Perform POST and Repair, copy repair vectors to persistent bits, set Warm Boot persistent bit.

Table 1004. RAM POST and Repair (Continued)

TEST MODE	FPOST	FREPAIR	EREPAIR	USB	WBOOT	RUN POST	RUN REPAIR	DESCRIPTION
0	0	1	X	1	1	0	1	No POST. Copy repair vectors from persistent bits to SRAM.
1	X	X	0	X	X	0	0	No POST, no Repair.
1	X	X	1	X	0	1	1	Perform POST and Repair, copy repair vectors to persistent bits, set Warm Boot persistent bit.
1	X	X	1	X	1	0	1	No POST. Copy repair vectors from persistent bits to SRAM.

Where:

TEST MODE =True(1) if one of the special boot modes is selected via boot mode pins, False (0) otherwise.

FPOST = True(1) if result of bit-wise or of FORCE_POST and ALT_FORCE_POST is True, False(0) otherwise.

FREPAIR = True (1) if result of bit-wise or of FORCE_REPAIR and ALT_FORCE_REPAIR is True, False(0) otherwise.

EREPAIR = Logical value of EREPAIR pin (GPIO0, bit 7).

USB = True(1) if USB connection is detected, False(0) otherwise.

WBOOT = Value of PERSISTENT2, bit 31

32.3. Boot Loader

The primary function of the boot loader is to load blocks of data into on-chip RAM. It is also capable of initializing a block of memory to a specified value. Data may be loaded/initialized to other parts of the memory map, provided they are accessible (the loader will blindly do as instructed, so beware).

The boot loader can initialize the EMI to support an external SDRAM through code loaded and executed from the boot device. Once initialized, the boot loader can load code and data directly to the SDRAM for subsequent execution.

The boot loader is invoked to load from one of the supported peripheral devices and reads a stream of commands from an encrypted and authenticated image found on the desired boot device. One command can fill a block of physical address space (any memory) or it can copy a block code/data from the peripheral storage device into any memory in the processors physical address space. At the end of loading an image, there is a command that can be used to branch into the loaded code and begin execution.

32.3.1. Transition from Boot Loader to Runtime Image

The boot loader is designed to ensure a well-defined hardware and software state before transitioning to the loaded runtime image using the following:

- The loader assumes the MMU is off. However, nothing precludes a boot image from containing one-shot code to turn it on. The loader does not turn off the MMU, so the user must take care when doing this.

- The loader turns on the instruction cache unless prohibited by laser fuse. The loader does not enable the data cache, but nothing precludes a boot image from containing one-shot code to turn it on. The loader does not turn off the data cache. The user must take care when doing this. Note: The MMU must be on for the data cache to be on.
- The runtime image may temporarily make use of the stacks and vectors left behind by the ROM, but it is expected that the runtime image will establish its own environment, and the ROM resources will then be discarded.
- FIQ and IRQ are turned off.
- Debug UART is disabled.
- The loader zeroes out the memory holding the encryption master key set.

32.3.2. Constructing Image to Be Loaded by Boot Loader

The image is stored in an encrypted form that includes an authenticating hash. SigmaTel supplies a program called *elftosb* to convert a fully resolved executable binary image into a boot image usable by the boot loader. A key set must be input to the *elftosb* program to properly encrypt and authenticate the image. A default key set is supplied with *elftosb*. The process of creating a boot loader image is shown in Figure 144.

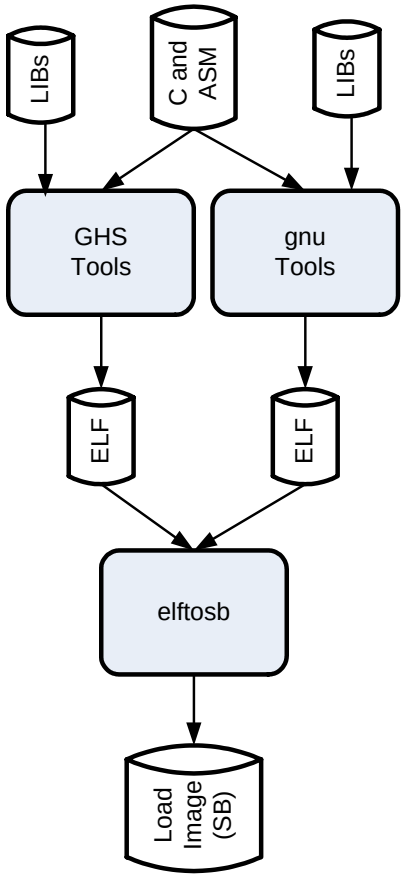


Figure 144. Creating a Boot Loader Image

32.3.3. On-Chip RAM Used by Boot Loader

The loader initializes the supervisor mode, IRQ, FIQ, and abort stacks. Together, these occupy approximately 2 Kbytes at address 0x0003D000. Customer code is free to change these assignments as soon as it begins execution.

In addition, for its own internal operation, the boot loader consumes, approximately 20 Kbytes located at 0x0003B000. This area must not be overloaded by a boot load operation, but can be used for any purpose immediately after the load completes.

32.4. Preparing Bootable Images

Preparing a bootable image for all boot modes (except Special NOR boot mode, as described below) includes the following high-level steps:

- Prepare the ELF file for the firmware that is to be booted by the STMP36xx ROM.
- Run the ELF file through the *elftosb* program (available from SigmaTel), which generates an encrypted SB file that can be booted from ROM.

Any additional requirements for individual boot modes are identified in sections that follow.

32.4.1. I^2C EEPROM Boot Mode

I^2C EEPROM boot mode does not require any special configuration. To boot from I^2C EEPROM, simply write the SB file to the appropriate device.

32.4.2. Regular NOR Boot Mode

Regular NOR boot mode does not require any special configuration. To boot from regular NOR, simply write the SB file to the appropriate device.

32.4.3. STMP (USB Recovery) Boot Mode

With STMP boot mode, also known as USB recovery mode, the user can configure a boot mode that always boots from USB. STMP boot mode requires that an SB file be sent to the 36xx and is otherwise treated like a boot from any other regular boot device.

32.4.4. ATA Hard Disk Drive Boot Mode

After preparing the encrypted image through *elftosb*, the user simply writes that image onto a CompactFlash/ATA device as consecutive sectors, starting at the location specified below.

To prepare a CF/ATA device for booting with the STMP36xx:

- Add a partition entry to the Master Boot Record with a drive type of 0x53 (‘S’).
- In all other ways, the partition table entry should meet all the specifications required for a partition table by applicable specifications from the personal computer world. In particular, BIOS specifications may detail the use of the partition table.
- Make the ‘S’ partition at least the size of the firmware plus four sectors, because the ROM expects to find the first byte of the .sb file at the beginning of the fifth sector of the S partition. (The first four sectors form the LDT, or Logical Drive Table, used by SigmaTel firmware to describe the S partition layout.)

This is all the information needed to locate and boot from a CF/ATA device. However, the ROM may parse the LDT header. The beginning of the LDT is at sector 0 of the S partition. The header is as follows:

```
struct
{
     uint32_t magic;              // 'STMP' or 0x504d5453
     uint32_t version;
     uint32_t sectorLBA// points to the current LBA + 4
     uint32_t reserved;
}
```

The version field is not relevant to booting from ROM. Figure 145 shows the ATA media layout.

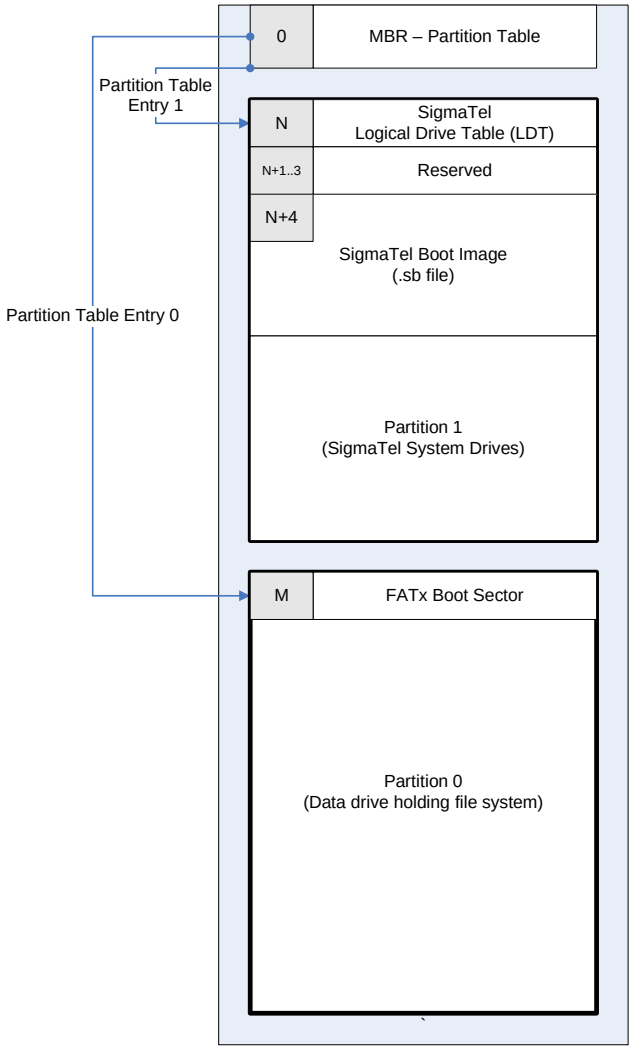


Figure 145. ATA Media Layout

32.4.5. Large-Block NAND Boot Mode

Large-Block NANDs use 2K pages and are a preferred NAND type for portable media players because of their large capacities. These devices generally do not allow partial writes, meaning that all 2K bytes must be written at one time.

After preparing the encrypted file through *elftosb*, the user writes the image onto the NAND as specified in this section.

To prepare a large-block NAND device for booting with the STMP36xx:

- Prepare and write a Boot Control Block (BCB) to block 0 of the NAND. CRC and ECC must be calculated for the pages written. (See the sample CRC code in Section 32.4.5.5).
- Program the NAND on a page-by-page basis (one page has four sectors) at the starting location specified in the BCB (see Section 32.4.5.4). Be sure to calculate the ECC for each sector written to the NAND (four times for each page) using the layout specified in this section. Also, be certain to fill out the metadata fields once per page and include that in the appropriate ECC calculation (see below).
- ECC is calculated over 512 bytes for the first three sectors of a page. ECC for the last sector should also include the physically first 7-byte metadata field for the page. Tip: Set the other three 7-byte metadata fields to 0, and the ECC can include those bytes, making for uniform DMA descriptors.

32.4.5.1. NAND Page Layout

Figure 146 illustrates an LB NAND with generalized page layout detail. (The specialized page layout required by the BCB is described in Section 32.4.5.4.)

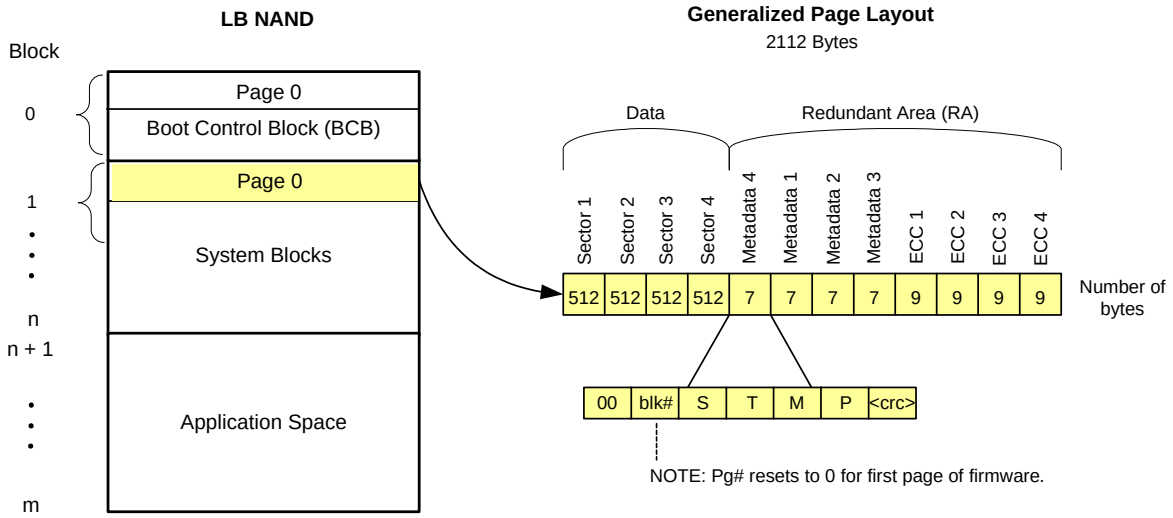


Figure 146. LB NAND with Generalized Page Layout Detail

LB NANDs are composed of blocks and typically have the following characteristics, as illustrated in Figure 146:

- 1 Block = 128 Kbytes, organized as 64 pages
- 1 Page = 2112 bytes, of which 2048 bytes are Data and 64 bytes are Redundant Area (RA)

- 1 Sector = 512 bytes of Data
- 1 RA Metadata field = 7 bytes
- 1 RA ECC field = 9 bytes
- Each page is composed of 4 Sectors, 4 Metadata fields, and 4 ECC fields.
- The Redundant Area of the Page includes the four 7-byte Metadata fields and the four 9-byte ECC fields.
- Physical Block 0 is the BCB (Boot Control Block)
- Page 0 of this block is defined in a special way (different from pages in System Blocks). See the specialized BCB page layout in Figure 148.
- Blocks 1 through n in the System Block are typically for the boot code, which is generated by taking the boot application, running it through the SigmaTel-supplied elftosb program with a cipher key to generate an encrypted file. Pages in these blocks follow the generalized page layout, as shown in Figure 146.

NOTE: Blocks in the System Block area start numbering from 0.

Blocks n+1 through the end of the device are typically for application space (usually a file system) and are not described here.

32.4.5.2. Data

In keeping with the ROM requirement that all data is accessed in 512 byte chunks, each 2112-byte page consists of four 512-byte data sectors and a 64-byte redundant area. LB NANDs require that the entire 2112 bytes be written at once.

The first three ECCs cover their respective 512-byte data areas only, as shown in Figure 147. The last 512 bytes are paired with the metadata bytes (M4 in Figure 147). The fourth ECC covers this entire grouping.

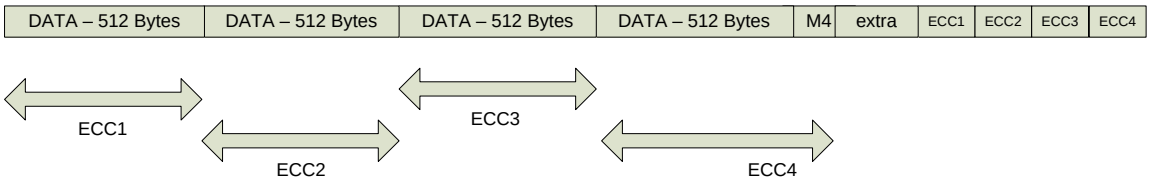


Figure 147. ECC Coverage for NAND Pages

32.4.5.3. Redundant Area (RA)

The redundant area consists of the last 64 bytes of the 2112 bytes in a page. The 64 redundant bytes start with the metadata bytes (M4 in Figure 147) and extend through ECC4. M4 refers to the seven metadata bytes in the RA immediately following the data. Only one metadata block is required per 2048 data bytes, which leaves 21 bytes of extra area for future use. Currently, these extra bytes are set to zero, but they can be any value as long as they are included in the ECC calculation. Future use of these bytes by SigmaTel is possible.

The redundant area the ROM expects for the boot image sectors is defined in Table 1005.

Table 1005. Redundant Area Byte Definitions

BYTE #	DEFINITION
2048	Block Status Byte = 0x00
2049	Block Number
2050	STMP Tag 0 'S'
2051	STMP Tag 1 'T'
2052	STMP Tag 2 'M'
2053	STMP Tag 3 'P'
2054	RA CRC
2055–2075	Extra area (0)
2076	RS_1_ECC1
2077	RS_1_ECC2
2078	RS_1_ECC3
2079	RS_1_ECC4
2080	RS_1_ECC5
2081	RS_1_ECC6
2082	RS_1_ECC7
2083	RS_1_ECC8
2084	RS_1_ECC9
2085	RS_2_ECC1
2086	RS_2_ECC2
2087	RS_2_ECC3
2088	RS_2_ECC4
2089	RS_2_ECC5
2090	RS_2_ECC6
2091	RS_2_ECC7
2092	RS_2_ECC8
2093	RS_2_ECC9
2094	RS_3_ECC1
2095	RS_3_ECC2
2096	RS_3_ECC3
2097	RS_3_ECC4
2098	RS_3_ECC5
2099	RS_3_ECC6
2100	RS_3_ECC7
2101	RS_3_ECC8
2102	RS_3_ECC9
2103	RS_4_ECC1
2104	RS_4_ECC2
2105	RS_4_ECC3
2106	RS_4_ECC4
2107	RS_4_ECC5
2108	RS_4_ECC6
2109	RS_4_ECC7
2110	RS_4_ECC8
2111	RS_4_ECC9

As shown in Table 1005, the first byte contains the Block Status Byte set to '0x00', which normally denotes a bad block, but when combined with the proper tag in bytes 2050–2053, is used to denote a proper boot image. The first byte in the metadata area is the block status byte, because that position is typically used by the factory to mark blocks as bad.

The second byte (Block Number) contains a counter that counts up from zero for each good block that the boot image resides in. The first block of sectors generally resides in Block 1. (Physical Block 1 maps to logical System Block 0.)

The next four bytes contain the boot tag, currently specified as BCB<space> or STMP.

The last byte includes a CRC that covers the previous six bytes (See Section 32.4.5.5).

Following that are ECC parity bytes that cover the 512 bytes of the sector, the seven bytes in the Metadata area, and (possibly) the extra bytes.

32.4.5.4. Boot Control Block Structure

The boot configuration block (BCB) resides on the NAND attached to CE0. The BCB is the first sector in the first good block. The data held in the BCB includes:

- Which NANDs are have the boot image on it (CE0, CE1, CE2, CE3)
- Sector and NAND of start of boot region
- Additionally, the BCB is marked with a signature, both in the sector and in the redundant area (Tag of 'BCB ') [The last character is a space]

Any other configuration information is stored on the following sectors in the same erase block.

Figure 148 shows a NAND with the specialized page layout required by the Boot Control Block.

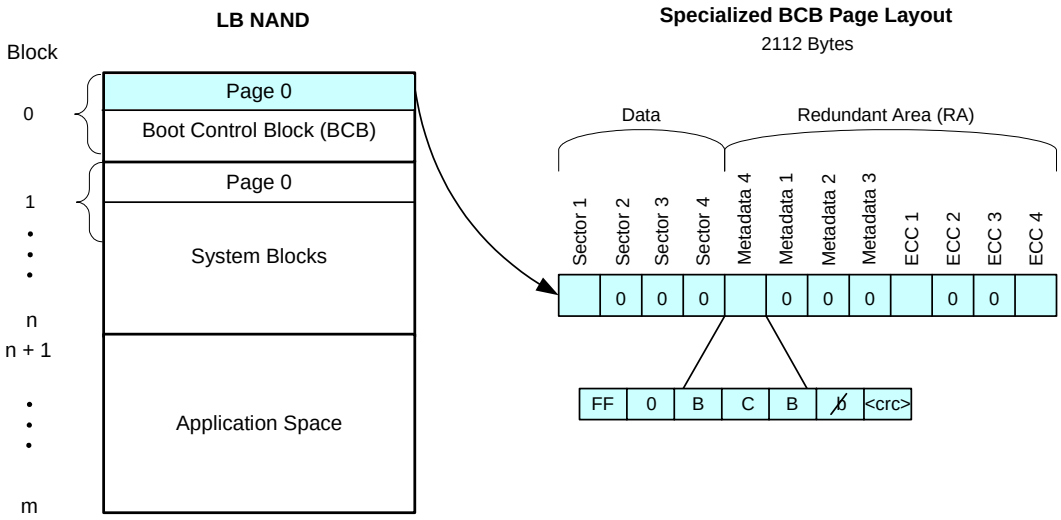


Figure 148. LB NAND with Specialized BCB Page Layout Detail

32.4.5.5. BCB Structure and Initialization

The following code describes the fields in the first sector of the BCB (block 0) on the NAND. The structure below is packed.

```
struct _bcb_sctruct
{
    u32    m_u32Signature1;
    struct
        {
            u16 m_u16Major;
            u16 m_u16Minor;
            u16 m_u16Sub;
        } BCBVersion;
    u32    m_u32NANDBitmap;//bit 0 == NAND 0, bit 1 == NAND 1, bit 2 = NAND 2, bit 3 = NAND3
    u32    m_u32Signature2;
    u32    m_u32Firmware_startingNAND;
    u32    m_u32Firmware_startingSector;
    u32    m_uSectorsInFirmware;
    u32    m_uFirmwareBootTag;
    struct
        {
            u16 m_u16Major;
            u16 m_u16Minor;
            u16 m_u16Sub;
        } FirmwareVersion;
    u32    Rsvd[10];                                // set to zero
    u32    m_u32Signature3;
} bcb;

#define BCB_SIGNATURE1    0x504d5453// 'STMP'
#define BCB_SIGNATURE2    0x32424342// 'BCB '
#define BCB_SIGNATURE3    0x41434143// 'CACA'

#define BCB_VERSION_MAJOR   0x0001
#define BCB_VERSION_MINOR   0x0000
#define BCB_VERSION_SUB     0x0000
```

It has been noted that since many of these elements are not aligned on word boundaries, the compiled code is larger to handle these offsets.

BCBVersion structure—Must be set to Major = 1, Minor = 0; Sub = anything.

u32NANDBitmap—32-bit word detailing the NANDs that are in the system. Bit0=NAND0; Bit1=NAND1; Bit2=NAND2; Bit3=NAND3. Therefore, a single NAND implementation would be initialized with 0x00000001.

m_u32Firmware_startingNAND—Which NAND holds the boot manager that will get downloaded. This is zero-based, so NAND0 will be 0, NAND1 will be 1, etc.

m_u32Firmware_startingSector—Which sector on the NAND to start with. Remember that sectors from the ROM's perspective are 512 bytes. Since the supported NANDs store data in 2K pages, this conversion will need a <<2 (or * 4).

m_uSectorsInFirmware—Number of sectors (512 byte chunks) to be read by the ROM.

m_uFirmwareBootTag—32-bit word consisting of STMP (stored as 0x504D5453)

FirmwareVersion—Not used by the ROM. Used for tracking the version of software.

Here is sample code to initialize the NAND BCB:

```
bcb.m_u32Signature1 = BCB_SIGNATURE1;

    bcb.BCBVersion.m_u16Major = BCB_VERSION_MAJOR;
    bcb.BCBVersion.m_u16Minor = BCB_VERSION_MINOR;
    bcb.BCBVersion.m_u16Sub = BCB_VERSION_SUB;

    bcb.m_u32NANDBitmap = 0x00000001;
    bcb.m_u32Signature2 = BCB_SIGNATURE2;
    bcb.m_u32Firmware_startingNAND = 0;
    bcb.m_u32Firmware_startingSector = 256;// assumes the f/w goes in first block after BCB
    bcb.m_uSectorsInFirmware = g_num_sectors;// number of 512 byte sectors in f/w. ROUND UP!
    bcb.m_uFirmwareBootTag = BCB_SIGNATURE1;

    bcb.FirmwareVersion.m_u16Major = 0;
    bcb.FirmwareVersion.m_u16Minor = 0;
    bcb.FirmwareVersion.m_u16Sub = 0;

    for (i=0; i < 10; i++)
    {
        bcb.Rsvd[i] = 0;
    }

    bcb.m_u32Signature3 = BCB_SIGNATURE3;
```

Here is sample CRC code:

```
u8 CRC(u8 *pRA)
{
    int
        i,
        wFFcnt = 0;
    u8
// this is better as global data, but shown inside this function for clarity.
        crcvalues[256] =
        {
            0x00,0x07,0x0E,0x09,0x1c,0x1b,0x12,0x15,0x38,0x3f,0x36,0x31,0x24,0x23,0x2a,0x2d,
            0x70,0x77,0x7e,0x79,0x6c,0x6b,0x62,0x65,0x48,0x4f,0x46,0x41,0x54,0x53,0x5a,0x5d,
            0xe0,0xe7,0xee,0xe9,0xfc,0xfb,0xf2,0xf5,0xd8,0xdf,0xd6,0xd1,0xc4,0xc3,0xca,0xcd,
            0x90,0x97,0x9E,0x99,0x8c,0x8b,0x82,0x85,0xa8,0xaf,0xa6,0xa1,0xb4,0xb3,0xba,0xbd,
            0xc7,0xc0,0xc9,0xce,0xdb,0xdc,0xd5,0xd2,0xff,0xf8,0xf1,0xf6,0xe3,0xe4,0xed,0xea,
            0xb7,0xb0,0xb9,0xbe,0xab,0xac,0xa5,0xa2,0x8f,0x88,0x81,0x86,0x93,0x94,0x9d,0x9a,
            0x27,0x20,0x29,0x2e,0x3b,0x3c,0x35,0x32,0x1f,0x18,0x11,0x16,0x03,0x04,0x0d,0x0a,
            0x57,0x50,0x59,0x5e,0x4b,0x4c,0x45,0x42,0x6f,0x68,0x61,0x66,0x73,0x74,0x7d,0x7a,
            0x89,0x8e,0x87,0x80,0x95,0x92,0x9b,0x9c,0xb1,0xb6,0xbf,0xb8,0xad,0xaa,0xa3,0xa4,
            0xf9,0xfe,0xf7,0xf0,0xe5,0xe2,0xeb,0xec,0xc1,0xc6,0xcf,0xc8,0xdd,0xda,0xd3,0xd4,
            0x69,0x6e,0x67,0x60,0x75,0x72,0x7b,0x7c,0x51,0x56,0x5f,0x58,0x4d,0x4a,0x43,0x44,
            0x19,0x1e,0x17,0x10,0x05,0x02,0x0b,0x0c,0x21,0x26,0x2f,0x28,0x3d,0x3a,0x33,0x34,
            0x4e,0x49,0x40,0x47,0x52,0x55,0x5c,0x5b,0x76,0x71,0x78,0x7f,0x6a,0x6d,0x64,0x63,
            0x3e,0x39,0x30,0x37,0x22,0x25,0x2c,0x2b,0x06,0x01,0x08,0x0f,0x1a,0x1d,0x14,0x13,
            0xae,0xa9,0xa0,0xa7,0xb2,0xb5,0xbc,0xbb,0x96,0x91,0x97,0x9f,0x8a,0x8d,0x84,0x38,
            0xde,0xd9,0xd0,0xd7,0xc2,0xc5,0xcc,0xcb,0xe6,0xe1,0xe8,0xef,0xfa,0xfd,0xf4,0xf3
        },
        temp,
        crc,
        newindex;

    for(crc=0, i=0; i < 6; i++)
    {
        temp = pRA[i];
        newindex = crc ^ temp;
        crc = crcvalues[newindex];

        if (temp == 0xFF)
          wFFcnt++;
    }

////////////////////////////////////////////////////////////////////////////////
    //  If the RA is all FFs, it's probably erased, so the CRC byte will contain 0xFF.
    //  To match that, we force the computation to 0xFF, here.
    ////////////////////////////////////////////////////////////////////////////

    if (wFFcnt == 6)
      crc = 0xFF;

    return (crc);
}
```

32.4.5.6. Data Organization in Multiple NANDs

The first sector of the boot image is in the first good block found in the array of NANDs, starting at NAND0 and progressing to whatever other NANDs are reported to be present in the BCB, as illustrated in Figure 149. The next sector is drawn from the same block, until all sectors are drawn from the block. Then, the block from the next NAND is read, then the next NAND, etc., returning to NAND0 and repeating the process. If a block is determined to be bad (either the Block Status Byte of the block is marked bad or fails the ECC), then the algorithm skips to the next NAND, etc.

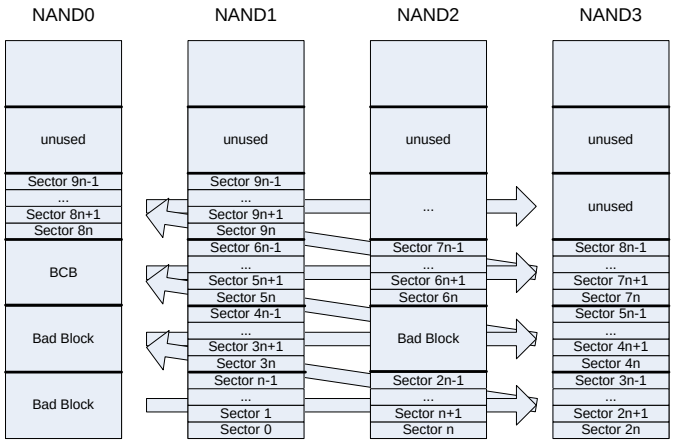


Figure 149. Data Organization in Multiple NANDs

33. REGISTER MACRO USAGE

This chapter provides background on the STMP36xx register set and illustrates a consistent use of the C macros for registers. The examples provided here show how to use the hardware register macros generated from the XML chip database.

33.1. Definitions

```
////////////////////////////////////////////////////////////////////////////////

// These macros will be generated from the XML in the future
#define BF_GPMI_CTRL0_SFTRST_V(v)            (BV_GPMI_CTRL0_SFTRST__##v << 31)
#define BF_GPMI_CTRL0_CLKGATE_V(v)           (BV_GPMI_CTRL0_CLKGATE__##v << 30)
#define BF_GPMI_CTRL0_RUN_V(v)               (BV_GPMI_CTRL0_RUN__##v << 29)
#define BF_GPMI_CTRL0_UDMA_V(v)              (BV_GPMI_CTRL0_UDMA__##v << 26)
#define BF_GPMI_CTRL0_COMMAND_MODE_V(v)      (BV_GPMI_CTRL0_COMMAND_MODE__##v << 24)
#define BF_GPMI_CTRL0_WORD_LENGTH_V(v)       (BV_GPMI_CTRL0_WORD_LENGTH__##v << 23)
#define BF_GPMI_CTRL0_LOCK_CS_V(v)           (BV_GPMI_CTRL0_LOCK_CS__##v << 22)
#define BF_GPMI_CTRL0_ADDRESS_V(v)           (BV_GPMI_CTRL0_ADDRESS__##v << 17)
#define  BF_GPMI_CTRL0_ADDRESS_INCREMENT_V(v)  (BV_GPMI_CTRL0_ADDRESS_INCREMENT__##v  <<
16)

#define BF_TIMROT_TIMCTRLn_SELECT_V(v)       (BV_TIMROT_TIMCTRLn_SELECT__##v << 0)

// These macros will be included in regs.h in the future
#define OR2(b,f1,f2)        (b##_##f1 | b##_##f2)
#define OR3(b,f1,f2,f3)     (b##_##f1 | b##_##f2 | b##_##f3)
#define OR4(b,f1,f2,f3,f4)  (b##_##f1 | b##_##f2 | b##_##f3 | b##_##f4)


////////////////////////////////////////////////////////////////////////////////
// Prototypes
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
// Variables
////////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////////
//! \brief  Provides examples of how to use the register access macros.
//!
//! \fntype Function
//!
//! Provides examples of how to use the register access macros.
////////////////////////////////////////////////////////////////////////////////
void hw_regs_Example(void)
{
    int i, iMode = 0, iRun = 0;

//
```

33.2. Background

The STMP36xx SOC is built on a 32-bit architecture using an ARM926 core. All hardware blocks are controlled and accessed through 32-bit wide registers. The design of these registers is maintained in an XML database that is part of the overall chip design. As part of the chip build process, a set of C include files are generated from the XML register descriptions. These include files provide a consistent set of C defines and macros that should be used to access the hardware registers.

The STMP36xx SOC has a complex architecture that uses multiple buses to segment I/O traffic and clock domains. To facilitate low power consumption, clocks are set to just meet application demands. In general, the I/O buses and associated hardware blocks run at speeds much slower than the CPU. As a result, reading a hardware register incurs a potentially large number of wait cycles, as the CPU must wait for the register data to travel multiple buses and bridges. The SOC does provide write buffering, meaning the CPU does not wait for register write transactions to complete. From the CPU perspective, register writes occur much faster than reads.

Most of the 32-bit registers are subdivided into smaller functional fields. These bit fields can be any number of bits wide and are usually packed. Thus, most fields do not align on byte or half-word boundaries.

A common operation is to update one field without disturbing the contents of the remaining fields in the register. Normally, this requires a read-modify-write (RMW) operation, where the CPU reads the register, modifies the target field, then writes the results back to the register. As already noted, this is an expensive operation in terms of CPU cycles, because of the initial register read.

To address this issue, most hardware registers are implemented as a set, including registers that can be used to either set, clear, or toggle (SCT) individual bits of the primary register. When writing to an SCT register, all bits set to 1 perform the associated operation on the primary register, while all bits set to 0 are not affected. The SCT registers always read back zero, and should be considered write-only. The SCT registers are not implemented if the primary register is read-only.

With this architecture, it is possible to update one or more fields using only register writes. First, all bits of the target fields are cleared by a write to the associated clear register, then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the zero state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (i.e., one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read/modify/write operations. When atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

33.3. Naming Convention

The generated include files and macros follow a consistent naming convention that matches the SOC documentation. This prevents name-space collisions and makes the macros easier to remember.

```
//
// The include file for a specific hardware module is named:
//
//     regs<module>.h
//
// Every register has an associated typedef that provides a C definition of
// the register.  The definition is always a union of a 32-bit unsigned int
// (i.e., reg32_t), and an anonymous bitfield structure.
//
//     hw_<module>_<regname>_t
//
// Macros and defines that relate to a register as a whole are named:
//
//     HW_<module>_<regname>_ADDR
//     HW_<module>_<regname>_<SET | CLR | TOG>_ADDR
//         - defines for the indicated register address
//
//     HW_<module>_<regname>
//         - a define for accessing the primary register using the typedef.
//           Should be used as an rvalue (i.e., for reading), but avoided as
//           an lvalue (i.e., for writing). Will usually generate RMW when
//           used as an lvalue.
//
//     HW_<module>_<regname>_RD()
//     HW_<module>_<regname>_WR()
//         - macros for reading/writing the primary register as a whole
//
```

```
//      HW_<module>_<regname>_<SET | CLR | TOG>()
//          - macros for writing the associated set | clear | toggle registers
//
// Macros and defines that relate to the fields of a register are named:
//
//      BM_<module>_<regname>_<field>
//      BP_<module>_<regname>_<field>
//          - defines for the field's bit mask and bit position
//
//      BF_<module>_<regname>_<field>()
//      BF_<module>_<regname>_<field>_V(<valuename>)
//          - macros for generating a bitfield value. The parameter is masked
//            and shifted to the field position.
//
//      BW_<module>_<regname>_<field>()
//          - macro for writing a bitfield. Usually expands to a CS operation.
//            Not generated for read-only fields.
//
//      BV_<module>_<regname>_<field>__<valuename>
//          - define equates to an unshifted named value for the field
//
// Some hardware modules repeat the same register definition multiple times.  An
// example is a block that implements multiple channels.  For these registers,
// the name adds a lowercase 'n' after the module, and the HW_ macros take a
// numbered parameter to select the channel (or instance).  This allows these
// macros to be used in for loops.
//
//      HW_<module>n_<regname><macrotype>(n, ...)
//          - the n parameter must evaluate to an integer, and selects the channel
//            or instance number.
//
// The regs.h include file provides several “generic” macros that can be used
// as an alternate syntax for the various register operations.  Because most
// operations involve using two or more of the above defines/macros, the <module>,
// <regname> and <field> are often repeated in a C expression. The generic
// macros provide shorthand to avoid the repetition.  Refer to the following
// examples for the alternate syntax.
```

33.4. Examples

The following examples show how to code common register operations using the predefined include files. Each example shows preferred and alternate syntax and also shows constructs to avoid. Summaries are provided toward the end.

The examples are valid C and will compile without errors. The reader is encouraged to compile this file and examine the resulting assembly code.

33.4.1. Setting 1-Bit Wide Field

```
// Preferred (one atomic write to SET register)
HW_GPMI_CTRL0_SET(BM_GPMI_CTRL0_UDMA);

// Alternate (same as above, just different syntax)
BF_SET(GPMI_CTRL0, UDMA);

// Avoid
BW_GPMI_CTRL0_UDMA(1);              // writes 1 to _CLR then 1 to _SET register
BF_WR(GPMI_CTRL0, UDMA, 1);         // same as above, just different syntax
HW_GPMI_CTRL0.B.UDMA = 1;             // RMW
```

33.4.2. Clearing 1-Bit Wide Field

```
// Preferred (one atomic write to _CLR register)
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_DEV_IRQ_EN);

// Alternate (same as above, just different syntax)
BF_CLR(GPMI_CTRL0, DEV_IRQ_EN);

// Avoid
BW_GPMI_CTRL0_DEV_IRQ_EN(0);        // writes 1 to _CLR then 0 to _SET register
BF_WR(GPMI_CTRL0, DEV_IRQ_EN, 0);   // same as above, just different syntax
HW_GPMI_CTRL0.B.DEV_IRQ_EN = 0;       // RMW
```

33.4.3. Toggling 1-Bit Wide Field

```
// Preferred (one atomic write to _TOG register)
HW_GPMI_CTRL0_TOG(BM_GPMI_CTRL0_RUN);

// Alternate (same as above, just different syntax)
BF_TOG(GPMI_CTRL0, RUN);

// Avoid
HW_GPMI_CTRL0.B.RUN ^= 1;             // RMW
```

33.4.4. Modifying n-Bit Wide Field

```
// Preferred (does CS operation or byte/halfword write if the field is
//   8 or 16 bits wide and properly aligned)
BW_GPMI_CTRL0_COMMAND_MODE(BV_GPMI_CTRL0_COMMAND_MODE__READ_AND_COMPARE);
BW_GPMI_CTRL0_COMMAND_MODE(iMode);
BW_GPMI_CTRL0_XFER_COUNT(2);        // this does a halfword write

// Alternate (same as above, just different syntax)
BF_WR(GPMI_CTRL0, COMMAND_MODE, BV_GPMI_CTRL0_COMMAND_MODE__READ_AND_COMPARE);
BF_WR(GPMI_CTRL0, COMMAND_MODE, iMode);
BF_WR(GPMI_CTRL0, XFER_COUNT, 2);   // this does a halfword write

// Avoid (RMW)
HW_GPMI_CTRL0.B.COMMAND_MODE = BV_GPMI_CTRL0_COMMAND_MODE__READ_AND_COMPARE;
HW_GPMI_CTRL0.B.COMMAND_MODE = iMode;
```

33.4.5. Modifying Multiple Fields

```
// Preferred (explicit CS operation)
HW_GPMI_CTRL0_CLR( OR3(BM_GPMI_CTRL0, RUN, DEV_IRQ_EN, COMMAND_MODE) );
            HW_GPMI_CTRL0_SET(    OR3(BF_GPMI_CTRL0,    RUN(iRun),    DEV_IRQ_EN(1),
COMMAND_MODE_V(READ_AND_COMPARE)) );

// Alternate (same as above, just different syntax)
            BF_CS3(GPMI_CTRL0,    RUN,    iRun,    DEV_IRQ_EN,    1,    COMMAND_MODE,
BV_GPMI_CTRL0_COMMAND_MODE__READ_AND_COMPARE);

// Avoid (multiple RMW - the C compiler does NOT merge into one RMW)
HW_GPMI_CTRL0.B.RUN          = iRun;
HW_GPMI_CTRL0.B.DEV_IRQ_EN   = 1;
HW_GPMI_CTRL0.B.COMMAND_MODE = BV_GPMI_CTRL0_COMMAND_MODE__READ_AND_COMPARE;
```

33.4.6. Writing Entire Register (All Fields Updated at Once)

```
// Preferred
HW_GPMI_CTRL0_WR(BM_GPMI_CTRL0_SFTRST); // all other fields are set to 0

// Alternate (same as above, just different syntax)
HW_GPMI_CTRL0.U = BM_GPMI_CTRL0_SFTRST;
```

33.4.7. Reading a Bit Field

```
// Preferred
iRun = HW_GPMI_CTRL0.B.RUN;

// Alternate (same as above, just different syntax)
iRun = BF_RD(GPMI_CTRL0, RUN);

// Verbose Alternate (example of using bit position (BP_) define)
iRun = (HW_GPMI_CTRL0_RD() & BM_GPMI_CTRL0_RUN) >> BP_GPMI_CTRL0_RUN;
```

33.4.8. Reading Entire Register

```
0 // Preferred
    i = HW_GPMI_CTRL0_RD();

    // Alternate (same as above, just different syntax)
    i = HW_GPMI_CTRL0.U;
```

33.4.9. Accessing Multiple Instance Register

```
    // Preferred
    for (i = 0; i < HW_TIMROT_TIMCTRLn_COUNT; i++)
    {
        // Set 1-bit wide field
        HW_TIMROT_TIMCTRLn_SET(i, BM_TIMROT_TIMCTRLn_IRQ_EN);

        // Write n-bit wide field
        BW_TIMROT_TIMCTRLn_PRESCALE(i, BV_TIMROT_TIMCTRLn_PRESCALE__DIV_BY_1);

        // Write multiple fields
        HW_TIMROT_TIMCTRLn_CLR(i, OR2(BM_TIMROT_TIMCTRLn, RELOAD, SELECT));
                        HW_TIMROT_TIMCTRLn_CLR(i,   OR2(BF_TIMROT_TIMCTRLn,   RELOAD(1),
SELECT_V(1KHZ_XTAL)));

        // Read a field
        iRun = HW_TIMROT_TIMCTRLn(i).B.IRQ;
    }

    // Alternate (same as above, just different syntax)
    for (i = 0; i < HW_TIMROT_TIMCTRLn_COUNT; i++)
    {
        // Set 1-bit wide field
        BF_SETn(TIMROT_TIMCTRLn, i, IRQ_EN);

        // Write n-bit wide field
        BF_WRn(TIMROT_TIMCTRLn, i, PRESCALE, BV_TIMROT_TIMCTRLn_PRESCALE__DIV_BY_1);

        // Write multiple fields
                                BF_CS2n(TIMROT_TIMCTRLn,    i,    RELOAD,    1,    SELECT,
BV_TIMROT_TIMCTRLn_SELECT__1KHZ_XTAL);

        // Read a field
        iRun = BF_RDn(TIMROT_TIMCTRLn, i, IRQ);
    }
```

33.4.10. Correct Way to Soft Reset a Block

```
    // A soft reset can take multiple clocks to complete, so do NOT gate the
    // clock when setting soft reset. The reset process will gate the clock
    // automatically. It is safe to issue these writes back-to-back.
    HW_GPMI_CTRL0_SET(BM_GPMI_CTRL0_SFTRST);
    HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST | BM_GPMI_CTRL0_CLKGATE);
```

33.5. Summary Preferred

```
    // Setting, clearing, toggling 1-bit wide field
    HW_GPMI_CTRL0_SET(BM_GPMI_CTRL0_UDMA);
    HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_DEV_IRQ_EN);
    HW_GPMI_CTRL0_TOG(BM_GPMI_CTRL0_RUN);

    // Modifying n-bit wide field
    BW_GPMI_CTRL0_XFER_COUNT(2);

    // Modifying multiple fields
    HW_GPMI_CTRL0_CLR( OR3(BM_GPMI_CTRL0, RUN, DEV_IRQ_EN, COMMAND_MODE) );
                HW_GPMI_CTRL0_SET(    OR3(BF_GPMI_CTRL0,    RUN(iRun),    DEV_IRQ_EN(1),
COMMAND_MODE_V(READ_AND_COMPARE)) );

    // Reading a bitfield
    iRun = HW_GPMI_CTRL0.B.RUN;

    // Writing or reading entire register (all fields updated at once)
    HW_GPMI_CTRL0_WR(BM_GPMI_CTRL0_SFTRST);
    i = HW_GPMI_CTRL0_RD();
```

33.6. Summary Alternate Syntax

```
    // Setting, clearing, toggling 1-bit wide field
    BF_SET(GPMI_CTRL0, UDMA);
    BF_CLR(GPMI_CTRL0, DEV_IRQ_EN);
    BF_TOG(GPMI_CTRL0, RUN);

    // Modifying n-bit wide field
    BF_WR(GPMI_CTRL0, XFER_COUNT, 2);

    // Modifying multiple fields
                BF_CS3(GPMI_CTRL0,    RUN,    iRun,    DEV_IRQ_EN,    1,    COMMAND_MODE,
BV_GPMI_CTRL0_COMMAND_MODE__READ_AND_COMPARE);

    // Reading a bitfield
    iRun = BF_RD(GPMI_CTRL0, RUN);

    // Writing or reading entire register (all fields updated at once)
    HW_GPMI_CTRL0.U = BM_GPMI_CTRL0_SFTRST;
    i = HW_GPMI_CTRL0.U;
```

33.7. Assembly Example

```
//
// The generated include files are safe to use with assembly code as well. Not
// all of the defines make sense in the assembly context, but many should prove
// useful.
//
//     HW_<module>_<regname>_ADDR
//     HW_<module>_<regname>_<SET | CLR | TOG>_ADDR
//         - defines for the indicated register address
//
//     BM_<module>_<regname>_<field>
//     BP_<module>_<regname>_<field>
//         - defines for the field's bit mask and bit position
//
//     BF_<module>_<regname>_<field>()
//     BF_<module>_<regname>_<field>_V(<valuename>)
//         - macros for generating a bitfield value. The parameter is masked
//           and shifted to the field position.
//
//     BV_<module>_<regname>_<field>__<valuename>
//         - define equates to an unshifted named value for the field
//
// 6.1 Take GPMI block out of reset and remove clock gate.
// 6.2 Write a value to GPMI CTRL0 register. All other fields are set to 0.
#pragma asm
    ldr    r0, =HW_GPMI_CTRL0_CLR_ADDR
    ldr    r1, =BM_GPMI_CTRL0_SFTRST | BM_GPMI_CTRL0_CLKGATE
    str    r1, [r0]

    ldr    r0, =HW_GPMI_CTRL0_ADDR
    ldr    r1, =BF_GPMI_CTRL0_COMMAND_MODE_V(READ_AND_COMPARE)
    str    r1, [r0]
#pragma endasm
}

////////////////////////////////////////////////////////////////////////////////
//! \brief  Standalone application main entry point.
//!
//! \fntype Function
//!
//! Provides main entry point when building as a standalone application.
//! Simply calls the example register access function.
////////////////////////////////////////////////////////////////////////////////
void main(void)
{
    hw_regs_Example();
}
////////////////////////////////////////////////////////////////////////////////
// End of file
////////////////////////////////////////////////////////////////////////////////
//! }@
```

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

34. MEMORY MAP

Table 1006 shows the memory map for the STMP36xx.

Table 1006. STMP36xx Memory Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
00		Alias space for copies of on-chip SRAM												256 Kbyte On-Chip SRAM																	
01		00	256MB SDRAM/NOR/SYNC FLASH Chip Select 0																												
		01	256MB SDRAM/NOR/SYNC FLASH Chip Select 1																												
		10	256MB SDRAM/NOR/SYNC FLASH Chip Select 2																												
		11	256MB SDRAM/NOR/SYNC FLASH Chip Select 3																												
100		XXXXXXXXX										00		APBH Decoder 0 Interrupt Collector		XXXXX				Register Selects				SET CLEAR TOGGLE				Byte			
														APBH Decoder 1 APBH DMA																	
														APBH Decoder 2 HWECC																	
														APBH Decoder 3 GPMI																	
														APBH Decoder 4 NEWSPI																	
														APBH Decoder 5 MEMCPY																	
														APBH Decode 6 PINCNTRL																	
														APBH Decoder 7 DIGCTL																	
														APBH Decode 8 EMI																	
														APBH Decode 9 APBX DMA																	
														APBH Decode A-E		Default APBH SLAVE Decode A-E unused															
														APBH Decode F		Simulation termination and printf Registers. Default APBH SLAVE on real chip.															

OFFICIAL PRODUCT DOCUMENTATION 5/3/06

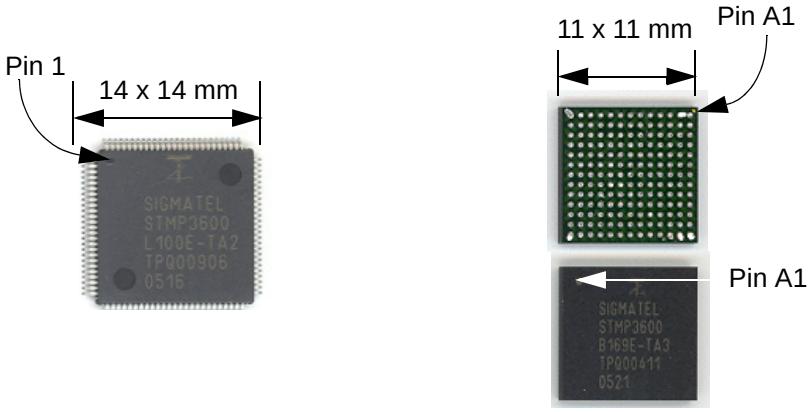
Table 1006. STMP36xx Memory Map (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
100			XXXXXXXXX									01		APBX 0 CLKCTRL				XXXX				Register Selects						SET CLEAR TOGGLE		Byte	
														APBX 1 POWER																	
														APBX 2 AUDIOOUT																	
														APBX 3 AUDIOIN																	
														APBX 4 LRADC																	
														APBX 5 SPDIF																	
														APBX 6 I2C																	
														APBX 7 RTC																	
														APBX 8 LCD IF																	
														APBX 9 PWM																	
														APBX A TIMROT																	
														APBX B UARTA																	
														APBX C UARTB																	
														APBX D DRI																	
														APBX E IRDA																	
														APBX F USB PHY																	
												10		XXXXXXXXXX										OTG USB Registers							
												11		XXXX				Default First-Level Page Table16KB													
101			AHB Default Slave																												
11			XXXXXXXXXXXXX													64K“on-chip” ROM or Interrupt Vector Register															

35. PIN DESCRIPTIONS

This chapter provides various views of the pinout for the STMP36xx. Section 35.1.1 details the 100-pin TQFP package pinout. Section 35.1.2 details the BGA pinout. Functional subsets of the pinout are listed in Section 35.2.

Table 1007 lists the abbreviations used in the pin tables in this chapter.



100 TQFP 169 fpBGA

Note: For additional package measurements, see Chapter 36, "Package Drawings" on page 843.

Figure 150. Chip Package Photographs

35.1. Pin Placement and Definitions

Table 1007. Nomenclature for Pin Tables

Type	Description
A	Analog pin
I	Input pin
I/O	Input/output pin
O	Output pin
P	Power pin

Module	Description
CODEC	Analog pins
DCDC	DC-DC Converter pins
EMI-SD	External Memory Interface pins (SDRAM)
EMI-NOR	External Memory Interface pins (NOR)
ETM	Embedded Trace Macrocell
GPIO	General-Purpose Input/Output pins
GPMI	General-Purpose Media Interface (NAND/ATA/CMOS)
HP	Head Phones
I^2C	I^2C pins
LCDIF	LCD Interface
POWER	Power pins
PWM	Pulse Width Modulator
SSP	Synchronous Serial Port pins
SYSTEM	System pins
TIMER	Timer pins
UART	Either debug or application UARTs
USB	USB pins

Note:Almost all digital pins are powered down (i.e., high-impedance) at reset, until reprogrammed. The only exceptions are: TESTMODE, ONCE_DSI, ONCE_DSK, ONCE_DSO, ONCE_DRN; these pins are always active.

35.1.1. Pin Definitions for 100-Pin TQFP Package

Table 1008. Pin Definitions—100-Pin TQFP Package

Pin Number	Pin Name	Module	Type	Pin Mux	Description
1	GPMI_RDY2	GPMI	I/O	0	ATA DMACK or NAND2 Ready/Busy#
		PINCTRL	I/O	3	GPIO0[20]
2	GPMI_D00	GPMI	I/O	0	ATA/NAND Data 0
		PINCTRL	I/O	3	GPIO0[0]
3	GPMI_D01	GPMI	I/O	0	ATA/NAND Data 1
		PINCTRL	I/O	3	GPIO0[1]
4	VDDIO1	POWER	P	0	Digital I/O Power 1 / LRADC7–2
5	VSSD1	POWER	P	0	Digital Ground 1
6	GPMI_D02	GPMI	I/O	0	ATA/NAND Data 2
		PINCTRL	I/O	3	GPIO0[2]
7	GPMI_D03	GPMI	I/O	0	ATA/NAND Data 3
		PINCTRL	I/O	3	GPIO0[3]
8	GPMI_D04	GPMI	I/O	0	ATA/NAND Data 4
		PINCTRL	I/O	3	GPIO0[4]
9	GPMI_D05	GPMI	I/O	0	ATA/NAND Data 5
		PINCTRL	I/O	3	GPIO0[5]
10	GPMI_D06	GPMI	I/O	0	ATA/NAND Data 6
		PINCTRL	I/O	3	GPIO0[6]
11	GPMI_D07	GPMI	I/O	0	ATA/NAND Data 7
		PINCTRL	I/O	3	GPIO0[7]
12	VDDD1	POWER	P		Digital Core Power 1 / LRADC7–1
13	VSSD2	POWER	P		Digital Ground 2
14	GPMI_IRQ	GPMI	I	0	ATA INTRQ or NAND1 Ready/Busy#
		PINCTRL	I/O	3	GPIO0[16]
15	GPMI_RDY	GPMI	I	0	ATA IORDY:DSTROBE or NAND0 Ready/Busy#
		PINCTRL	I/O	3	GPIO0[18]
16	GPMI_A0	GPMI	O	0	ATA_A0 or NAND CLE
		EMI	O	1	EMI_A23
		PINCTRL	I/O	3	GPIO0[22]
17	GPMI_A1	GPMI	O	0	ATA_A1 or NAND ALE
		EMI	O	1	EMI_A24
		PINCTRL	I/O	3	GPIO0[23]
18	GPMI_RDN	GPMI	O	0	ATA DIOR-:HSTROBE or NAND Read Strobe
		PINCTRL	I/O	3	GPIO0[17]
19	VDDIO2	POWER	P		Digital I/O Power 2
20	VSSD3	POWER	P		Digital Ground 3
21	GPMI_WRN	GPMI	O	0	ATA DIOW-:STOP or NAND Write Strobe
		PINCTRL	I/O	3	GPIO0[21]

Table 1008. Pin Definitions—100-Pin TQFP Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
22	EMI_CE0N	EMI	O	0	EMI CE0n
		GPMI	O	1	GPMI_CE0n
		PINCTRL	I/O	3	GPIO3[0]
23	EMI_CE1N	EMI	O	0	EMI CE1n
		GPMI	O	1	GPMI_CE1n
		PINCTRL	I/O	3	GPIO3[1]
24	ROTARYA	TIMER	I/O	0	Rotary Encoder A
		PINCTRL	I/O	3	GPIO3[15]
25	ROTARYB	TIMER	I/O	0	Rotary Encoder B
		PINCTRL	I/O	3	GPIO3[16]
26	I2C_SCL	I2C	I/O	0	I^2C Serial Clock (o.d.)
		PINCTRL	I/O	3	GPIO3[17]
27	I2C_SDA	I2C	I/O	0	I^2C Serial Data (o.d.)
		PINCTRL	I/O	3	GPIO3[18]
28	PWM0	PWM	I/O	0	PWM
		ETM	O	1	ETM_TSYNCA
		UARTDBG	I	2	UART1 RX (Debug)
		PINCTRL	I/O	3	GPIO3[10]
29	GPMI_RESETN	GPMI	O	0	ATA Reset or NAND Write Protect or Renesas Reset
		ETM	O	1	EMI Reset
		PINCTRL	I/O	3	GPIO1[20]
30	LCD_BUSY	LCDIF	I	0	LCD Busy
		PINCTRL	I/O	3	GPIO1[21]
31	LCD_CS	LCDIF	O	0	LCD Interface Chip Select
		ETM	O	1	ETM_TCLK
		PINCTRL	I/O	3	GPIO1[19]
32	LCD_D00	LCDIF	O	0	LCD Interface Data 0
		ETM	O	1	ETM_DA0
		PINCTRL	I/O	3	GPIO1[0]
33	LCD_D01	LCDIF	O	0	LCD Interface Data 1
		ETM	O	1	ETM_DA1
		PINCTRL	I/O	3	GPIO1[1]
34	LCD_D02	LCDIF	O	0	LCD Interface Data 2
		ETM	O	1	ETM_DA2
		PINCTRL	I/O	3	GPIO1[2]
35	LCD_D03	LCDIF	O	0	LCD Interface Data 3
		ETM	O	1	ETM_DA3
		PINCTRL	I/O	3	GPIO1[3]
36	VSSD4	POWER	P		Digital Ground 4
37	VDDD2	POWER	P		Digital Core Power 2
38	LCD_D04	LCDIF	O		LCD Interface Data 4

Table 1008. Pin Definitions—100-Pin TQFP Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
39	LCD_D05	LCDIF	O		LCD Interface Data 5
		ETM	O	1	ETM_DA5
		PINCTRL	I/O	3	GPIO1[5]
40	VSSD5	POWER	P		Digital Ground 5
41	VDDIO3	POWER	P		Digital I/O Power 3
42	LCD_D06	LCDIF	O	0	LCD Interface Data 6
		ETM	O	1	ETM_DA6
		PINCTRL	I/O	3	GPIO1[6]
43	LCD_D07	LCDIF	O	0	LCD Interface Data 7
		ETM	O	1	ETM_DA7
		PINCTRL	I/O	3	GPIO1[7]
44	LCD_RS	LCDIF	O	0	LCD Interface Register Select
		ETM	O	1	ETM_PSA0
		PINCTRL	I/O	3	GPIO1[17]
45	LCD_RESET	LCDIF	O	0	LCD Interface Reset Out
		ETM	O	1	ETM_PSA1
		PINCTRL	I/O	3	GPIO1[16]
46	LCD_WR	LCDIF	O	0	LCD Interface Data Write
		ETM	O	1	ETM_PSA2
		PINCTRL	I/O	3	GPIO1[18]
47	DCDC1_VDDIO	DCDC	P		DCDC#1 VDDIO Output Mode 3, 1 Battery Connection Mode 2, 0
48	DCDC1_VDDD	DCDC	P		DCDC#1 VDDD Output Mode 3, 1
49	DCDC1_BATT	DCDC	P		DCDC#1 Inductor
50	DCDC1_GND	DCDC	P		DCDC#1 Ground
51	JTAG_RESET	SYSTEM	I		Debug Reset
52	DCDC_MODE	DCDC	A		DCDC Mode Select
53	DCDC2_GND	DCDC	P		DCDC#2 Ground
54	DCDC2_L1	DCDC	P		DCDC#2 Inductor Mode 2, 1, 0 Peripheral VDDIO mode 3
55	DCDC2_PFET	DCDC	P		DCDC#2 Battery Connection Mode 0, 1 VDDIO Connection Mode 2, 3
56	VDD5V	POWER	P		5-V Power Input
57	BATT	POWER	P		Battery Input / LRADC7–0
58	LRADC1	LRADC	A		LRADC1 (Button 2, Temp, or MicBias)
59	LRADC0	LRADC	A		LRADC0 (Button 1 or Temp)
60	HP_SENSE	HP	A		Direct Coupled Headphone Sense
61	HPR	HP	A		Headphone Right
62	VDDA1	POWER			Analog Power1
63	HP_VGND	HP	A		Direct Coupled Headphone Virtual Ground
64	VSSA1	POWER	P		Analog Ground 1
65	HPL	HP	A		Headphone Left

Table 1008. Pin Definitions—100-Pin TQFP Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
66	LINE1R	ADC	A		Line-In 1 Right
	DRI_CLK	DRI	I	N/A	Used for digital radio clock input if enabled by HW_DRI_CTRL_ENABLE_INPUTS
67	LINE1L	ADC	A		Line-In 1 Left
	DRI_DATA	DRI	I	N/A	Used for Digital Radio Data Input if Enabled by HW_DRI_CTRL_ENABLE_INPUTS
68	MIC	ADC	A		Microphone Input
69	VAG	HP	A		Analog Decoupling Capacitor
70	REFP	ADC	A		ADC Positive Reference Capacitor
71	VSSA3	POWER	P		Analog Ground 3 - DB
72	REF_RES	USB	A		USB Reference Resistor
73	PSWITCH	DCDC	P		Power-On/Recovery/Software-Visible
74	XTALO	CLOCK	A		Crystal out—24 MHz
75	XTALI	CLOCK	A		Crystal in—24 MHz
76	VDDXTAL	CLOCK	A		Crystal Power Filter Cap
77	TESTMODE	SYSTEM	I		Test Mode Pin
78	USB_OTG_ID	USB	A		USB OTG ID Sense
79	JTAG_TMS	SYSTEM	I		Debug Test Mode Select
80	JTAG_TDI	SYSTEM	I		Debug Data In
81	JTAG_TCK	SYSTEM	I		Debug Clock
82	USB_DP	USB	A		USB Positive Data Line
83	USB_DM	USB	A		USB Negative Data Line
84	JTAG_TDO	SYSTEM	O		Debug Data Out
85	VDDIO4	POWER	P		Digital I/O Power 4
86	VSSD6	POWER	P		Digital Ground 6
87	PWM4	PWM	I/O	0	PWM - 16ma drive for OTG Vbus
		ETM	O	1	ETM_PSB1
		PINCTRL	I/O	3	GPIO3[14]
88	PWM2	PWM	I/O	0	PWM
		ETM	O	1	ETM_PSB2
		JTAG		2	RTCK - JTAG Return Clock
		PINCTRL	I/O	3	GPIO3[12]
89	PWM3	PWM	I/O	0	PWM - 16ma Drive for SPDIF Out
		ETM	O	1	ETM_PSB0
		SPDIF	O	2	SPDIF Out
		PINCTRL	I/O	3	GPIO3[13]
90	PWM1	PWM	I/O	0	PWM
		ETM	O	1	ETM_TSYNCB
		UARTDBG	O	2	UART1 TX (Debug)
		PINCTRL	I/O	3	GPIO3[11]
91	VDDD3	POWER	P		Digital Core Power 3
92	VSSD7	POWER	P		Digital Ground 7

Table 1008. Pin Definitions—100-Pin TQFP Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
93	EMI_CLK	EMI	O	0	EMI Clock
		PINCTRL	I/O	3	GPIO3[4]
94	SSP_DETECT	SSP	I/O	0	Removable Card Detect
		JTAG		2	RTCK - JTAG Return Clock
		PINCTRL	I/O	3	GPIO0[25]
95	SSP_DATA0	SSP	I/O	0	SPI MISO or SD/MMC DAT0
		PINCTRL	I/O	3	GPIO0[28]
96	SSP_CMD	SSP	I/O	0	SPI MOSI or MS SDIO or SD/MMC CMD
		PINCTRL	I/O	3	GPIO0[26]
97	SSP_DATA2	SSP	I/O	0	SD/MMC Data 2
		PINCTRL	I/O	3	GPIO0[30]
98	SSP_DATA3	SSP	I/O	0	SPI Slave Select 0 or MS BS or SD/MMC DAT3
		PINCTRL	I/O	3	GPIO0[31]
99	SSP_DATA1	SSP	I/O	0	SD/MMC Data 1
100	SSP_SCK	SSP	I/O	0	SPI Serial Clock
		PINCTRL	I/O	3	GPIO0[27]

35.1.2. Pin Definitions for 169-Pin BGA Package

Table 1009. Pin Definitions—169-Pin BGA Package

Pin Number	Pin Name	Module	Type	Pin Mux	Description
A1	TESTMODE	SYSTEM	I		Test Mode Pin
A2	XTALI	CLOCK	A		Crystal In - 24 MHz
A3	XTALO	CLOCK	A		Crystal Out - 24 MHz
A4	RTC_XTALI	RTC	A		32.768 kHz Xtal In
A5	RTC_XTALO	RTC	A		32.768 kHz Xtal Out
A6	LINE1L	ADC	A		Line-In 1 Left
	DRI_DATA	DRI	I	N/A	Used for Digital Radio Data Input if Enabled by HW_DRI_CTRL_ENABLE_INPUTS
A7	HP_VGND	HP	A		Direct Coupled Headphone Virtual Ground
A8	LRADC0	LRADC	A		LRADC0 (Button 1, Temp, or MicBias)
A9	BATT	POWER	P		Battery Input / LRADC7–0
A10	DCDC2_PFET	DCDC	P		DCDC#2 Battery Connection Mode 0, 1 VDDIO Connection Mode 2, 3
A11	DCDC2_L1	DCDC	P		DCDC#2 Inductor Mode 2, 1, 0 Peripheral VDDIO Mode 3
A12	DCDC2_GND	DCDC	P		DCDC#2 Ground
A13	DCDC1_BATT	DCDC	P		DCDC#1 Inductor
B1	JTAG_TMS	SYSTEM	I		Debug Test Mode Select

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
B2	REFP	ADC	A		ADC Positive Reference Capacitor
B3	VDDXTAL	CLOCK	A		Crystal Power Filter Cap - Cross Bond to Side 4
B4	REF_RES	USB	A		USB Reference Resistor
B5	LINE1R	ADC	A		Line-In 1 Right
	DRI_CLK	DRI	I	N/A	Used for digital radio clock input if enabled by HW_DRI_CTRL_ENABLE_INPUTS
B6	MIC	ADC	A		Microphone Input
B7	HPR	HP	A		Headphone Right
B8	HP_SENSE	HP	A		Direct Coupled Headphone Sense
B9	LRADC2	LRADC	A		LRADC2 (Touchscreen 0 or Line2 Left)
B10	DCDC2_L2	DCDC	P		DCDC#2 Inductor Mode 0 Peripheral VDDIO Mode 2
B11	DCDC2_VDDIO	DCDC	P		DCDC#2 VDDIO Output Mode 0 VDDIO Connection Mode 2
B12	DCDC1_GND	DCDC	P		DCDC#1 Ground
B13	DCDC1_VDDD	DCDC	P		DCDC#1 VDDD Output Mode 3, 1
C1	USB_DM	USB	A		USB Negative Data Line
C2	USB_DP	USB	A		USB Positive Data Line
C3	JTAG_TDI	SYSTEM	I		Debug Data In
C4	PSWITCH	DCDC	P		Power-On / Recovery / Software-Visible
C5	VAG	HP	A		Analog Decoupling Capacitor
C6	HPL	HP	A		Headphone Left
C7	LRADC1	LRADC	A		LRADC1 (Button 2, Temp, or MicBias)
C8	LRADC5	LRADC	A		LRADC5 (Touchscreen 3)
C9	JTAG_RESET	SYSTEM	I		Debug Reset
C10	DCDC_MODE	DCDC	A		DCDC Mode Select
C11	LCD_WR	LCDIF	O	0	LCD Interface Data Write
		ETM	O	1	ETM_PSA2
		PINCTRL	I/O	3	GPIO1[18]
C12	LCD_RS	LCDIF	O	0	LCD Interface Register Select
		ETM	O	1	ETM_PSA0
		PINCTRL	I/O	3	GPIO1[17]
C13	DCDC1_VDDIO	DCDC	P	0	USB#1 VDDIO Output Mode 3, 1 Battery Connection Mode 2, 0
D1	EMI_D03	EMI	I/O	0	EMI Data 3
		PINCTRL	I/O	3	GPIO2[3]
D2	JTAG_TCK	SYSTEM	I	0	Debug Clock
D3	EMI_D00	EMI	I/O	0	EMI Data 0
		PINCTRL	I/O	3	GPIO2[0]
D4	JTAG_TDO	SYSTEM	O		Debug Data Out
D5	VSSA3	POWER	P		Analog Ground 3 - DB
D6	VDDA1	POWER	P		Analog Power1
D7	SPEAKERM	SPEAKER	A		Speaker Output
D8	VSSA1	POWER	P	0	Analog Ground 1

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
D9	VDD5V	POWER	P		5-V Power Input
D10	LRADC4	LRADC	A		LRADC4 (Touchscreen 2)
D11	LCD_RESET	LCDIF	O	0	LCD Interface Reset Out
		ETM	O	1	ETM_PSA1
		PINCTRL	I/O	3	GPIO1[16]
D12	LCD_D07	LCDIF	O	0	LCD Interface Data 7
		ETM	O	1	ETM_DA7
		PINCTRL	I/O	3	GPIO1[7]
D13	LCD_D06	LCDIF	O	0	LCD Interface Data 6
		ETM	O	1	ETM_DA6
		PINCTRL	I/O	3	GPIO1[6]
E1	PWM2	PWM	I/O	0	PWM
		ETM	O	1	ETM_PSB2
		SYSTEM	O	2	RTCK - JTAG Return Clock
		PINCTRL	I/O	3	GPIO3[12]
E2	EMI_D02	EMI	I/O	0	EMI Data 2
		PINCTRL	I/O	3	GPIO2[2]
E3	EMI_D01	EMI	I/O	0	EMI Data 1
		PINCTRL	I/O	3	GPIO2[1]
E4	VSSD6	POWER	P	0	Digital Ground 6
E5	VDDIO4	POWER	P	0	Digital I/O Power 4
E6	EMI_D05	EMI	I/O	0	EMI Data 5
		PINCTRL	I/O	3	GPIO2[5]
E7	SPEAKERP	SPEAKER	A	0	Speaker Output +
E8	USB_OTG_ID	USB	A	0	USB OTG ID Sense
E9	LRADC3	LRADC	A	0	LRADC3 (Touchscreen 1 or Line2 Right)
E10	LCD_D08	LCDIF	O	0	LCD Interface Data 8
		ETM	O	1	ETM_DB0
		PINCTRL	I/O	3	GPIO1[8]
E11	LCD_D09	LCDIF	O	0	LCD Interface Data 9
		ETM	O	1	ETM_DB1
		PINCTRL	I/O	3	GPIO1[9]
E12	LCD_D11	LCDIF	O	0	LCD Interface Data 11
		ETM	O	1	ETM_DB3
		PINCTRL	I/O	3	GPIO1[11]
E13	LCD_D10	LCDIF	O	0	LCD Interface Data 10
		ETM	O	1	ETM_DB2
		PINCTRL	I/O	3	GPIO1[10]
F1	PWM3	PWM	I/O	0	PWM - 16ma Drive for SPDIF Out
		ETM	O	1	ETM_PSB0
		SPDIF	O	2	SPDIF Out
		PINCTRL	I/O	3	GPIO3[13]

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
F2	EMI_D04	EMI	I/O	0	EMI Data 4
		PINCTRL	I/O	3	GPIO2[4]
F3	PWM4	PWM	I/O	0	PWM - 16ma Drive for OTG Vbus
		ETM	O	1	ETM_PSB1
		PINCTRL	I/O	3	GPIO3[14]
F4	EMI_D07	EMI	I/O	0	EMI Data 7
		PINCTRL	I/O	3	GPIO2[7]
F5	EMI_D09	EMI	I/O	0	EMI Data 9
		PINCTRL	I/O	3	GPIO2[9]
F6	EMI_D06	EMI	I/O	0	EMI Data 6
		PINCTRL	I/O	3	GPIO2[6]
F7	LCD_D14	LCDIF	O	0	LCD Interface Data 14
		ETM	O	1	ETM_DB6
		PINCTRL	I/O	3	GPIO1[14]
F8	LCD_D13	LCDIF	O	0	LCD Interface Data 13
		ETM	O	1	ETM_DB5
		PINCTRL	I/O	3	GPIO1[13]
F9	LCD_D12	LCDIF	O	0	LCD Interface Data 12
		ETM	O	1	ETM_DB4
		PINCTRL	I/O	3	GPIO1[12]
F10	VDDD2	POWER	P		Digital Core Power 2
F11	LCD_D02	LCDIF	O	0	LCD Interface Data 2
		ETM	O	1	ETM_DA2
		PINCTRL	I/O	3	GPIO1[2]
F12	LCD_D05	LCDIF	O	0	LCD Interface Data 5
		ETM	O	1	ETM_DA5
		PINCTRL	I/O	3	GPIO1[5]
F13	LCD_D04	LCDIF	O	0	LCD Interface Data 4
		ETM	O	1	ETM_DA4
		PINCTRL	I/O	3	GPIO1[4]
G1	EMI_CLK	EMI	O	0	EMI clock
		PINCTRL	I/O	3	GPIO3[4]
G2	EMI_D11	EMI	I/O	0	EMI data 11
		PINCTRL	I/O	3	GPIO2[11]
G3	EMI_D08	EMI	I/O	0	EMI Data 8
		PINCTRL	I/O	3	GPIO2[8]
G4	VDDD3	POWER	P	0	Digital Core Power 3
G5	EMI_D10	EMI	I/O	0	EMI Data 10
		PINCTRL	I/O	3	GPIO2[10]
G6	VSSD7	POWER	P		Digital Ground 7

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
G7	PWM1	PWM	I/O	0	PWM
		ETM	O	1	ETM_TSYNCB
		UARTDBG	O	2	UART1 TX (Debug)
		PINCTRL	I/O	3	GPIO3[11]
G8	GPMI_D09	GPMI	I/O	0	ATA/NAND Data 9
		ETM	O	1	EMI_A16
		PINCTRL	I/O	3	GPIO0[9]
G9	VSSD4	POWER	P		Digital Ground 4
G10	LCD_BUSY	LCDIF	I	0	LCD Busy
		PINCTRL	I/O	3	GPIO1[21]
G11	LCD_D15	LCDIF	O	0	LCD Interface Data 15
		ETM	O	1	ETM_DB7
		SYSTEM	O	2	RTCK - JTAG Return Clock
		PINCTRL	I/O	3	GPIO1[15]
G12	LCD_D03	LCDIF	O	0	LCD Interface Data 3
		ETM	O	1	ETM_DA3
		PINCTRL	I/O	3	GPIO1[3]
G13	LCD_D01	LCDIF	O	0	LCD Interface Data 1
		ETM	O	1	ETM_DA1
		PINCTRL	I/O	3	GPIO1[1]
H1	EMI_D14	EMI	I/O	0	Emi Data 14
		PINCTRL	I/O	3	GPIO2[14]
H2	SSP_DETECT	SSP	I/O	0	Removable Card Detect
		SYSTEM	O	2	RTCK - JTAG Return Clock
		PINCTRL	I/O	3	GPIO0[25]
H3	EMI_D13	EMI	I/O	0	EMI Data 13
		PINCTRL	I/O	3	GPIO2[13]
H4	EMI_D15	EMI	I/O	0	EMI Data 15
		PINCTRL	I/O	3	GPIO2[15]
H5	EMI_D12	EMI	I/O	0	EMI Data 12
		PINCTRL	I/O	3	GPIO2[12]
H6	VSSD1	POWER	P		Digital Ground 1
H7	GPMI_D02	GPMI	I/O	0	ATA/NAND Data 2
		PINCTRL	I/O	3	GPIO0[2]
H8	GPMI_D07	GPMI	I/O	0	ATA/NAND Data 7
		PINCTRL	I/O	3	GPIO0[7]
H9	VDDIO2	POWER	P	0	Digital I/O Power 2
H10	UART2_CTS	UART	I	0	High-Speed UART CTS Flow Control
		PINCTRL	I/O	3	GPIO1[22]
H11	LCD_CS	LCDIF	O	0	LCD Interface Chip Select.
		ETM	O	1	ETM_TCLK
		PINCTRL	I/O	3	GPIO1[19]

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
H12	UART2_TX	UART	I/O	0	High-Speed UART TX
		IR	O	2	IR_TX
		PINCTRL	I/O	3	GPIO1[25]
H13	LCD_D00	LCDIF	O	0	LCD Interface Data 0
		ETM	O	1	ETM_DA0
		PINCTRL	I/O	3	GPIO1[0]
J1	EMI_A00	EMI	O	0	EMI Address 0
		PINCTRL	I/O	3	GPIO2[16]
J2	EMI_A01	EMI	O	0	EMI Address 1
		PINCTRL	I/O	3	GPIO2[17]
J3	SSP_CMD	SSP	I/O	0	SPI MOSI or MS, SDIO, or SD/MMC CMD
		PINCTRL	I/O	3	GPIO0[26]
J4	EMI_A02	EMI	O	0	EMI Address 2
		PINCTRL	I/O	3	GPIO2[18]
J5	SSP_DATA0	SSP	I/O	0	SPI MISO or SD/MMC DAT0
		PINCTRL	I/O	3	GPIO0[28]
J6	GPMI_D01	GPMI	I/O	0	ATA/NAND Data 1
		PINCTRL	I/O	3	GPIO0[1]
J7	VSSD2	POWER	P		Digital Ground 2
J8	GPMI_A0	GPMI	O	0	ATA_A0 or NAND Cle
		EMI	O	1	EMI_A23
		PINCTRL	I/O	3	GPIO0[22]
J9	VSSD3	POWER	P		Digital Ground 3
J10	EMI_CE2N	EMI	I/O	0	EMI CE2n
		GPMI	O	1	GPMI_CE2n
		PINCTRL	I/O	3	GPIO3[2]
J11	PWM0	PWM	I/O	0	PWM
		ETM	O	1	ETM_TSYNCA
		UARTDBG	I	2	UART1 RX (Debug)
		PINCTRL	I/O	3	GPIO3[10]
J12	GPMI_RESETN	GPMI	O	0	ATA Reset or NAND Write Protect or Renesas Reset
		ETM	O	1	EMI Reset
		PINCTRL	I/O	3	GPIO1[20]
J13	UART2_RX	UART	I	0	High-Speed UART RX
		IR	I	2	IR_RX
		PINCTRL	I/O	3	GPIO1[24]
K1	EMI_A03	EMI	O	0	EMI Address 3
		PINCTRL	I/O	3	GPIO2[19]
K2	SSP_DATA3	SSP	I/O	0	SPI Slave Select 0 or MS BS or SD/MMC DAT3
		PINCTRL	I/O	3	GPIO0[31]
K3	EMI_A04	EMI	O	0	EMI Address 4
		PINCTRL	I/O	3	GPIO2[20]

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
K4	SSP_DATA2	SSP	I/O	0	SD/MMC Data 2
		PINCTRL	I/O	3	GPIO0[30]
K5	VDDIO1	POWER	P	0	Digital I/O Power 1 / LRADC7–2
K6	GPMI_D12	GPMI	I/O	0	ATA/NAND Data 12
		EMI	O	1	EMI_A19
		GPMI	O	2	GPMI_CE0n
		PINCTRL	I/O	3	GPIO0[12]
K7	VDDD1	POWER	P	0	Digital Core Power 1 / LRADC7–1
K8	EMI_A13	EMI	O	0	EMI Address 13 / SDRAM BA0
		PINCTRL	I/O	3	GPIO2[29]
K9	GPMI_A1	GPMI	O	0	ATA_A1 or NAND ALE
		EMI	O	1	EMI_A24
		PINCTRL	I/O	3	GPIO0[23]
K10	EMI_A10	EMI	O	0	EMI Address 10
		PINCTRL	I/O	3	GPIO2[26]
K11	I2C_SCL	I2C	I/O	0	I^2C Serial Clock (o.d.)
		PINCTRL	I/O	3	GPIO3[17]
K12	I2C_SDA	I2C	I/O	0	I^2C Serial Data (o.d.)
		PINCTRL	I/O	3	GPIO3[18]
K13	UART2_RTS	UART	O	0	High-Speed UART RTS Flow Control
		SYSTEM	O	1	RTCK - JTAG Return Clock
		IR	O	2	IR_CLK
		PINCTRL	I/O	3	GPIO1[23]
L1	EMI_A05	EMI	O	0	EMI Address 5
		PINCTRL	I/O	3	GPIO2[21]
L2	EMI_A06	EMI	O	0	EMI Address 6
		PINCTRL	I/O	3	GPIO2[22]
L3	EMI_A08	EMI	O	0	EMI Address 8
		PINCTRL	I/O	3	GPIO2[24]
L4	EMI_A09	EMI	O	0	EMI Address 9
		PINCTRL	I/O	3	GPIO2[25]
L5	GPMI_D14	GPMI	I/O	0	ATA/NAND Data 14
		EMI	O	1	EMI_A21
		GPMI	O	2	GPMI_CE2n
		PINCTRL	I/O	3	GPIO0[14]
L6	GPMI_D11	GPMI	I/O	0	ATA/NAND Data 11
		EMI	O	1	EMI_A18
		PINCTRL	I/O	3	GPIO0[11]
L7	EMI_A14	EMI	O	0	EMI Address 14 / SDRAM BA1
		PINCTRL	I/O	3	GPIO2[30]
L8	GPMI_RDY	GPMI	I	0	ATA IORDY:DSTROBE or NAND0 Ready/Busy#
		PINCTRL	I/O	3	GPIO0[18]

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
L9	GPMI_A2	GPMI	O	0	ATA_A2
		EMI	O	1	EMI_A25
		PINCTRL	I/O	3	GPIO0[24]
L10	EMI_RASN	EMI	O	0	EMI RASN
		PINCTRL	I/O	3	GPIO2[31]
L11	EMI_CE1N	EMI	O	0	EMI CE1n
		GPMI	O	1	GPMI_CE1n
		PINCTRL	I/O	3	GPIO3[1]
L12	ROTARYB	TIMER	I/O	0	Rotary Encoder B
		PINCTRL	I/O	3	GPIO3[16]
L13	GPMI_RDY3	GPMI	I/O	0	ATA DMARQ or NAND3 R/B#
		EMI	O	1	EMI_OEN
		PINCTRL	I/O	3	GPIO0[19]
M1	SSP_DATA1	SSP	I/O	0	SD/MMC Data 1
		PINCTRL	I/O	3	GPIO0[29]
M2	EMI_A07	EMI	O	0	EMI Address 7
		PINCTRL	I/O	3	GPIO2[23]
M3	EMI_WEN	EMI	O	0	EMI WEN
		PINCTRL	I/O	3	GPIO3[9]
M4	GPMI_D15	GPMI	I/O	0	ATA/NAND Data 15
		EMI	O	1	EMI_A22
		GPMI	O	2	GPMI_CE3n
		PINCTRL	I/O	3	GPIO0[15]
M5	GPMI_D13	GPMI	I/O	0	ATA/NAND Data 13
		EMI	O	1	EMI_A20
		GPMI	O	2	GPMI_CE1n
		PINCTRL	I/O	3	GPIO0[13]
M6	GPMI_D04	GPMI	I/O	0	ATA/NAND Data 4
		PINCTRL	I/O	3	GPIO0[4]
M7	GPMI_D06	GPMI	I/O	0	ATA/NAND Data 6
		PINCTRL	I/O	3	GPIO0[6]
M8	GPMI_IRQ	GPMI	I	0	ATA INTRQ or NAND1 Ready/Busy#
		PINCTRL	I/O	3	GPIO0[16]
M9	EMI_A11	EMI	O	0	EMI Address 11
		PINCTRL	I/O	3	GPIO2[27]
M10	GPMI_WRN	GPMI	O	0	ATA DIOW-:STOP or NAND Write Strobe
		PINCTRL	I/O	3	GPIO0[21]
M11	EMI_CE0N	EMI	O	0	EMI CE0n
		GPMI	O	1	GPMI_CE0n
		PINCTRL	I/O	3	GPIO3[0]
M12	EMI_DQM0	EMI	O	0	EMI DQM0
		PINCTRL	I/O	3	GPIO3[7]

Table 1009. Pin Definitions—169-Pin BGA Package (Continued)

Pin Number	Pin Name	Module	Type	Pin Mux	Description
M13	EMI_DQM1	EMI	O	0	EMI DQM1
		PINCTRL	I/O	3	GPIO3[8]
N1	EMI_CKE	EMI	O	0	EMI Clock Enable
		PINCTRL	I/O	3	GPIO3[5]
N2	SSP_SCK	SSP	I/O	0	SPI Serial Clock - (Bond to Pin 100 in 100 TQFP)
		PINCTRL	I/O	3	GPIO0[27]
N3	GPMI_RDY2	GPMI	I/O	0	ATA DMACK or NAND2 Ready/Busy#
		PINCTRL	I/O	3	GPIO0[20]
N4	GPMI_D00	GPMI	I/O	0	ATA/NAND Data 0
		PINCTRL	I/O	3	GPIO0[0]
N5	GPMI_D03	GPMI	I/O	0	ATA/NAND Data 3
		PINCTRL	I/O	3	GPIO0[3]
N6	GPMI_D05	GPMI	I/O	0	ATA/NAND Data 5
		PINCTRL	I/O	3	GPIO0[5]
N7	GPMI_D10	GPMI	I/O	0	ATA/NAND Data 10
		EMI	O	1	EMI_A17
		PINCTRL	I/O	3	GPIO0[10]
N8	GPMI_D08	GPMI	I/O	0	ATA/NAND Data 8
		EMI	O	1	EMI_A15
		PINCTRL	I/O	3	GPIO0[8]
N9	EMI_A12	EMI	O	0	EMI Address 12
		PINCTRL	I/O	3	GPIO2[28]
N10	GPMI_RDN	GPMI	O	0	ATA DIOR-:HSTROBE or NAND Read Strobe
		PINCTRL	I/O	3	GPIO0[17]
N11	EMI_CE3N	EMI	I/O	0	EMI CE3n
		GPMI	O	1	GPMI_CE3n
		PINCTRL	I/O	3	GPIO3[3]
N12	EMI_CASN	EMI	O	0	EMI casn
		PINCTRL	I/O	3	GPIO3[6]
N13	ROTARYA	TIMER	I/O	0	Rotary Encoder A
		PINCTRL	I/O	3	GPIO3[15]

Table 1010. BGA Package Pin Map

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	TESTMODE	XTALI	XTALO	RTC_XTALI	RTC_XTALO	LINE1L	HP_VGND	LRADC0	BATT	DCDC2_PFET	DCDC2_L1	DCDC2_GND	DCDC1_BATT
B	JTAG_TMS	REFP	VDDXTAL	REF_RES	LINE1R	MIC	HPR	HP_SENSE	LRADC2	DCDC2_L2	DCDC2_VDDIO	DCDC1_GND	DCDC1_VDDD
C	USB_DM	USB_DP	JTAG_TDI	PSWITCH	VAG	HPL	LRADC1	LRADC5	JTAG_RESET	DCDC_MODE	LCD_WR	LCD_RS	DCDC1_VDDIO
D	EMI_D03	JTAG_TCK	EMI_D00	JTAG_TDO	VSSA3	VDDA1	SPEAKERM	VSSA1	VDD5V	LRADC4	LCD_RESET	LCD_D07	LCD_D06
E	PWM2	EMI_D02	EMI_D01	VSSD6	VDDIO4	EMI_D05	SPEAKERP	USB_OTG_ID	LRADC3	LCD_D08	LCD_D09	LCD_D11	LCD_D10
F	PWM3	EMI_D04	PWM4	EMI_D07	EMI_D09	EMI_D06	LCD_D14	LCD_D13	LCD_D12	VDDD2	LCD_D02	LCD_D05	LCD_D04
G	EMI_CLK	EMI_D11	EMI_D08	VDDD3	EMI_D10	VSSD7	PWM1	GPMI_D09	VSSD4	LCD_BUSY	LCD_D15	LCD_D03	LCD_D01
H	EMI_D14	SSP_DETECT	EMI_D13	EMI_D15	EMI_D12	VSSD1	GPMI_D02	GPMI_D07	VDDIO2	UART2_CTS	LCD_CS	UART2_TX	LCD_D00
J	EMI_A00	EMI_A01	SSP_CMD	EMI_A02	SSP_DATA0	GPMI_D01	VSSD2	GPMI_A0	VSSD3	EMI_CE2N	PWM0	GPMI_RESETN	UART2_RX
K	EMI_A03	SSP_DATA3	EMI_A04	SSP_DATA2	VDDIO1	GPMI_D12	VDDD1	EMI_A13	GPMI_A1	EMI_A10	I2C_SCL	I2C_SDA	UART2_RTS
L	EMI_A05	EMI_A06	EMI_A08	EMI_A09	GPMI_D14	GPMI_D11	EMI_A14	GPMI_RDY	GPMI_A2	EMI_RASN	EMI_CE1N	ROTARYB	GPMI_RDY3
M	SSP_DATA1	EMI_A07	EMI_WEN	GPMI_D15	GPMI_D13	GPMI_D04	GPMI_D06	GPMI_IRQ	EMI_A11	GPMI_WRN	EMI_CE0N	EMI_DQM0	EMI_DQM1
N	EMI_CKE	SSP_SCK	GPMI_RDY2	GPMI_D00	GPMI_D03	GPMI_D05	GPMI_D10	GPMI_D08	EMI_A12	GPMI_RDN	EMI_CE3N	EMI_CASN	ROTARYA

35.2. Functional Pin Groups

This section includes all pins, listed in tables by function. To find the pin number for a given pin name, consult either Table 1008 for the TQFP pins or Table 1009 for the BGA pins.

35.2.1. Analog Pins

Table 1011. Analog Pins

Pin Name	Module	Type	Pin Mux	Description
HP_SENSE	HP	A		Direct Coupled Headphone Sense
HP_VGND	HP	A		Direct Coupled Headphone Virtual Ground
HPL	HP	A		Headphone Left
HPR	HP	A		Headphone Right
LINE1L	ADC	A		Line-In 1 Left
LINE1R	ADC	A		Line-In 1 Right
LRADC0	LRADC	A		LRADC0 (Button 1, Temp, or MicBias)

Table 1011. Analog Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
LRADC1	LRADC	A		LRADC1 (Button 2, Temp, or MicBias)
LRADC2	LRADC	A		LRADC2 (Touchscreen 0 or Line2 Left)
LRADC3	LRADC	A		LRADC3 (Touchscreen 1 Line2 Right)
LRADC4	LRADC	A		LRADC4 (Touchscreen 2)
LRADC5	LRADC	A		LRADC5 (Touchscreen 3)
MIC	ADC	A		Microphone Input
REF_RES	USB	A		USB Reference Resistor
REFP	ADC	A		ADC Positive Reference Capacitor
SPEAKERM	SPEAKER	A		Speaker Output –
SPEAKERP	SPEAKER	A		Speaker Output +
VAG	HP	A		Analog Decoupling Capacitor

35.2.2. DC-DC Converter Pins

Table 1012. DC-DC Converter Pins

Pin Name	Module	Type	Pin Mux	Description
BATT	POWER	P		Battery Input
DCDC_MODE	DCDC	A		DC-DC Mode Select
DCDC1_BATT	DCDC	P		DC-DC#1 Inductor
DCDC1_GND	DCDC	P		DC-DC#1 Ground
DCDC1_VDDD	DCDC	P		DC-DC#1 VDDD Output Mode 3, 1
DCDC1_VDDIO	DCDC	P		DC-DC#1 VDDIO Output Mode 3, 1 Battery Connection Mode 2, 0
DCDC2_GND	DCDC	P		DC-DC#22 Ground
DCDC2_L1	DCDC	P		DC-DC#2 Inductor Mode 2, 1, 0 Peripheral VDDIO Mode 3
DCDC2_L2	DCDC	P		DC-DC#2 Inductor Mode 0 Peripheral VDDIO Mode 2
DCDC2_PFET	DCDC	P		DC-DC#2 Battery Connection Mode 0, 1 VDDIO Connection Mode 2, 3
DCDC2_VDDIO	DCDC	P		DC-DC#2 VDDIO Output Mode 0 VDDIO Connection Mode 2
PSWITCH	DCDC	P		Power-On/Recovery/Software-Visible

35.2.3. General-Purpose Media Interface (GPMI) Pins

Table 1013. GPMI Pins

Pin Name	Module	Type	Pin Mux	Description
GPMI_A0	GPMI	O	0	ATA_A0 or NAND CLE
	EMI	O	1	EMI_A23
	PINCTRL	I/O	3	GPIO0[22]
GPMI_A1	GPMI	O	0	ATA_A1 or NAND ALE
	EMI	O	1	EMI_A24
	PINCTRL	I/O	3	GPIO0[23]
GPMI_A2	GPMI	O	0	ATA_A2
	EMI	O	1	EMI_A25
	PINCTRL	I/O	3	GPIO0[24]
EMI_CE0N	EMI	O	0	EMI CE0n
	GPMI	O	1	**GPMI_CE0n**
	PINCTRL	I/O	3	GPIO3[0]
EMI_CE1N	EMI	O	0	EMI CE1n
	GPMI	O	1	**GPMI_CE1n**
	PINCTRL	I/O	3	GPIO3[1]
EMI_CE2N	EMI	I/O	0	EMI CE2n
	GPMI	O	1	**GPMI_CE2n**
	PINCTRL	I/O	3	GPIO3[2]
EMI_CE3N	EMI	I/O	0	EMI CE3n
	GPMI	O	1	**GPMI_CE3n**
	PINCTRL	I/O	3	GPIO3[3]
GPMI_D00	GPMI	I/O	0	ATA/NAND Data 0
	PINCTRL	I/O	3	GPIO0[0]
GPMI_D01	GPMI	I/O	0	ATA/NAND Data 1
	PINCTRL	I/O	3	GPIO0[1]
GPMI_D02	GPMI	I/O	0	ATA/NAND Data 2
	PINCTRL	I/O	3	GPIO0[2]
GPMI_D03	GPMI	I/O	0	ATA/NAND Data 3
	PINCTRL	I/O	3	GPIO0[3]
GPMI_D04	GPMI	I/O	0	ATA/NAND Data 4
	PINCTRL	I/O	3	GPIO0[4]
GPMI_D05	GPMI	I/O	0	ATA/NAND Data 5
	PINCTRL	I/O	3	GPIO0[5]
GPMI_D06	GPMI	I/O	0	ATA/NAND Data 6
	PINCTRL	I/O	3	GPIO0[6]
GPMI_D07	GPMI	I/O	0	ATA/NAND Data 7
	PINCTRL	I/O	3	GPIO0[7]
GPMI_D08	GPMI	I/O	0	ATA/NAND Data 8
	EMI	O	1	EMI_A15
	PINCTRL	I/O	3	GPIO0[8]

Table 1013. GPMI Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
GPMI_D09	GPMI	I/O	0	ATA/NAND Data 9
	ETM	O	1	EMI_A16
	PINCTRL	I/O	3	GPIO0[9]
GPMI_D10	GPMI	I/O	0	ATA/NAND Data 10
	EMI	O	1	EMI_A17
	PINCTRL	I/O	3	GPIO0[10]
GPMI_D11	GPMI	I/O	0	ATA/NAND Data 11
	EMI	O	1	EMI_A18
	PINCTRL	I/O	3	GPIO0[11]
GPMI_D12	GPMI	I/O	0	ATA/NAND Data 12
	EMI	O	1	EMI_A19
	GPMI	O	2	**GPMI_CE0n**
	PINCTRL	I/O	3	GPIO0[12]
GPMI_D13	GPMI	I/O	0	ATA/NAND Data 13
	EMI	O	1	EMI_A20
	GPMI	O	2	**GPMI_CE1n**
	PINCTRL	I/O	3	GPIO0[13]
GPMI_D14	GPMI	I/O	0	ATA/NAND Data 14
	EMI	O	1	EMI_A21
	GPMI	O	2	**GPMI_CE2n**
	PINCTRL	I/O	3	GPIO0[14]
GPMI_D15	GPMI	I/O	0	ATA/NAND Data 15
	EMI	O	1	EMI_A22
	GPMI	O	2	**GPMI_CE3n**
	PINCTRL	I/O	3	GPIO0[15]
GPMI_IRQ	GPMI	I	0	ATA INTRQ or NAND1 Ready/Busy#
	PINCTRL	I/O	3	GPIO0[16]
GPMI_RDN	GPMI	O	0	ATA DIOR-:HSTROBE or NAND Read Strobe
	PINCTRL	I/O	3	GPIO0[17]
GPMI_RDY	GPMI	I	0	ATA IORDY:DSTROBE or NAND0 Ready/Busy#
	PINCTRL	I/O	3	GPIO0[18]
GPMI_RDY2	GPMI	I/O	0	ATA DMACK or NAND2 Ready/Busy#
	PINCTRL	I/O	3	GPIO0[20]
GPMI_RDY3	GPMI	I/O	0	ATA DMARQ or NAND3 R/B#
	EMI	O	1	EMI_OEN
	PINCTRL	I/O	3	GPIO0[19]
GPMI_RESETN	GPMI	O	0	ATA Reset, NAND Write Protect, or Renesas Reset
	ETM	O	1	EMI_RESET
	PINCTRL	I/O	3	GPIO1[20]
GPMI_WRN	GPMI	O	0	ATA DIOW-:STOP or NAND Write Strobe
	PINCTRL	I/O	3	GPIO0[21]

35.2.4. Synchronous Serial Port (SSP) Pins

Table 1014. Synchronous Serial Port Pins

Pin Name	Module	Type	Pin Mux	Description
SSP_CMD	SSP	I/O	0	SPI MOSI or MS, SDIO, or SD/MMC CMD
	PINCTRL	I/O	3	GPIO0[26]
SSP_DATA0	SSP	I/O	0	SPI MISO or SD/MMC DAT0
	PINCTRL	I/O	3	GPIO0[28]
SSP_DATA1	SSP	I/O	0	SD/MMC Data 1
	PINCTRL	I/O	3	GPIO0[29]
SSP_DATA2	SSP	I/O	0	SD/MMC Data 2
	PINCTRL	I/O	3	GPIO0[30]
SSP_DATA3	SSP	I/O	0	SPI Slave Select 0 or MS BS or SD/MMC DAT3
	PINCTRL	I/O	3	GPIO0[31]
SSP_DETECT	SSP	I/O	0	Removable Card Detect
	SYSTEM	O	2	RTCK - JTAG Return Clock
	PINCTRL	I/O	3	GPIO0[25]
SSP_SCK	SSP	I/O	0	SPI Serial Clock - (Bond to Pin 100 in 100 TQFP)
	PINCTRL	I/O	3	GPIO0[27]

35.2.5. Application and Debug UART Pins

Table 1015. UART Pins

Pin Name	Module	Type	Pin Mux	Description
UART2_CTS	UART	I	0	High-Speed UART CTS Flow Control
	PINCTRL	I/O	3	GPIO1[22]
UART2_RTS	UART	O	0	High-Speed UART RTS Flow Control
	SYSTEM	O	1	RTCK - JTAG Return Clock
	IR	O	2	IR_CLK
	PINCTRL	I/O	3	GPIO1[23]
UART2_RX	UART	I	0	High-Speed UART RX
	IR	I	2	IR_RX
	PINCTRL	I/O	3	GPIO1[24]
UART2_TX	UART	I/O	0	High-Speed UART TX
	IR	O	2	IR_TX
	PINCTRL	I/O	3	GPIO1[25]
PWM0	PWM	I/O	0	PWM
	ETM	O	1	ETM_TSYNCA
	UARTDBG	I	2	**UART1 RX (Debug)**
	PINCTRL	I/O	3	GPIO3[10]

Table 1015. UART Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
PWM1	PWM	I/O	0	PWM
	ETM	O	1	ETM_TSYNCB
	UARTDBG	O	2	**UART1 TX (Debug)**
	PINCTRL	I/O	3	GPIO3[11]

35.2.6. External Memory Interface (SDRAM/NOR) Pins

Table 1016. External Memory Interface (SDRAM/NOR) Pins

Pin Name	Module	Type	Pin Mux	Description
EMI_A00	EMI	O	0	EMI Address 0
	PINCTRL	I/O	3	GPIO2[16]
EMI_A01	EMI	O	0	EMI Address 1
	PINCTRL	I/O	3	GPIO2[17]
EMI_A02	EMI	O	0	EMI Address 2
	PINCTRL	I/O	3	GPIO2[18]
EMI_A03	EMI	O	0	EMI Address 3
	PINCTRL	I/O	3	GPIO2[19]
EMI_A04	EMI	O	0	EMI Address 4
	PINCTRL	I/O	3	GPIO2[20]
EMI_A05	EMI	O	0	EMI Address 5
	PINCTRL	I/O	3	GPIO2[21]
EMI_A06	EMI	O	0	EMI Address 6
	PINCTRL	I/O	3	GPIO2[22]
EMI_A07	EMI	O	0	EMI Address 7
	PINCTRL	I/O	3	GPIO2[23]
EMI_A08	EMI	O	0	EMI Address 8
	PINCTRL	I/O	3	GPIO2[24]
EMI_A09	EMI	O	0	EMI Address 9
	PINCTRL	I/O	3	GPIO2[25]
EMI_A10	EMI	O	0	EMI Address 10
	PINCTRL	I/O	3	GPIO2[26]
EMI_A11	EMI	O	0	EMI Address 11
	PINCTRL	I/O	3	GPIO2[27]
EMI_A12	EMI	O	0	EMI Address 12
	PINCTRL	I/O	3	GPIO2[28]
EMI_A13	EMI	O	0	EMI Address 13 / SDRAM BA0
	PINCTRL	I/O	3	GPIO2[29]
EMI_A14	EMI	O	0	EMI Address 14 / SDRAM BA1
	PINCTRL	I/O	3	GPIO2[30]
GPMI_D08	GPMI	I/O	0	ATA/NAND Data 8
	EMI	O	1	**EMI_A15**
	PINCTRL	I/O	3	GPIO0[8]

Table 1016. External Memory Interface (SDRAM/NOR) Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
GPMI_D09	GPMI	I/O	0	ATA/NAND Data 9
	ETM	O	1	**EMI_A16**
	PINCTRL	I/O	3	GPIO0[9]
GPMI_D10	GPMI	I/O	0	ATA/NAND Data 10
	EMI	O	1	**EMI_A17**
	PINCTRL	I/O	3	GPIO0[10]
GPMI_D11	GPMI	I/O	0	ATA/NAND Data 11
	EMI	O	1	**EMI_A18**
	PINCTRL	I/O	3	GPIO0[11]
GPMI_D12	GPMI	I/O	0	ATA/NAND Data 12
	EMI	O	1	**EMI_A19**
	PINCTRL	I/O	3	GPIO0[12]
GPMI_D13	GPMI	I/O	0	ATA/NAND Data 13
	EMI	O	1	**EMI_A20**
	PINCTRL	I/O	3	GPIO0[13]
GPMI_D14	GPMI	I/O	0	ATA/NAND Data 14
	EMI	O	1	**EMI_A21**
	PINCTRL	I/O	3	GPIO0[14]
GPMI_D15	GPMI	I/O	0	ATA/NAND Data 15
	EMI	O	1	**EMI_A22**
	PINCTRL	I/O	3	GPIO0[15]
GPMI_A0	GPMI	O	0	ATA_A0 or NAND CLE
	EMI	O	1	**EMI_A23**
	PINCTRL	I/O	3	GPIO0[22]
GPMI_A1	GPMI	O	0	ATA_A1 or NAND ALE
	EMI	O	1	**EMI_A24**
	PINCTRL	I/O	3	GPIO0[23]
GPMI_A2	GPMI	O	0	ATA_A2
	EMI	O	1	**EMI_A25**
	PINCTRL	I/O	3	GPIO0[24]
EMI_CASN	EMI	O	0	EMI CASn
	PINCTRL	I/O	3	GPIO3[6]
EMI_CE0N	EMI	O	0	EMI CE0n
	GPMI	O	1	GPMI_CE0n
	PINCTRL	I/O	3	GPIO3[0]
EMI_CE1N	EMI	O	0	EMI CE1n
	GPMI	O	1	GPMI_CE1n
	PINCTRL	I/O	3	GPIO3[1]
EMI_CE2N	EMI	I/O	0	EMI CE2n
	GPMI	O	1	GPMI_CE2n
	PINCTRL	I/O	3	GPIO3[2]
EMI_CE3N	EMI	I/O	0	EMI CE3n
	GPMI	O	1	GPMI_CE3n
	PINCTRL	I/O	3	GPIO3[3]

Table 1016. External Memory Interface (SDRAM/NOR) Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
EMI_CKE	EMI	O	0	EMI Clock Enable
	PINCTRL	I/O	3	GPIO3[5]
EMI_CLK	EMI	O	0	EMI Clock
	PINCTRL	I/O	3	GPIO3[4]
EMI_D00	EMI	I/O	0	EMI Data 0
	PINCTRL	I/O	3	GPIO2[0]
EMI_D01	EMI	I/O	0	EMI Data 1
	PINCTRL	I/O	3	GPIO2[1]
EMI_D02	EMI	I/O	0	EMI Data 2
	PINCTRL	I/O	3	GPIO2[2]
EMI_D03	EMI	I/O	0	EMI Data 3
	PINCTRL	I/O	3	GPIO2[3]
EMI_D04	EMI	I/O	0	EMI Data 4
	PINCTRL	I/O	3	GPIO2[4]
EMI_D05	EMI	I/O	0	EMI Data 5
	PINCTRL	I/O	3	GPIO2[5]
EMI_D06	EMI	I/O	0	EMI Data 6
	PINCTRL	I/O	3	GPIO2[6]
EMI_D07	EMI	I/O	0	EMI Data 7
	PINCTRL	I/O	3	GPIO2[7]
EMI_D08	EMI	I/O	0	EMI Data 8
	PINCTRL	I/O	3	GPIO2[8]
EMI_D09	EMI	I/O	0	EMI Data 9
	PINCTRL	I/O	3	GPIO2[9]
EMI_D10	EMI	I/O	0	EMI Data 10
	PINCTRL	I/O	3	GPIO2[10]
EMI_D11	EMI	I/O	0	EMI Data 11
	PINCTRL	I/O	3	GPIO2[11]
EMI_D12	EMI	I/O	0	EMI Data 12
	PINCTRL	I/O	3	GPIO2[12]
EMI_D13	EMI	I/O	0	EMI Data 13
	PINCTRL	I/O	3	GPIO2[13]
EMI_D14	EMI	I/O	0	EMI Data 14
	PINCTRL	I/O	3	GPIO2[14]
EMI_D15	EMI	I/O	0	EMI Data 15
	PINCTRL	I/O	3	GPIO2[15]
EMI_DQM0	EMI	O	0	EMI DQM0
	PINCTRL	I/O	3	GPIO3[7]
EMI_DQM1	EMI	O	0	EMI DQM1
	PINCTRL	I/O	3	GPIO3[8]
GPMI_RDY3	GPMI	I/O	0	ATA DMARQ or NAND3 R/B#
	EMI	O	1	**EMI_OEN**
	PINCTRL	I/O	3	GPIO0[19]

Table 1016. External Memory Interface (SDRAM/NOR) Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
EMI_RASN	EMI	O	0	EMI RASn
	PINCTRL	I/O	3	GPIO2[31]
EMI_WEN	EMI	O	0	EMI WEN
	PINCTRL	I/O	3	GPIO3[9]

35.2.7. I^2C Interface Pins

Table 1017. I^2C Interface Pins

Pin Name	Module	Type	Pin Mux	Description
I2C_SCL	I2C	I/O	0	I^2C Serial Clock (o.d.)
	PINCTRL	I/O	3	GPIO3[17]
I2C_SDA	I2C	I/O	0	I^2C Serial Data (o.d.)
	PINCTRL	I/O	3	GPIO3[18]

35.2.8. Digital Radio Interface (DRI) Pins

Table 1018. Digital Radio Interface Pins

Pin Name	Module	Type	Pin Mux	Description
DRI_CLK	DRI	I	N/A	Used for digital radio clock input if enabled by HW_DRI_CTRL_ENABLE_INPUTS. NOTE: This is the same as the LINE1R pin.
DRI_DATA	DRI	I	N/A	Used for digital radio data input if enabled by HW_DRI_CTRL_ENABLE_INPUTS. NOTE: This is the same as the LINE1L pin.

35.2.9. LCD Interface (LCDIF) Pins

Table 1019. LCDIF Interface Pins

Pin Name	Module	Type	Pin Mux	Description
LCD_BUSY	LCDIF	I	0	LCD Busy
	PINCTRL	I/O	3	GPIO1[21]
LCD_CS	LCDIF	O	0	LCD Interface Chip Select
	ETM	O	1	ETM_TCLK
	PINCTRL	I/O	3	GPIO1[19]
LCD_D00	LCDIF	O	0	LCD Interface Data 0
	ETM	O	1	ETM_DA0
	PINCTRL	I/O	3	GPIO1[0]
LCD_D01	LCDIF	O	0	LCD Interface Data 1
	ETM	O	1	ETM_DA1
	PINCTRL	I/O	3	GPIO1[1]

Table 1019. LCDIF Interface Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
LCD_D02	LCDIF	O	0	LCD Interface Data 2
	ETM	O	1	ETM_DA2
	PINCTRL	I/O	3	GPIO1[2]
LCD_D03	LCDIF	O	0	LCD Interface Data 3
	ETM	O	1	ETM_DA3
	PINCTRL	I/O	3	GPIO1[3]
LCD_D04	LCDIF	O	0	LCD Interface Data 4
	ETM	O	1	ETM_DA4
	PINCTRL	I/O	3	GPIO1[4]
LCD_D05	LCDIF	O	0	LCD Interface Data 5
	ETM	O	1	ETM_DA5
	PINCTRL	I/O	3	GPIO1[5]
LCD_D06	LCDIF	O	0	LCD Interface Data 6
	ETM	O	1	ETM_DA6
	PINCTRL	I/O	3	GPIO1[6]
LCD_D07	LCDIF	O	0	LCD Interface Data 7
	ETM	O	1	ETM_DA7
	PINCTRL	I/O	3	GPIO1[7]
LCD_D08	LCDIF	O	0	LCD Interface Data 8
	ETM	O	1	ETM_DB0
	PINCTRL	I/O	3	GPIO1[8]
LCD_D09	LCDIF	O	0	LCD Interface Data 9
	ETM	O	1	ETM_DB1
	PINCTRL	I/O	3	GPIO1[9]
LCD_D10	LCDIF	O	0	LCD Interface Data 10
	ETM	O	1	ETM_DB2
	PINCTRL	I/O	3	GPIO1[10]
LCD_D11	LCDIF	O	0	LCD Interface Data 11
	ETM	O	1	ETM_DB3
	PINCTRL	I/O	3	GPIO1[11]
LCD_D12	LCDIF	O	0	LCD Interface Data 12
	ETM	O	1	ETM_DB4
	PINCTRL	I/O	3	GPIO1[12]
LCD_D13	LCDIF	O	0	LCD Interface Data 13
	ETM	O	1	ETM_DB5
	PINCTRL	I/O	3	GPIO1[13]
LCD_D14	LCDIF	O	0	LCD Interface Data 14
	ETM	O	1	ETM_DB6
	PINCTRL	I/O	3	GPIO1[14]
LCD_D15	LCDIF	O	0	LCD Interface Data 15
	ETM	O	1	ETM_DB7
	SYSTEM	O	2	RTCK - JTAG Return Clock
	PINCTRL	I/O	3	GPIO1[15]

Table 1019. LCDIF Interface Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
LCD_RESET	LCDIF	O	0	LCD Interface Reset Out
	ETM	O	1	ETM_PSA1
	PINCTRL	I/O	3	GPIO1[16]
LCD_RS	LCDIF	O	0	LCD Interface Register Select
	ETM	O	1	ETM_PSA0
	PINCTRL	I/O	3	GPIO1[17]
LCD_WR	LCDIF	O	0	LCD Interface Data Write
	ETM	O	1	ETM_PSA2
	PINCTRL	I/O	3	GPIO1[18]

35.2.10. Power Pins

Table 1020. Power Pins

Pin Name	Module	Type	Pin Mux	Description
BATT	POWER	P		Battery Input / LRADC7–0
DCDC2_PFET	DCDC	P		DC-DC2 PFET Drain Connection
VDD5V	POWER	P		5-V Power Input
VDDA1	POWER			Analog Power 1
VDDD1	POWER	P		Digital Core Power 1 / LRADC7–1
VDDD2	POWER	P		Digital Core Power 2
VDDD3	POWER	P		Digital Core Power 3
VDDIO1	POWER	P		Digital I/O Power 1 / LRADC7–2
VDDIO2	POWER	P		Digital I/O Power 2
VDDIO4	POWER	P		Digital I/O Power 4
VSSA1	POWER			Analog Ground 1
VSSA3	POWER	P		Analog Ground 3 - DB
VSSD1	POWER	P		Digital Ground 1
VSSD2	POWER	P		Digital Ground 2
VSSD3	POWER	P		Digital Ground 3
VSSD4	POWER	P		Digital Ground 4
VSSD6	POWER	P		Digital Ground 6
VSSD7	POWER	P		Digital Ground 7

35.2.11. System Pins

Table 1021. System Pins

Pin Name	Module	Type	Pin Mux	Description
JTAG_RESET	SYSTEM	I		Debug Reset
JTAG_TCK	SYSTEM	I		Debug Clock
JTAG_TDI	SYSTEM	I		Debug Data In
JTAG_TDO	SYSTEM	O		Debug Data Out

Table 1021. System Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
JTAG_TMS	SYSTEM	I		Debug Test Mode Select
LCD_D15	LCDIF	O	0	LCD Interface Data 15
	ETM	O	1	ETM_DB7
	SYSTEM	O	2	**RTCK - JTAG Return Clock**
	PINCTRL	I/O	3	GPIO1[15]
PWM2	PWM	I/O	0	PWM
	ETM	O	1	ETM_PSB2
	SYSTEM	O	2	**RTCK - JTAG Return Clock**
	PINCTRL	I/O	3	GPIO3[12]
SSP_DETECT	SSP	I/O	0	Removable Card Detect
	SYSTEM	O	2	**RTCK - JTAG Return Clock**
	PINCTRL	I/O	3	GPIO0[25]
UART2_RTS	UART	O	0	High-Speed UART RTS Flow Control
	SYSTEM	O	1	**RTCK - JTAG Return Clock**
	IR	O	2	IR_CLK
	PINCTRL	I/O	3	GPIO1[23]
REF_RES	USB	A		USB Reference Resistor
RTC_XTALI	RTC	A		32.768-kHz Xtal In
RTC_XTALO	RTC	A		32.768-kHz Xtal Out
TESTMODE	SYSTEM	I		Test Mode Pin
VDDXTAL	CLOCK	A		Crystal Power Filter Cap - Cross Bond to Side 4
XTALI	CLOCK	A		Crystal In - 24 MHz
XTALO	CLOCK	A		Crystal Out - 24 MHz

35.2.12. Timer and PWM Pins

Table 1022. Timer and PWM Pins

Pin Name	Module	Type	Pin Mux	Description
PWM0	PWM	I/O	0	PWM
	ETM	O	1	ETM_TSYNCA
	UARTDBG	I	2	UART1_RX (Debug)
	PINCTRL	I/O	3	GPIO3[10]
PWM1	PWM	I/O	0	PWM
	ETM	O	1	ETM_TSYNCB
	UARTDBG	O	2	UART1_TX (Debug)
	PINCTRL	I/O	3	GPIO3[11]
PWM2	PWM	I/O	0	PWM
	ETM	O	1	ETM_PSB2
	SYSTEM	O	2	RTCK - JTAG Return Clock
	PINCTRL	I/O	3	GPIO3[12]

Table 1022. Timer and PWM Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
PWM3	PWM	I/O	0	PWM - 16ma Drive for SPDIF Out
	ETM		1	ETM_PSB0
	SPDIF	O	2	SPDIF Out
	PINCTRL	I/O	3	GPIO3[13]
PWM4	PWM	I/O	0	PWM - 16ma Drive for OTG Vbus
	ETM	O	1	ETM_PSB1
	PINCTRL	I/O	3	GPIO3[14]
ROTARYA	TIMER	I/O	0	Rotary Encoder A
	PINCTRL	I/O	3	GPIO3[15]
ROTARYB	TIMER	I/O	0	Rotary Encoder B
	PINCTRL	I/O	3	GPIO3[16]

35.2.13. USB Pins

Table 1023. USB Pins

Pin Name	Module	Type	Pin Mux	Description
REF_RES	USB	A		USB Reference Resistor
USB_DM	USB	A		USB Negative Data Line
USB_DP	USB	A		USB Positive Data Line
USB_OTG_ID	USB	A		USB OTG ID Sense

35.2.14. General-Purpose Input/Output (GPIO) Pins

Table 1024. Pin Control—GPIO Pins

Pin Name	Module	Type	Pin Mux	Description
GPMI_D00	GPMI	I/O	0	ATA/NAND Data 0
	PINCTRL	I/O	3	GPIO0[0]
GPMI_D01	GPMI	I/O	0	ATA/NAND DATA 1
	PINCTRL	I/O	3	GPIO0[1]
GPMI_D02	GPMI	I/O	0	ATA/NAND Data 2
	PINCTRL	I/O	3	GPIO0[2]
GPMI_D03	GPMI	I/O	0	ATA/NAND Data 3
	PINCTRL	I/O	3	GPIO0[3]
GPMI_D04	GPMI	I/O	0	ATA/NAND Data 4
	PINCTRL	I/O	3	GPIO0[4]
GPMI_D05	GPMI	I/O	0	ATA/NAND Data 5
	PINCTRL	I/O	3	GPIO0[5]
GPMI_D06	GPMI	I/O	0	ATA/NAND Data 6
	PINCTRL	I/O	3	GPIO0[6]
GPMI_D07	GPMI	I/O	0	ATA/NAND Data 7
	PINCTRL	I/O	3	GPIO0[7]

Table 1024. Pin Control—GPIO Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
GPMI_D08	GPMI	I/O	0	ATA/NAND Data 8
	EMI	O	1	EMI_A15
	PINCTRL	I/O	3	**GPIO0[8]**
GPMI_D09	GPMI	I/O	0	ATA/NAND Data 9
	ETM	O	1	EMI_A16
	PINCTRL	I/O	3	**GPIO0[9]**
GPMI_D10	GPMI	I/O	0	ATA/NAND Data 10
	EMI	O	1	EMI_A17
	PINCTRL	I/O	3	**GPIO0[10]**
GPMI_D11	GPMI	I/O	0	ATA/NAND Data 11
	EMI	O	1	EMI_A18
	PINCTRL	I/O	3	**GPIO0[11]**
GPMI_D12	GPMI	I/O	0	ATA/NAND Data 12
	EMI	O	1	EMI_A19
	GPMI	O	2	GPMI_CE0n
	PINCTRL	I/O	3	**GPIO0[12]**
GPMI_D13	GPMI	I/O	0	ATA/NAND Data 13
	EMI	O	1	EMI_A20
	GPMI	O	2	GPMI_CE1n
	PINCTRL	I/O	3	**GPIO0[13]**
GPMI_D14	GPMI	I/O	0	ATA/NAND Data 14
	EMI	O	1	EMI_A21
	GPMI	O	2	GPMI_CE2n
	PINCTRL	I/O	3	**GPIO0[14]**
GPMI_D15	GPMI	I/O	0	ATA/NAND Data 15
	EMI	O	1	EMI_A22
	GPMI	O	2	GPMI_CE3n
	PINCTRL	I/O	3	**GPIO0[15]**
GPMI_IRQ	GPMI	I	0	ATA INTRQ or NAND1 Ready/Busy#
	PINCTRL	I/O	3	**GPIO0[16]**
GPMI_RDN	GPMI	O	0	ATA DIOR-:HSTROBE or NAND Read Strobe
	PINCTRL	I/O	3	**GPIO0[17]**
GPMI_RDY	GPMI	I	0	ATA IORDY:DSTROBE or NAND0 Ready/Busy#
	PINCTRL	I/O	3	**GPIO0[18]**
GPMI_RDY3	GPMI	I/O	0	ATA DMARQ or NAND3 R/B#
	EMI	O	1	EMI_OEN
	PINCTRL	I/O	3	**GPIO0[19]**
GPMI_RDY2	GPMI	I/O	0	ATA DMACK or NAND2 Ready/Busy#
	PINCTRL	I/O	3	**GPIO0[20]**
GPMI_WRN	GPMI	O	0	ATA DIOW-:STOP or NAND Write Strobe
	PINCTRL	I/O	3	**GPIO0[21]**
GPMI_A0	GPMI	O	0	ATA_A0 or NAND CLE
	EMI	O	1	EMI_A23
	PINCTRL	I/O	3	**GPIO0[22]**

Table 1024. Pin Control—GPIO Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
GPMI_A1	GPMI	O	0	ATA_A1 or NAND ALE
	EMI	O	1	EMI_A24
	PINCTRL	I/O	3	**GPIO0[23]**
GPMI_A2	GPMI	O	0	ATA_A2
	EMI	O	1	EMI_A25
	PINCTRL	I/O	3	**GPIO0[24]**
SSP_DETECT	SSP	I/O	0	Removable Card Detect
	SYSTEM	O	2	RTCK - JTAG Return Clock
	PINCTRL	I/O	3	**GPIO0[25]**
SSP_CMD	SSP	I/O	0	SPI MOSI or MS SDIO or SD/MMC CMD
	PINCTRL	I/O	3	**GPIO0[26]**
SSP_SCK	SSP	I/O	0	SPI Serial Clock - (Bond to Pin 100 in 100 TQFP)
	PINCTRL	I/O	3	**GPIO0[27]**
SSP_DATA0	SSP	I/O	0	SPI MISO or SD/MMC DAT0
	PINCTRL	I/O	3	**GPIO0[28]**
SSP_DATA1	SSP	I/O	0	SD/MMC Data 1
	PINCTRL	I/O	3	**GPIO0[29]**
SSP_DATA2	SSP	I/O	0	SD/MMC Data 2
	PINCTRL	I/O	3	**GPIO0[30]**
SSP_DATA3	SSP	I/O	0	SPI Slave Select 0 or MS BS or SD/MMC DAT3
	PINCTRL	I/O	3	**GPIO0[31]**
LCD_D00	LCDIF	O	0	LCD Interface Data 0
	ETM	O	1	ETM_DA0
	PINCTRL	I/O	3	**GPIO1[0]**
LCD_D01	LCDIF	O	0	LCD Interface Data 1
	ETM	O	1	ETM_DA1
	PINCTRL	I/O	3	**GPIO1[1]**
LCD_D02	LCDIF	O	0	LCD Interface Data 2
	ETM	O	1	ETM_DA2
	PINCTRL	I/O	3	**GPIO1[2]**
LCD_D03	LCDIF	O	0	LCD Interface Data 3
	ETM	O	1	ETM_DA3
	PINCTRL	I/O	3	**GPIO1[3]**
LCD_D04	LCDIF	O	0	LCD Interface Data 4
	ETM	O	1	ETM_DA4
	PINCTRL	I/O	3	**GPIO1[4]**
LCD_D05	LCDIF	O	0	LCD Interface Data 5
	ETM	O	1	ETM_DA5
	PINCTRL	I/O	3	**GPIO1[5]**
LCD_D06	LCDIF	O	0	LCD Interface Data 6
	ETM	O	1	ETM_DA6
	PINCTRL	I/O	3	**GPIO1[6]**

Table 1024. Pin Control—GPIO Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
LCD_D07	LCDIF	O	0	LCD Interface Data 7
	ETM	O	1	ETM_DA7
	PINCTRL	I/O	3	**GPIO1[7]**
LCD_D08	LCDIF	O	0	LCD Interface Data 8
	ETM	O	1	ETM_DB0
	PINCTRL	I/O	3	**GPIO1[8]**
LCD_D09	LCDIF	O	0	LCD Interface Data 9
	ETM	O	1	ETM_DB1
	PINCTRL	I/O	3	**GPIO1[9]**
LCD_D10	LCDIF	O	0	LCD Interface Data 10
	ETM	O	1	ETM_DB2
	PINCTRL	I/O	3	**GPIO1[10]**
LCD_D11	LCDIF	O	0	LCD Interface Data 11
	ETM	O	1	ETM_DB3
	PINCTRL	I/O	3	**GPIO1[11]**
LCD_D12	LCDIF	O	0	LCD Interface Data 12
	ETM	O	1	ETM_DB4
	PINCTRL	I/O	3	**GPIO1[12]**
LCD_D13	LCDIF	O	0	LCD Interface Data 13
	ETM	O	1	ETM_DB5
	PINCTRL	I/O	3	**GPIO1[13]**
LCD_D14	LCDIF	O	0	LCD Interface Data 14
	ETM	O	1	ETM_DB6
	PINCTRL	I/O	3	**GPIO1[14]**
LCD_D15	LCDIF	O	0	LCD Interface Data 15
	ETM	O	1	ETM_DB7
	SYSTEM	O	2	RTCK - JTAG Return Clock
	PINCTRL	I/O	3	**GPIO1[15]**
LCD_RESET	LCDIF	O	0	LCD Interface Reset Out
	ETM	O	1	ETM_PSA1
	PINCTRL	I/O	3	**GPIO1[16]**
LCD_RS	LCDIF	O	0	LCD Interface Register Select
	ETM	O	1	ETM_PSA0
	PINCTRL	I/O	3	**GPIO1[17]**
LCD_WR	LCDIF	O	0	LCD Interface Data Write
	ETM	O	1	ETM_PSA2
	PINCTRL	I/O	3	**GPIO1[18]**
LCD_CS	LCDIF	O	0	LCD Interface Chip Select
	ETM	O	1	ETM_TCLK
	PINCTRL	I/O	3	**GPIO1[19]**
GPMI_RESETN	GPMI	O	0	ATA Reset, NAND Write Protect, or Renesas Reset
	ETM	O	1	EMI_RESET
	PINCTRL	I/O	3	**GPIO1[20]**

Table 1024. Pin Control—GPIO Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
LCD_BUSY	LCDIF	I	0	LCD Busy
	PINCTRL	I/O	3	**GPIO1[21]**
UART2_CTS	UART	I	0	High-Speed UART CTS Flow Control
	PINCTRL	I/O	3	**GPIO1[22]**
UART2_RTS	UART	O	0	High-Speed UART RTS Flow Control
	SYSTEM	O	1	RTCK - JTAG Return Clock
	IR	O	2	IR_CLK
	PINCTRL	I/O	3	**GPIO1[23]**
UART2_RX	UART	I	0	High-Speed UART RX
	IR	I	2	IR_RX
	PINCTRL	I/O	3	**GPIO1[24]**
UART2_TX	UART	I/O	0	High-Speed UART TX
	IR	O	2	IR_TX
	PINCTRL	I/O	3	**GPIO1[25]**
EMI_D00	EMI	I/O	0	EMI Data 0
	PINCTRL	I/O	3	**GPIO2[0]**
EMI_D01	EMI	I/O	0	EMI Data 1
	PINCTRL	I/O	3	**GPIO2[1]**
EMI_D02	EMI	I/O	0	EMI Data 2
	PINCTRL	I/O	3	**GPIO2[2]**
EMI_D03	EMI	I/O	0	EMI Data 3
	PINCTRL	I/O	3	**GPIO2[3]**
EMI_D04	EMI	I/O	0	EMI Data 4
	PINCTRL	I/O	3	**GPIO2[4]**
EMI_D05	EMI	I/O	0	EMI Data 5
	PINCTRL	I/O	3	**GPIO2[5]**
EMI_D06	EMI	I/O	0	EMI Data 6
	PINCTRL	I/O	3	**GPIO2[6]**
EMI_D07	EMI	I/O	0	EMI Data 7
	PINCTRL	I/O	3	**GPIO2[7]**
EMI_D08	EMI	I/O	0	EMI Data 8
	PINCTRL	I/O	3	**GPIO2[8]**
EMI_D09	EMI	I/O	0	EMI Data 9
	PINCTRL	I/O	3	**GPIO2[9]**
EMI_D10	EMI	I/O	0	EMI Data 10
	PINCTRL	I/O	3	**GPIO2[10]**
EMI_D11	EMI	I/O	0	EMI Data 11
	PINCTRL	I/O	3	**GPIO2[11]**
EMI_D12	EMI	I/O	0	EMI Data 12
	PINCTRL	I/O	3	**GPIO2[12]**
EMI_D13	EMI	I/O	0	EMI Data 13
	PINCTRL	I/O	3	**GPIO2[13]**
EMI_D14	EMI	I/O	0	EMI Data 14
	PINCTRL	I/O	3	**GPIO2[14]**

Table 1024. Pin Control—GPIO Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
EMI_D15	EMI	I/O	0	EMI Data 15
	PINCTRL	I/O	3	**GPIO2[15]**
EMI_A00	EMI	O	0	EMI Address 0
	PINCTRL	I/O	3	**GPIO2[16]**
EMI_A01	EMI	O	0	EMI Address 1
	PINCTRL	I/O	3	**GPIO2[17]**
EMI_A02	EMI	O	0	EMI Address 2
	PINCTRL	I/O	3	**GPIO2[18]**
EMI_A03	EMI	O	0	EMI Address 3
	PINCTRL	I/O	3	**GPIO2[19]**
EMI_A04	EMI	O	0	EMI Address 4
	PINCTRL	I/O	3	**GPIO2[20]**
EMI_A05	EMI	O	0	EMI Address 5
	PINCTRL	I/O	3	**GPIO2[21]**
EMI_A06	EMI	O	0	EMI Address 6
	PINCTRL	I/O	3	**GPIO2[22]**
EMI_A07	EMI	O	0	EMI Address 7
	PINCTRL	I/O	3	**GPIO2[23]**
EMI_A08	EMI	O	0	EMI Address 8
	PINCTRL	I/O	3	**GPIO2[24]**
EMI_A09	EMI	O	0	EMI Address 9
	PINCTRL	I/O	3	**GPIO2[25]**
EMI_A10	EMI	O	0	EMI Address 10
	PINCTRL	I/O	3	**GPIO2[26]**
EMI_A11	EMI	O	0	EMI Address 11
	PINCTRL	I/O	3	**GPIO2[27]**
EMI_A12	EMI	O	0	EMI Address 12
	PINCTRL	I/O	3	**GPIO2[28]**
EMI_A13	EMI	O	0	EMI Address 13 / SDRAM ba0
	PINCTRL	I/O	3	**GPIO2[29]**
EMI_A14	EMI	O	0	EMI Address 14 / SDRAM BA1
	PINCTRL	I/O	3	**GPIO2[30]**
EMI_RASN	EMI	O	0	EMI RASn
	PINCTRL	I/O	3	**GPIO2[31]**
EMI_CE0N	EMI	O	0	EMI CE0n
	GPMI	O	1	GPMI_CE0n
	PINCTRL	I/O	3	**GPIO3[0]**
EMI_CE1N	EMI	O	0	EMI CE1n
	GPMI	O	1	GPMI_CE1n
	PINCTRL	I/O	3	**GPIO3[1]**
EMI_CE2N	EMI	I/O	0	EMI CE2n
	GPMI	O	1	GPMI_CE2n
	PINCTRL	I/O	3	**GPIO3[2]**

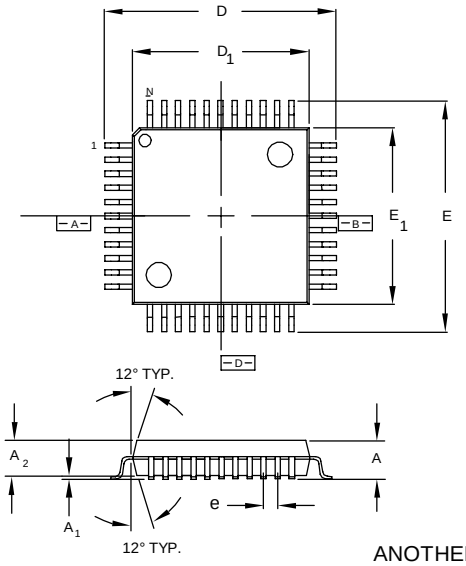
Table 1024. Pin Control—GPIO Pins (Continued)

Pin Name	Module	Type	Pin Mux	Description
EMI_CE3N	EMI	I/O	0	EMI CE3n
	GPMI	O	1	GPMI_CE3n
	PINCTRL	I/O	3	**GPIO3[3]**
EMI_CLK	EMI	O	0	EMI Clock
	PINCTRL	I/O	3	**GPIO3[4]**
EMI_CKE	EMI	O	0	EMI Clock Enable
	PINCTRL	I/O	3	**GPIO3[5]**
EMI_CASN	EMI	O	0	EMI CASn
	PINCTRL	I/O	3	**GPIO3[6]**
EMI_DQM0	EMI	O	0	EMI DQM0
	PINCTRL	I/O	3	**GPIO3[7]**
EMI_DQM1	EMI	O	0	EMI DQM1
	PINCTRL	I/O	3	**GPIO3[8]**
EMI_WEN	EMI	O	0	EMI WEN
	PINCTRL	I/O	3	**GPIO3[9]**
PWM0	PWM	I/O	0	PWM
	ETM	O	1	ETM_TSYNCA
	UARTDBG	I	2	UART1 RX (Debug)
	PINCTRL	I/O	3	**GPIO3[10]**
PWM1	PWM	I/O	0	PWM
	ETM	O	1	ETM_TSYNCB
	UARTDBG	O	2	UART1 TX (Debug)
	PINCTRL	I/O	3	**GPIO3[11]**
PWM2	PWM	I/O	0	PWM
	ETM	O	1	ETM_PSB2
	SYSTEM	O	2	RTCK - JTAG Return Clock
	PINCTRL	I/O	3	**GPIO3[12]**
PWM3	PWM	I/O	0	PWM - 16ma DRIVE for SPDIF Out
	ETM		1	ETM_PSB0
	SPDIF	O	2	SPDIF Out
	PINCTRL	I/O	3	**GPIO3[13]**
PWM4	PWM	I/O	0	PWM - 16ma Drive for OTG Vbus
	ETM	O	1	ETM_PSB1
	PINCTRL	I/O	3	**GPIO3[14]**
ROTARYA	TIMER	I/O	0	Rotary Encoder A
	PINCTRL	I/O	3	**GPIO3[15]**
ROTARYB	TIMER	I/O	0	Rotary Encoder B
	PINCTRL	I/O	3	**GPIO3[16]**
I2C_SCL	I2C	I/O	0	I^2C Serial Clock (o.d.)
	PINCTRL	I/O	3	**GPIO3[17]**
I2C_SDA	I2C	I/O	0	I^2C Serial Data (o.d.)
	PINCTRL	I/O	3	**GPIO3[18]**

36. PACKAGE DRAWINGS

The STMP36xx is available in two different packages. This chapter includes the package drawings for the 100-pin TQFP and the 169-pin fpBGA.

36.1. 100-Pin TQFP



BODY + 2.00 mm FOOTPRINT			
	LEADS	100L-1.4 THK.	
DIMS.	TOL. / L/F	.127	.152
A	MAX.	1.60	
A_1		.05 MIN./.15 MAX.	
A_2	±.05	1.40	
D	±.20	16.00	
D_1	±.05	14.00	
E	±.20	16.00	
E_1	±.05	14.00	
L	+.15/-.10	.60	
e	BASIC	.50	
b	±.05	.22	
Θ		0°-7D	
ddd	MAX.	.08	
ccc	MAX.	.08	

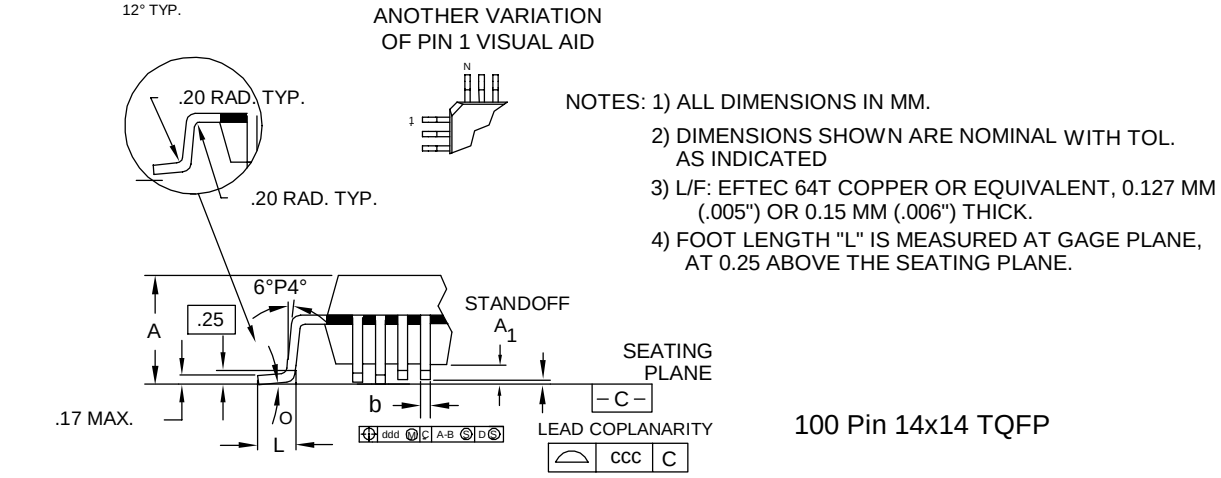
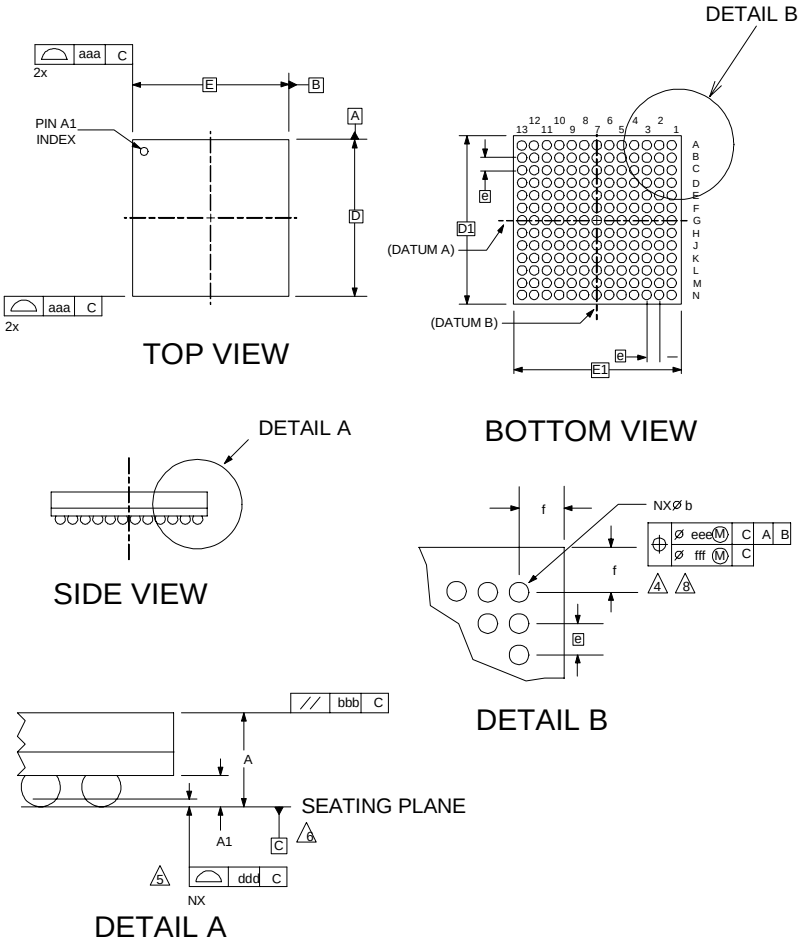


Figure 151. 100-Pin TQFP Package Drawing

36.2. 169-Pin fpBGA



169 fpBGA (11 x 11 mm)

DIMENSIONAL REFERENCES			
REF.	MIN.	NOM.	MAX.
A	1.14	1.30	1.43
A1	0.21	0.28	0.35
D	10.80	11.00	11.20
D1		9.60 BSC	
E	10.80	11.00	11.20
E1		9.60 BSC	
b	0.37	0.43	0.49
e		0.80 BSC	
f	0.60	0.70	0.80
aaa			0.10
bbb			0.10
ddd			0.15
eee			0.15
fff			0.08
M		13	
N		169	

ALL DIMENSIONS ARE IN MILLIMETERS.
'e' REPRESENTS THE BASIC SOLDER BALL GRID PITCH.
'M' REPRESENTS THE BASIC SOLDER BALL MATRIX SIZE. SYMBOL 'N'
IS THE NUMBER OF BALLS IN THE BALL MATRIX.
'b' IS MEASURABLE AT THE MAXIMUM SOLDER BALL DIAMETER PARALLEL
TO PRIMARY DATUM C.
DIMENSION 'ddd' IS MEASURED PARALLEL TO PRIMARY DATUM C.
PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL
CROWNS OF THE SOLDER BALLS.
SOLDER BALL DIAMETER 'b' REFERS TO POST REFLOW CONDITION. THE
PRE-REFLOW DIAMETER IS 0.40mm.
SUBSTRATE MATERIAL BASE IS BT RESIN.
THE OVERALL PACKAGE THICKNESS 'A' ALREADY CONSIDERS COLLAPSE BALLS.
DIMENSIONING AND TOLERENCING PER ASME Y14.5-1994.
PACKAGE DIMENSIONS TAKE REFERENCE TO JEDEC MO-205 F.

Figure 152. 169-Pin fpBGA Package Drawing

37. STMP36XX PART NUMBERS AND ORDERING INFORMATION

The STMP36xx family comprises a large set of parts targeted at specific applications and customers. Table 1025 summarizes the family members and provides part numbers for order placement.

Customers with prepaid royalties or other royalty arrangements for certain intellectual property items can order parts without the corresponding royalty fees included in the purchase price. Currently, the only royalty options are for certain MP3 items; see www.mp3licensing.com. The letter N at the end of the part number signifies a part that does not have the royalty payment included in the purchase price.

Table 1025. Part Numbers for STMP36xx Family Members

Part Number	Royalty	Package	Description	Available
STMP3660XXBBEB1M	MP3 Decode License Included	169-Pin fpBGA	MP3 Encode Enabled	4Q05
STMP3660XXBBEB1N	No MP3 Decode License			
STMP3650XXBBEB1M	MP3 Decode License Included	169-Pin fpBGA	No MP3 Encode Support	
STMP3650XXBBEB1N	No MP3 Decode License			
STMP3640XXBBEB1M	MP3 Decode License Included	169-Pin fpBGA	2MB SDRAM Support No USB Host No USB OTG MP3 Encode Enabled	1Q06
STMP3640XXBBEB1N	No MP3 Decode License			
STMP3630XXBBEB1M	MP3 Decode License Included	169-Pin fpBGA	2MB SDRAM Support No USB Host No USB OTG No MP3 Encode Support	
STMP3630XXBBEB1N	No MP3 Decode License			

Table 1025. Part Numbers for STMP36xx Family Members (Continued)

Part Number	Royalty	Package	Description	Available
STMP3620XXBBEB1M	MP3 Decode License Included	169-Pin fpBGA	No SDRAM or NOR support No IR Support No USB Host No USB OTG No MPEG4 Decode Support MP3 Encode Enabled	2Q06
STMP3620XXBBEB1N	No MP3 Decode License			
STMP3610XXBBEB1M	MP3 Decode License Included	169-Pin fpBGA	No SDRAM or NOR Support No IR Support No USB Host No USB OTG No MPEG4 Decode Support No MP3 Encode Support	
STMP3610XXBBEB1N	No MP3 Decode License			
STMP3620XXLAEB1M	MP3 Decode License Included	100-Pin TQFP	No LRADC Channels 2–5 No Application UART No SDRAM or NOR Support No 32-kHz RTC XTAL Driver No Speaker Driver 8-Bit LCD Data Bus No IR Support No USB Host No USB OTG No MPEG4 Decode Support MP3 Encode Enabled	
STMP3620XXLAEB1N	No MP3 Decode License			
STMP3610XXLAEB1M	MP3 Decode License Included	100-Pin TQFP	No LRADC Channels 2–5 No Application UART No SDRAM or NOR Support No 32-kHz RTC XTAL Driver No Speaker Driver 8-Bit LCD Data Bus No IR Support No USB Host No USB OTG No MPEG4 Decode Support No MP3 Encode Support	
STMP3610XXLAEB1N	No MP3 Decode License			

APPENDIX:ACRONYMS AND ABBREVIATIONS

This appendix includes definitions for many of the acronyms and abbreviations found in this product data sheet.

AAC: Advanced Audio Coding
ADC: Analog-to-Digital Converter
ADC: Adaptive Differential Pulse-Code Modulation
AHB: Advanced High-performance Bus
AIO: Analog Input/Output
AMBA: Advanced Microcontroller Bus Architecture
APB: Advanced Peripheral Bus
APBH: Advanced Peripheral Bus—HCLK Domain
APBX: Advanced Peripheral Bus—XCLK Domain
ARC: ARC International (corporate name)
ARM: Advanced RISC Machine (formerly Acorn RISC Machine)
ATA: Advanced Technology Attachment (hard drive interface)
AVC: Adaptive Voltage Control
BATT: Battery
BIST: Built-In Self-Test
BKPT: Breakpoint
CLKCTRL: Clock Control
CP: Charge Pump
CPUCLK: Processor (ARM CPU) Clock (see Table 9. “Clock Domains” on page 48.)
CTS: Clear To Send
DABT: Data Abort
DAC: Digital-to-Analog Converter
dB: Decibel
DC: Direct Current
DFLPT: Default First-Level Page Table
DIGCTL: Digital Control
DIO: Digital Input/Output
DiVX: Digital video codec created by DivXNetworks, Inc.
ECC: Error Correction Code
EL: Electroluminescent
EMI: External Memory Interface
EMICLK: EMI Clock (see Table 9. “Clock Domains” on page 48.)
ETM: Embedded Trace Macrocell
FIQ: Fast Peripheral Interrupt
FIR: Finite Impulse Response; also Fast Infrared
FLPT: First-Level Page Table

FREQ: Frequency
FS: Full-Speed
FSM: Finite State Machine?
GPIO: General-Purpose Input/Output
GPMI: General-Purpose Media Interface
GPMICLK: GMPI Clock (see Table 9. “Clock Domains” on page 48.)
HCLK: Main and HBUS Peripherals Clock (see Table 9. “Clock Domains” on page 48.)
HS: High-Speed
HW: Hardware
H.264: High-Compression Digital Video Codec
ICOLL: Interrupt Collector
IR: Infrared
IrDA: Infrared Data Association
IROVCLK: IR Clock (sourced from PLL; see Table 9. “Clock Domains” on page 48.)
IRCLK: IR Clock (source from IROVCLK; see Table 9. “Clock Domains” on page 48.)
IRQ: Normal Peripheral Interrupt
ISR: Interrupt Service Register
JEDEC: Joint Electron Device Engineering Council
JPEG: Joint Photographic Experts Group (computer image format)
Li-Ion: Lithium Ion (battery type)
LRADC: Low Resolution ADC
MATT: Multi-chip Attachment mode
MIR: Mid Infrared
MPEG4: Motion Picture Experts Group 4 (standard for compressed video at 64 kbps)
MP3: Moving Picture Experts Group Layer-3 Audio
Mux: Multiplexer
NiMH: Nickel Metal Hydride
NRZI: Non-Return to Zero Inverted
OTG: On the Go
PABT: Instruction Pre-Fetch Abort
PDA: Personal Digital Assistant
PDDRM: Portable Device Digital Rights Management (DRM9)
PFD: Phase/Frequency Detector
PFM: Pulse Frequency Modulation
PHY: Physical Layer Protocol
PLL: Phase-Locked Loop
PWM: Pulse Width Modulation
RTC: Real-Time Clock
RTS: Request To Send

RMW: Read-Modify-Write
SDIO: Secure Digital Input/Output
SDK: Software Development Kit
SIR: Serial Infrared
SNR: Signal-to-Noise Ratio
SOC: System-on-a-Chip
SPDIF: Sony-Philips Digital Interface Format
SPDIFCLK: SPDIF Clock (see Table 9. “Clock Domains” on page 48.)
SWI: Software Interrupt
TBD: To Be Determined
THD: Total Harmonic Distortion
TPC: Transfer Protocol Commands
TQFP: Thin Quad Flat Pack
UNDEF: Undefined instruction
UDMA: Ultra Direct Memory Access
UTMI: USB 2.0 Transceiver Macrocell Interface
VAG: Analog Ground Voltage
VBG: Internal Bandgap Voltage
VCO: Variable Crystal Oscillator
VDDA: Analog Power
VDDD: Digital Power
VFIR: Very Fast IrDA
WMDRM10: Windows Media Digital Rights Management 10 (Janus)
WMA: Windows Media Audio
XCLK: XBUS Peripherals Clock (see Table 9. “Clock Domains” on page 48.)

INDEX: REGISTER NAMES

This index of register names appears in alphabetical order by register mnemonic. It includes the register address and the page number in the data sheet where each register is described.

DFLPT_ENTRY_0000 ..... 0x800C0000 121
DFLPT_ENTRY_2048 ..... 0x800C2000 120
DFLPT_ENTRY4080 ..... 0x800C3FC0 119
DFLPT_ENTRY4081 ..... 0x800C3FC4 119
DFLPT_ENTRY4082 ..... 0x800C3FC8 119
DFLPT_ENTRY4083 ..... 0x800C3FCC 119
DFLPT_ENTRY4084 ..... 0x800C3FD0 119
DFLPT_ENTRY4085 ..... 0x800C3FD4 119
DFLPT_ENTRY4086 ..... 0x800C3FD8 119
DFLPT_ENTRY4087 ..... 0x800C3FDC 119
DFLPT_ENTRY4088 ..... 0x800C3FE0 119
DFLPT_ENTRY4089 ..... 0x800C3FE4 119
DFLPT_ENTRY4090 ..... 0x800C3FE8 119
DFLPT_ENTRY4091 ..... 0x800C3FEC 119
DFLPT_ENTRY4092 ..... 0x800C3FF0 119
DFLPT_ENTRY4093 ..... 0x800C3FF4 119
DFLPT_ENTRY4094 ..... 0x800C3FF8 119
DFLPT_ENTRY4095 ..... 0x800C3FFC 118
HW_APBH_CH0_BAR 0x80004060 202
HW_APBH_CH0_CMD 0x80004050 200
HW_APBH_CH0_CURCMDAR 0x80004030 199
HW_APBH_CH0_DEBUG1 ..... 0x80004080 203
HW_APBH_CH0_DEBUG2 ..... 0x80004090 205
HW_APBH_CH0_NXTCMDAR 0x80004040 199
HW_APBH_CH0_SEMA 0x80004070 202
HW_APBH_CH1_BAR 0x800040D0 209
HW_APBH_CH1_CMD 0x800040C0 207
HW_APBH_CH1_CURCMDAR 0x800040A0 206
HW_APBH_CH1_DEBUG1 ..... 0x800040F0 211
HW_APBH_CH1_DEBUG2 ..... 0x80004100 212
HW_APBH_CH1_NXTCMDAR 0x800040B0 207
HW_APBH_CH1_SEMA 0x800040E0 210
HW_APBH_CH2_BAR 0x80004140 216
HW_APBH_CH2_CMD 0x80004130 214
HW_APBH_CH2_CURCMDAR 0x80004110 213
HW_APBH_CH2_DEBUG1 ..... 0x80004160 218
HW_APBH_CH2_DEBUG2 ..... 0x80004170 219
HW_APBH_CH2_NXTCMDAR 0x80004120 214
HW_APBH_CH2_SEMA 0x80004150 217
HW_APBH_CH3_BAR 0x800041B0 223
HW_APBH_CH3_CMD 0x800041A0 221
HW_APBH_CH3_CURCMDAR 0x80004180 220
HW_APBH_CH3_DEBUG1 ..... 0x800041D0 225
HW_APBH_CH3_DEBUG2 ..... 0x800041E0 226
HW_APBH_CH3_NXTCMDAR 0x80004190 221
HW_APBH_CH3_SEMA 0x800041C0 224
HW_APBH_CH4_BAR 0x80004220 230
HW_APBH_CH4_CMD 0x80004210 228
HW_APBH_CH4_CURCMDAR 0x800041F0 227
HW_APBH_CH4_DEBUG1 ..... 0x80004240 232
HW_APBH_CH4_DEBUG2 ..... 0x80004250 233
HW_APBH_CH4_NXTCMDAR 0x80004200 228
HW_APBH_CH4_SEMA 0x80004230 231
HW_APBH_CH5_BAR 0x80004290 237
HW_APBH_CH5_CMD 0x80004280 235
HW_APBH_CH5_CURCMDAR 0x80004260 234
HW_APBH_CH5_DEBUG1 ..... 0x800042B0 239

HW_APBH_CH5_DEBUG2 ..... 0x800042C0 240
HW_APBH_CH5_NXTCMDAR 0x80004270 235
HW_APBH_CH5_SEMA 0x800042A0 238
HW_APBH_CH6_BAR 0x80004300 244
HW_APBH_CH6_CMD 0x800042F0 242
HW_APBH_CH6_CURCMDAR 0x800042D0 241
HW_APBH_CH6_DEBUG1 ..... 0x80004320 246
HW_APBH_CH6_DEBUG2 ..... 0x80004330 247
HW_APBH_CH6_NXTCMDAR 0x800042E0 242
HW_APBH_CH6_SEMA 0x80004310 245
HW_APBH_CH7_BAR 0x80004370 251
HW_APBH_CH7_CMD 0x80004360 249
HW_APBH_CH7_CURCMDAR 0x80004340 248
HW_APBH_CH7_DEBUG1 ..... 0x80004390 253
HW_APBH_CH7_DEBUG2 ..... 0x800043A0 254
HW_APBH_CH7_NXTCMDAR 0x80004350 249
HW_APBH_CH7_SEMA 0x80004380 252
HW_APBH_CTRL0 ..... 0x80004000 195
HW_APBH_CTRL0_CLR 0x80004008 195
HW_APBH_CTRL0_SET 0x80004004 195
HW_APBH_CTRL0_TOG 0x8000400C 195
HW_APBH_CTRL1 ..... 0x80004010 196
HW_APBH_CTRL1_CLR 0x80004018 196
HW_APBH_CTRL1_SET 0x80004014 196
HW_APBH_CTRL1_TOG 0x8000401C 196
HW_APBH_DEVSEL 0x80004020 198
HW_APBX_CH0_BAR 0x80024060 270
HW_APBX_CH0_CMD 0x80024050 268
HW_APBX_CH0_CURCMDAR 0x80024030 267
HW_APBX_CH0_DEBUG1 ..... 0x80024080 271
HW_APBX_CH0_DEBUG2 ..... 0x80024090 273
HW_APBX_CH0_NXTCMDAR 0x80024040 267
HW_APBX_CH0_SEMA 0x80024070 270
HW_APBX_CH1_BAR 0x800240D0 277
HW_APBX_CH1_CMD 0x800240C0 275
HW_APBX_CH1_CURCMDAR 0x800240A0 274
HW_APBX_CH1_DEBUG1 ..... 0x800240F0 278
HW_APBX_CH1_DEBUG2 ..... 0x80024100 280
HW_APBX_CH1_NXTCMDAR 0x800240B0 275
HW_APBX_CH1_SEMA 0x800240E0 277
HW_APBX_CH2_BAR 0x80024140 284
HW_APBX_CH2_CMD 0x80024130 282
HW_APBX_CH2_CURCMDAR 0x80024110 281
HW_APBX_CH2_DEBUG1 ..... 0x80024160 285
HW_APBX_CH2_DEBUG2 ..... 0x80024170 287
HW_APBX_CH2_NXTCMDAR 0x80024120 282
HW_APBX_CH2_SEMA 0x80024150 284
HW_APBX_CH3_BAR 0x800241B0 291
HW_APBX_CH3_CMD 0x800241A0 289
HW_APBX_CH3_CURCMDAR 0x80024180 288
HW_APBX_CH3_DEBUG1 ..... 0x800241D0 292
HW_APBX_CH3_DEBUG2 ..... 0x800241E0 294
HW_APBX_CH3_NXTCMDAR 0x80024190 289
HW_APBX_CH3_SEMA 0x800241C0 291
HW_APBX_CH4_BAR 0x80024220 298
HW_APBX_CH4_CMD 0x80024210 296
HW_APBX_CH4_CURCMDAR 0x800241F0 295
HW_APBX_CH4_DEBUG1 ..... 0x80024240 299
HW_APBX_CH4_DEBUG2 ..... 0x80024250 301
HW_APBX_CH4_NXTCMDAR 0x80024200 296
HW_APBX_CH4_SEMA 0x80024230 298
HW_APBX_CH5_BAR 0x80024290 305
HW_APBX_CH5_CMD 0x80024280 303

HW_APBX_CH5_CURCMDAR 0x80024260 302
HW_APBX_CH5_DEBUG1 ..... 0x800242B0 306
HW_APBX_CH5_DEBUG2 ..... 0x800242C0 308
HW_APBX_CH5_NXTCMDAR 0x80024270 303
HW_APBX_CH5_SEMA 0x800242A0 305
HW_APBX_CH6_BAR 0x80024300 312
HW_APBX_CH6_CMD 0x800242F0 310
HW_APBX_CH6_CURCMDAR 0x800242D0 309
HW_APBX_CH6_DEBUG1 ..... 0x80024320 313
HW_APBX_CH6_DEBUG2 ..... 0x80024330 315
HW_APBX_CH6_NXTCMDAR 0x800242E0 310
HW_APBX_CH6_SEMA 0x80024310 312
HW_APBX_CH7_BAR 0x80024370 319
HW_APBX_CH7_CMD 0x80024360 317
HW_APBX_CH7_CURCMDAR 0x80024340 316
HW_APBX_CH7_DEBUG1 ..... 0x80024390 320
HW_APBX_CH7_DEBUG2 ..... 0x800243A0 322
HW_APBX_CH7_NXTCMDAR 0x80024350 317
HW_APBX_CH7_SEMA 0x80024380 319
HW_APBX_CTRL0 ..... 0x80024000 263
HW_APBX_CTRL0_CLR 0x80024008 263
HW_APBX_CTRL0_SET 0x80024004 263
HW_APBX_CTRL0_TOG 0x8002400C 263
HW_APBX_CTRL1 ..... 0x80024010 264
HW_APBX_CTRL1_CLR 0x80024018 264
HW_APBX_CTRL1_SET 0x80024014 264
HW_APBX_CTRL1_TOG 0x8002401C 264
HW_APBX_DEVSEL 0x80024020 266
HW_AUDIOIN_ADCDEBUG 0x8004C040 638
HW_AUDIOIN_ADCDEBUG_CLR 0x8004C048 638
HW_AUDIOIN_ADCDEBUG_SET 0x8004C044 638
HW_AUDIOIN_ADCDEBUG_TOG 0x8004C04C 638
HW_AUDIOIN_ADCSRR 0x8004C020 634
HW_AUDIOIN_ADCSRR_CLR 0x8004C028 634
HW_AUDIOIN_ADCSRR_SET 0x8004C024 634
HW_AUDIOIN_ADCSRR_TOG 0x8004C02C 634
HW_AUDIOIN_ADCVOL 0x8004C050 640
HW_AUDIOIN_ADCVOL_CLR 0x8004C058 640
HW_AUDIOIN_ADCVOL_SET 0x8004C054 640
HW_AUDIOIN_ADCVOL_TOG 0x8004C05C 640
HW_AUDIOIN_ADCVOLUME 0x8004C030 636
HW_AUDIOIN_ADCVOLUME_CLR 0x8004C038 636
HW_AUDIOIN_ADCVOLUME_SET 0x8004C034 636
HW_AUDIOIN_ADCVOLUME_TOG 0x8004C03C 636
HW_AUDIOIN_ANACLKCTRL 0x8004C070 643
HW_AUDIOIN_ANACLKCTRL_CLR 0x8004C078 643
HW_AUDIOIN_ANACLKCTRL_SET 0x8004C074 643
HW_AUDIOIN_ANACLKCTRL_TOG 0x8004C07C 643
HW_AUDIOIN_CTRL 0x8004C000 631
HW_AUDIOIN_CTRL_CLR 0x8004C008 631
HW_AUDIOIN_CTRL_SET 0x8004C004 631
HW_AUDIOIN_CTRL_TOG 0x8004C00C 631
HW_AUDIOIN_DATA 0x8004C080 644
HW_AUDIOIN_DATA_CLR 0x8004C088 644
HW_AUDIOIN_DATA_SET 0x8004C084 644
HW_AUDIOIN_DATA_TOG 0x8004C08C 644
HW_AUDIOIN_MICLINE 0x8004C060 641
HW_AUDIOIN_MICLINE_CLR 0x8004C068 641
HW_AUDIOIN_MICLINE_SET 0x8004C064 641
HW_AUDIOIN_MICLINE_TOG 0x8004C06C 641
HW_AUDIOIN_STAT 0x8004C010 634
HW_AUDIOIN_STAT_CLR 0x8004C018 634
HW_AUDIOIN_STAT_SET 0x8004C014 634

HW_AUDIOIN_STAT_TOG 0x8004C01C 634
HW_AUDIOOUT_ANACLKCTRL 0x800480e0 676
HW_AUDIOOUT_ANACLKCTRL_CLR 0x800480e8 676
HW_AUDIOOUT_ANACLKCTRL_SET 0x800480e4 676
HW_AUDIOOUT_ANACLKCTRL_TOG 0x800480eC 676
HW_AUDIOOUT_ANACTRL 0x80048090 671
HW_AUDIOOUT_ANACTRL_CLR 0x80048098 671
HW_AUDIOOUT_ANACTRL_SET 0x80048094 671
HW_AUDIOOUT_ANACTRL_TOG 0x8004809C 671
HW_AUDIOOUT_BISTCTRL 0x800480b0 675
HW_AUDIOOUT_BISTCTRL_CLR 0x800480b8 675
HW_AUDIOOUT_BISTCTRL_SET 0x800480b4 675
HW_AUDIOOUT_BISTCTRL_TOG 0x800480bC 675
HW_AUDIOOUT_BISTSTAT0 ..... 0x800480c0 675
HW_AUDIOOUT_BISTSTAT0_CLR 0x800480c8 675
HW_AUDIOOUT_BISTSTAT0_SET 0x800480c4 675
HW_AUDIOOUT_BISTSTAT0_TOG 0x800480cC 675
HW_AUDIOOUT_BISTSTAT1 ..... 0x800480d0 676
HW_AUDIOOUT_BISTSTAT1_CLR 0x800480d8 676
HW_AUDIOOUT_BISTSTAT1_SET 0x800480d4 676
HW_AUDIOOUT_BISTSTAT1_TOG 0x800480dC 676
HW_AUDIOOUT_CTRL 0x80048000 657
HW_AUDIOOUT_CTRL_CLR 0x80048008 657
HW_AUDIOOUT_CTRL_SET 0x80048004 657
HW_AUDIOOUT_CTRL_TOG 0x8004800C 657
HW_AUDIOOUT_DACDEBUG 0x80048040 664
HW_AUDIOOUT_DACDEBUG_CLR 0x80048048 664
HW_AUDIOOUT_DACDEBUG_SET 0x80048044 664
HW_AUDIOOUT_DACDEBUG_TOG 0x8004804C 664
HW_AUDIOOUT_DACSRR 0x80048020 660
HW_AUDIOOUT_DACSRR_CLR 0x80048028 660
HW_AUDIOOUT_DACSRR_SET 0x80048024 660
HW_AUDIOOUT_DACSRR_TOG 0x8004802C 660
HW_AUDIOOUT_DACVOLUME 0x80048030 662
HW_AUDIOOUT_DACVOLUME_CLR 0x80048038 662
HW_AUDIOOUT_DACVOLUME_SET 0x80048034 662
HW_AUDIOOUT_DACVOLUME_TOG 0x8004803C 662
HW_AUDIOOUT_DATA 0x800480f0 677
HW_AUDIOOUT_DATA_CLR 0x800480f8 677
HW_AUDIOOUT_DATA_SET 0x800480f4 677
HW_AUDIOOUT_DATA_TOG 0x800480fC 677
HW_AUDIOOUT_HPVOL 0x80048050 665
HW_AUDIOOUT_HPVOL_CLR 0x80048058 665
HW_AUDIOOUT_HPVOL_SET 0x80048054 665
HW_AUDIOOUT_HPVOL_TOG 0x8004805C 665
HW_AUDIOOUT_PWRDN 0x80048070 667
HW_AUDIOOUT_PWRDN_CLR 0x80048078 667
HW_AUDIOOUT_PWRDN_SET 0x80048074 667
HW_AUDIOOUT_PWRDN_TOG 0x8004807C 667
HW_AUDIOOUT_REFCTRL 0x80048080 668
HW_AUDIOOUT_REFCTRL_CLR 0x80048088 668
HW_AUDIOOUT_REFCTRL_SET 0x80048084 668
HW_AUDIOOUT_REFCTRL_TOG 0x8004808C 668
HW_AUDIOOUT_SPKRVOL 0x80048060 666
HW_AUDIOOUT_SPKRVOL_CLR 0x80048068 666
HW_AUDIOOUT_SPKRVOL_SET 0x80048064 666
HW_AUDIOOUT_SPKRVOL_TOG 0x8004806C 666
HW_AUDIOOUT_STAT 0x80048010 659
HW_AUDIOOUT_STAT_CLR 0x80048018 659
HW_AUDIOOUT_STAT_SET 0x80048014 659
HW_AUDIOOUT_STAT_TOG 0x8004801C 660
HW_AUDIOOUT_TEST 0x800480a0 673
HW_AUDIOOUT_TEST_CLR 0x800480a8 673

HW_AUDIOOUT_TEST_SET 0x800480a4 673
HW_AUDIOOUT_TEST_TOG 0x800480aC 673
HW_CLKCTRL_CPUCLKCTRL 0x80040020 58
HW_CLKCTRL_EMICLKCTRL 0x800400b0 67
HW_CLKCTRL_GPMICLKCTRL 0x80040090 65
HW_CLKCTRL_HBUSCLKCTRL 0x80040030 59
HW_CLKCTRL_IRCLKCTRL 0x800400c0 67
HW_CLKCTRL_OCRAMCLKCTRL 0x80040060 63
HW_CLKCTRL_PLLCTRL0 ..... 0x80040000 56
HW_CLKCTRL_PLLCTRL0_CLR 0x80040008 56
HW_CLKCTRL_PLLCTRL0_SET 0x80040004 56
HW_CLKCTRL_PLLCTRL0_TOG 0x8004000C 56
HW_CLKCTRL_PLLCTRL1 ..... 0x80040010 58
HW_CLKCTRL_PLLCTRL1_CLR 0x80040018 58
HW_CLKCTRL_PLLCTRL1_SET 0x80040014 58
HW_CLKCTRL_PLLCTRL1_TOG 0x8004001C 58
HW_CLKCTRL_SPDIFCLKCTRL 0x800400a0 66
HW_CLKCTRL_SSPCLKCTRL 0x80040080 64
HW_CLKCTRL_UTMICLKCTRL 0x80040070 63
HW_CLKCTRL_XBUSCLKCTRL 0x80040040 61
HW_CLKCTRL_XTALCLKCTRL 0x80040050 62
HW_DIGCTL_1TRAM_BIST_CSR 0x8001C0E0 140
HW_DIGCTL_1TRAM_BIST_CSR_CLR 0x8001C0E8 140
HW_DIGCTL_1TRAM_BIST_CSR_SET 0x8001C0E4 140
HW_DIGCTL_1TRAM_BIST_CSR_TOG 0x8001C0EC 140
HW_DIGCTL_1TRAM_BIST_REPAIR0 ..... 0x8001C0F0 140
HW_DIGCTL_1TRAM_BIST_REPAIR0_CLR 0x8001C0F8 141
HW_DIGCTL_1TRAM_BIST_REPAIR0_SET 0x8001C0F4 141
HW_DIGCTL_1TRAM_BIST_REPAIR0_TOG 0x8001C0FC 141
HW_DIGCTL_1TRAM_BIST_REPAIR1 ..... 0x8001C100 141
HW_DIGCTL_1TRAM_BIST_REPAIR1_CLR 0x8001C108 141
HW_DIGCTL_1TRAM_BIST_REPAIR1_SET 0x8001C104 141
HW_DIGCTL_1TRAM_BIST_REPAIR1_TOG 0x8001C10C 141
HW_DIGCTL_1TRAM_STATUS0 ..... 0x8001C110 142
HW_DIGCTL_1TRAM_STATUS0_CLR 0x8001C118 142
HW_DIGCTL_1TRAM_STATUS0_SET 0x8001C114 142
HW_DIGCTL_1TRAM_STATUS0_TOG 0x8001C11C 142
HW_DIGCTL_1TRAM_STATUS1 ..... 0x8001C120 142
HW_DIGCTL_1TRAM_STATUS1_CLR 0x8001C128 142
HW_DIGCTL_1TRAM_STATUS1_SET 0x8001C124 142
HW_DIGCTL_1TRAM_STATUS1_TOG 0x8001C12C 142
HW_DIGCTL_1TRAM_STATUS10 ..... 0x8001C1B0 147
HW_DIGCTL_1TRAM_STATUS10_CLR 0x8001C1B8 147
HW_DIGCTL_1TRAM_STATUS10_SET 0x8001C1B4 147
HW_DIGCTL_1TRAM_STATUS10_TOG 0x8001C1BC 147
HW_DIGCTL_1TRAM_STATUS11 ..... 0x8001C1C0 147
HW_DIGCTL_1TRAM_STATUS11_CLR 0x8001C1C8 147
HW_DIGCTL_1TRAM_STATUS11_SET 0x8001C1C4 147
HW_DIGCTL_1TRAM_STATUS11_TOG 0x8001C1CC 148
HW_DIGCTL_1TRAM_STATUS12 ..... 0x8001C1D0 148
HW_DIGCTL_1TRAM_STATUS12_CLR 0x8001C1D8 148
HW_DIGCTL_1TRAM_STATUS12_SET 0x8001C1D4 148
HW_DIGCTL_1TRAM_STATUS12_TOG 0x8001C1DC 148
HW_DIGCTL_1TRAM_STATUS13 ..... 0x8001C1E0 149
HW_DIGCTL_1TRAM_STATUS13_CLR 0x8001C1E8 149
HW_DIGCTL_1TRAM_STATUS13_SET 0x8001C1E4 149
HW_DIGCTL_1TRAM_STATUS13_TOG 0x8001C1EC 149
HW_DIGCTL_1TRAM_STATUS2 ..... 0x8001C130 143
HW_DIGCTL_1TRAM_STATUS2_CLR 0x8001C138 143
HW_DIGCTL_1TRAM_STATUS2_SET 0x8001C134 143
HW_DIGCTL_1TRAM_STATUS2_TOG 0x8001C13C 143
HW_DIGCTL_1TRAM_STATUS3 ..... 0x8001C140 143
HW_DIGCTL_1TRAM_STATUS3_CLR 0x8001C148 143

HW_DIGCTL_1TRAM_STATUS3_SET 0x8001C144 143
HW_DIGCTL_1TRAM_STATUS3_TOG ..... 0x8001C14C.....143
HW_DIGCTL_1TRAM_STATUS4 ..... 0x8001C150 144
HW_DIGCTL_1TRAM_STATUS4_CLR 0x8001C158 144
HW_DIGCTL_1TRAM_STATUS4_SET 0x8001C154 144
HW_DIGCTL_1TRAM_STATUS4_TOG ..... 0x8001C15C.....144
HW_DIGCTL_1TRAM_STATUS5 ..... 0x8001C160 144
HW_DIGCTL_1TRAM_STATUS5_CLR 0x8001C168 144
HW_DIGCTL_1TRAM_STATUS5_SET 0x8001C164 144
HW_DIGCTL_1TRAM_STATUS5_TOG ..... 0x8001C16C.....144
HW_DIGCTL_1TRAM_STATUS6 ..... 0x8001C170 145
HW_DIGCTL_1TRAM_STATUS6_CLR 0x8001C178 145
HW_DIGCTL_1TRAM_STATUS6_SET 0x8001C174 145
HW_DIGCTL_1TRAM_STATUS6_TOG ..... 0x8001C17C.....145
HW_DIGCTL_1TRAM_STATUS7 ..... 0x8001C180 145
HW_DIGCTL_1TRAM_STATUS7_CLR 0x8001C188 145
HW_DIGCTL_1TRAM_STATUS7_SET 0x8001C184 145
HW_DIGCTL_1TRAM_STATUS7_TOG ..... 0x8001C18C.....145
HW_DIGCTL_1TRAM_STATUS8 ..... 0x8001C190 146
HW_DIGCTL_1TRAM_STATUS8_CLR 0x8001C198 146
HW_DIGCTL_1TRAM_STATUS8_SET 0x8001C194 146
HW_DIGCTL_1TRAM_STATUS8_TOG ..... 0x8001C19C.....146
HW_DIGCTL_1TRAM_STATUS9 ..... 0x8001C1A0 146
HW_DIGCTL_1TRAM_STATUS9_CLR 0x8001C1A8 146
HW_DIGCTL_1TRAM_STATUS9_SET 0x8001C1A4 146
HW_DIGCTL_1TRAM_STATUS9_TOG 0x8001C1AC 146
HW_DIGCTL_AHBCYCLES 0x8001C070 136
HW_DIGCTL_AHBSTALLED 0x8001C080 136
HW_DIGCTL_ARMCACHE 0x8001C2B0 151
HW_DIGCTL_CHIPID 0x8001C310 152
HW_DIGCTL_CTRL 0x8001C000 128
HW_DIGCTL_CTRL_CLR 0x8001C008 128
HW_DIGCTL_CTRL_SET 0x8001C004 128
HW_DIGCTL_CTRL_TOG ..... 0x8001C00C.....128
HW_DIGCTL_DBG ..... 0x8001C0D0.....139
HW_DIGCTL_DBGRD ..... 0x8001C0C0.....139
HW_DIGCTL_ENTROPY 0x8001C090 137
HW_DIGCTL_HCLKCOUNT 0x8001C020 131
HW_DIGCTL_HCLKCOUNT_CLR 0x8001C028 131
HW_DIGCTL_HCLKCOUNT_SET 0x8001C024 131
HW_DIGCTL_HCLKCOUNT_TOG ..... 0x8001C02C.....131
HW_DIGCTL_MICROSECONDS ..... 0x8001C0B0.....138
HW_DIGCTL_MICROSECONDS_CLR ..... 0x8001C0B8.....138
HW_DIGCTL_MICROSECONDS_SET ..... 0x8001C0B4.....138
HW_DIGCTL_MICROSECONDS_TOG 0x8001C0BC 138
HW_DIGCTL_RAMCTRL 0x8001C030 131
HW_DIGCTL_RAMCTRL_CLR 0x8001C038 131
HW_DIGCTL_RAMCTRL_SET 0x8001C034 131
HW_DIGCTL_RAMCTRL_TOG ..... 0x8001C03C.....131
HW_DIGCTL_RAMREPAIR0 ..... 0x8001C040 133
HW_DIGCTL_RAMREPAIR0_CLR 0x8001C048 133
HW_DIGCTL_RAMREPAIR0_SET 0x8001C044 133
HW_DIGCTL_RAMREPAIR0_TOG ..... 0x8001C04C.....133
HW_DIGCTL_RAMREPAIR1 ..... 0x8001C050 134
HW_DIGCTL_RAMREPAIR1_CLR 0x8001C058 134
HW_DIGCTL_RAMREPAIR1_SET 0x8001C054 134
HW_DIGCTL_RAMREPAIR1_TOG ..... 0x8001C05C.....134
HW_DIGCTL_ROMSHIELD ..... 0x8001C0A0.....138
HW_DIGCTL_SCRATCH0 ..... 0x8001C290 150
HW_DIGCTL_SCRATCH1 ..... 0x8001C2A0 150
HW_DIGCTL_SGTL 0x8001C300 151
HW_DIGCTL_STATUS 0x8001C010 130
HW_DIGCTL_STATUS_CLR 0x8001C018 130

HW_DIGCTL_STATUS_SET 0x8001C014 130
HW_DIGCTL_STATUS_TOG 0x8001C01C 130
HW_DIGCTL_WRITEONCE 0x8001C060 135
HW_DRI_CTRL 0x80074000 697
HW_DRI_CTRL_CLR 0x80074008 697
HW_DRI_CTRL_SET 0x80074004 697
HW_DRI_CTRL_TOG 0x8007400C 697
HW_DRI_DATA 0x80074030 701
HW_DRI_DEBUG0 ..... 0x80074040 702
HW_DRI_DEBUG0_CLR 0x80074048 702
HW_DRI_DEBUG0_SET 0x80074044 702
HW_DRI_DEBUG0_TOG 0x8007404C 702
HW_DRI_DEBUG1 ..... 0x80074050 703
HW_DRI_DEBUG1_CLR 0x80074058 703
HW_DRI_DEBUG1_SET 0x80074054 703
HW_DRI_DEBUG1_TOG 0x8007405C 703
HW_DRI_STAT 0x80074020 700
HW_DRI_TIMING 0x80074010 699
HW_EMICTRL 0x80020000 329
HW_EMICTRL_CLR 0x80020008 329
HW_EMICTRL_SET 0x80020004 329
HW_EMICTRL_TOG 0x8002000C 329
HW_EMIDEBUG 0x80020020 331
HW_EMIDRAMADDR 0x800200A0 334
HW_EMIDRAMADDR_CLR 0x800200A8 334
HW_EMIDRAMADDR_SET 0x800200A4 334
HW_EMIDRAMADDR_TOG 0x800200AC 334
HW_EMIDRAMCTRL 0x80020090 333
HW_EMIDRAMCTRL_CLR 0x80020098 333
HW_EMIDRAMCTRL_SET 0x80020094 333
HW_EMIDRAMCTRL_TOG 0x8002009C 333
HW_EMIDRAMMODE 0x800200B0 335
HW_EMIDRAMSTAT 0x80020080 332
HW_EMIDRAMTIME 0x800200C0 336
HW_EMIDRAMTIME_CLR 0x800200C8 336
HW_EMIDRAMTIME_SET 0x800200C4 336
HW_EMIDRAMTIME_TOG 0x800200CC 336
HW_EMIDRAMTIME2 ..... 0x800200D0 338
HW_EMIDRAMTIME2_CLR 0x800200D8 338
HW_EMIDRAMTIME2_SET 0x800200D4 338
HW_EMIDRAMTIME2_TOG 0x800200DC 338
HW_EMISTAT 0x80020010 330
HW_EMISTATICCTRL 0x80020100 338
HW_EMISTATICCTRL_CLR 0x80020108 338
HW_EMISTATICCTRL_SET 0x80020104 338
HW_EMISTATICCTRL_TOG 0x8002010C 338
HW_EMISTATICTIME 0x80020110 339
HW_EMISTATICTIME_CLR 0x80020118 339
HW_EMISTATICTIME_SET 0x80020114 339
HW_EMISTATICTIME_TOG 0x8002011C 339
HW_GPMI_COMPARE 0x8000C010 351
HW_GPMI_CTRL0 ..... 0x8000C000 348
HW_GPMI_CTRL0_CLR 0x8000C008 348
HW_GPMI_CTRL0_SET 0x8000C004 348
HW_GPMI_CTRL0_TOG 0x8000C00C 348
HW_GPMI_CTRL1 ..... 0x8000C020 351
HW_GPMI_CTRL1_CLR 0x8000C028 351
HW_GPMI_CTRL1_SET 0x8000C024 351
HW_GPMI_CTRL1_TOG 0x8000C02C 351
HW_GPMI_DATA 0x8000C060 356
HW_GPMI_DEBUG 0x8000C080 357
HW_GPMI_STAT 0x8000C070 356
HW_GPMI_TIMING0 ..... 0x8000C030 353

HW_GPMI_TIMING1 ..... 0x8000C040 354
HW_GPMI_TIMING2 ..... 0x8000C050 355
HW_HWECC_CTRL 0x80008000 373
HW_HWECC_CTRL_CLR 0x80008008 373
HW_HWECC_CTRL_SET 0x80008004 373
HW_HWECC_CTRL_TOG 0x8000800C 373
HW_HWECC_DATA 0x80008090 381
HW_HWECC_DATA_CLR 0x80008098 381
HW_HWECC_DATA_SET 0x80008094 381
HW_HWECC_DATA_TOG 0x8000809C 381
HW_HWECC_DEBUG0 ..... 0x80008020 375
HW_HWECC_DEBUG0_CLR 0x80008028 375
HW_HWECC_DEBUG0_SET 0x80008024 375
HW_HWECC_DEBUG0_TOG 0x8000802C 375
HW_HWECC_DEBUG1 ..... 0x80008030 377
HW_HWECC_DEBUG1_CLR 0x80008038 377
HW_HWECC_DEBUG1_SET 0x80008034 377
HW_HWECC_DEBUG1_TOG 0x8000803C 377
HW_HWECC_DEBUG2 ..... 0x80008040 378
HW_HWECC_DEBUG2_CLR 0x80008048 378
HW_HWECC_DEBUG2_SET 0x80008044 378
HW_HWECC_DEBUG2_TOG 0x8000804C 378
HW_HWECC_DEBUG3 ..... 0x80008050 378
HW_HWECC_DEBUG3_CLR 0x80008058 378
HW_HWECC_DEBUG3_SET 0x80008054 378
HW_HWECC_DEBUG3_TOG 0x8000805C 378
HW_HWECC_DEBUG4 ..... 0x80008060 379
HW_HWECC_DEBUG4_CLR 0x80008068 379
HW_HWECC_DEBUG4_SET 0x80008064 379
HW_HWECC_DEBUG4_TOG 0x8000806C 379
HW_HWECC_DEBUG5 ..... 0x80008070 380
HW_HWECC_DEBUG5_CLR 0x80008078 380
HW_HWECC_DEBUG5_SET 0x80008074 380
HW_HWECC_DEBUG5_TOG 0x8000807C 380
HW_HWECC_DEBUG6 ..... 0x80008080 380
HW_HWECC_DEBUG6_CLR 0x80008088 381
HW_HWECC_DEBUG6_SET 0x80008084 381
HW_HWECC_DEBUG6_TOG 0x8000808C 381
HW_HWECC_STAT 0x80008010 374
HW_HWECC_STAT_CLR 0x80008018 375
HW_HWECC_STAT_SET 0x80008014 374
HW_HWECC_STAT_TOG 0x8000801C 375
HW_I2C_CTRL0 ..... 0x80058000 555
HW_I2C_CTRL0_CLR 0x80058008 555
HW_I2C_CTRL0_SET 0x80058004 555
HW_I2C_CTRL0_TOG 0x8005800C 555
HW_I2C_CTRL1 ..... 0x80058040 560
HW_I2C_CTRL1_CLR 0x80058048 560
HW_I2C_CTRL1_SET 0x80058044 560
HW_I2C_CTRL1_TOG 0x8005804C 560
HW_I2C_DATA 0x80058060 567
HW_I2C_DEBUG0 ..... 0x80058070 567
HW_I2C_DEBUG0_CLR 0x80058078 568
HW_I2C_DEBUG0_SET 0x80058074 568
HW_I2C_DEBUG0_TOG 0x8005807C 568
HW_I2C_DEBUG1 ..... 0x80058080 569
HW_I2C_DEBUG1_CLR 0x80058088 569
HW_I2C_DEBUG1_SET 0x80058084 569
HW_I2C_DEBUG1_TOG 0x8005808C 569
HW_I2C_STAT 0x80058050 563
HW_I2C_TIMING0 ..... 0x80058010 558
HW_I2C_TIMING0_CLR 0x80058018 558
HW_I2C_TIMING0_SET 0x80058014 558

HW_I2C_TIMING0_TOG 0x8005801C 558
HW_I2C_TIMING1 ..... 0x80058020 558
HW_I2C_TIMING1_CLR 0x80058028 558
HW_I2C_TIMING1_SET 0x80058024 558
HW_I2C_TIMING1_TOG 0x8005802C 558
HW_I2C_TIMING2 ..... 0x80058030 559
HW_I2C_TIMING2_CLR 0x80058038 559
HW_I2C_TIMING2_SET 0x80058034 559
HW_I2C_TIMING2_TOG 0x8005803C 559
HW_ICOLL_CTRL 0x80000020 79
HW_ICOLL_CTRL_CLR 0x80000028 79
HW_ICOLL_CTRL_SET 0x80000024 79
HW_ICOLL_CTRL_TOG 0x8000002C 79
HW_ICOLL_DBGFLAG 0x800001A0 113
HW_ICOLL_DBGFLAG_CLR 0x800001A8 113
HW_ICOLL_DBGFLAG_SET 0x800001A4 113
HW_ICOLL_DBGFLAG_TOG 0x800001AC 113
HW_ICOLL_DBGREAD0 ..... 0x80000180 112
HW_ICOLL_DBGREAD0_CLR 0x80000188 112
HW_ICOLL_DBGREAD0_SET 0x80000184 112
HW_ICOLL_DBGREAD0_TOG 0x8000018C 112
HW_ICOLL_DBGREAD1 ..... 0x80000190 112
HW_ICOLL_DBGREAD1_CLR 0x80000198 112
HW_ICOLL_DBGREAD1_SET 0x80000194 112
HW_ICOLL_DBGREAD1_TOG 0x8000019C 112
HW_ICOLL_DBGREQUEST0 ..... 0x800001B0 113
HW_ICOLL_DBGREQUEST0_CLR 0x800001B8 113
HW_ICOLL_DBGREQUEST0_SET 0x800001B4 113
HW_ICOLL_DBGREQUEST0_TOG 0x800001BC 113
HW_ICOLL_DBGREQUEST1 ..... 0x800001C0 114
HW_ICOLL_DBGREQUEST1_CLR 0x800001C8 114
HW_ICOLL_DBGREQUEST1_SET 0x800001C4 114
HW_ICOLL_DBGREQUEST1_TOG 0x800001CC 114
HW_ICOLL_DEBUG 0x80000170 110
HW_ICOLL_DEBUG_CLR 0x80000178 110
HW_ICOLL_DEBUG_SET 0x80000174 110
HW_ICOLL_DEBUG_TOG 0x8000017C 110
HW_ICOLL_LEVELACK 0x80000010 78
HW_ICOLL_PRIORITY0 ..... 0x80000060 83
HW_ICOLL_PRIORITY0_CLR 0x80000068 83
HW_ICOLL_PRIORITY0_SET 0x80000064 83
HW_ICOLL_PRIORITY0_TOG 0x8000006C 83
HW_ICOLL_PRIORITY1 ..... 0x80000070 85
HW_ICOLL_PRIORITY1_CLR 0x80000078 85
HW_ICOLL_PRIORITY1_SET 0x80000074 85
HW_ICOLL_PRIORITY1_TOG 0x8000007C 85
HW_ICOLL_PRIORITY10 ..... 0x80000100 100
HW_ICOLL_PRIORITY10_CLR 0x80000108 100
HW_ICOLL_PRIORITY10_SET 0x80000104 100
HW_ICOLL_PRIORITY10_TOG 0x8000010C 100
HW_ICOLL_PRIORITY11 ..... 0x80000110 101
HW_ICOLL_PRIORITY11_CLR 0x80000118 101
HW_ICOLL_PRIORITY11_SET 0x80000114 101
HW_ICOLL_PRIORITY11_TOG 0x8000011C 101
HW_ICOLL_PRIORITY12 ..... 0x80000120 103
HW_ICOLL_PRIORITY12_CLR 0x80000128 103
HW_ICOLL_PRIORITY12_SET 0x80000124 103
HW_ICOLL_PRIORITY12_TOG 0x8000012C 103
HW_ICOLL_PRIORITY13 ..... 0x80000130 105
HW_ICOLL_PRIORITY13_CLR 0x80000138 105
HW_ICOLL_PRIORITY13_SET 0x80000134 105
HW_ICOLL_PRIORITY13_TOG 0x8000013C 105
HW_ICOLL_PRIORITY14 ..... 0x80000140 106

HW_ICOLL_PRIORITY14_CLR 0x80000148 106
HW_ICOLL_PRIORITY14_SET 0x80000144 106
HW_ICOLL_PRIORITY14_TOG 0x8000014C 106
HW_ICOLL_PRIORITY15 ..... 0x80000150 108
HW_ICOLL_PRIORITY15_CLR 0x80000158 108
HW_ICOLL_PRIORITY15_SET 0x80000154 108
HW_ICOLL_PRIORITY15_TOG 0x8000015C 108
HW_ICOLL_PRIORITY2 ..... 0x80000080 86
HW_ICOLL_PRIORITY2_CLR 0x80000088 86
HW_ICOLL_PRIORITY2_SET 0x80000084 86
HW_ICOLL_PRIORITY2_TOG 0x8000008C 86
HW_ICOLL_PRIORITY3 ..... 0x80000090 88
HW_ICOLL_PRIORITY3_CLR 0x80000098 88
HW_ICOLL_PRIORITY3_SET 0x80000094 88
HW_ICOLL_PRIORITY3_TOG 0x8000009C 88
HW_ICOLL_PRIORITY4 ..... 0x800000A0 90
HW_ICOLL_PRIORITY4_CLR 0x800000A8 90
HW_ICOLL_PRIORITY4_SET 0x800000A4 90
HW_ICOLL_PRIORITY4_TOG 0x800000AC 90
HW_ICOLL_PRIORITY5 ..... 0x800000B0 91
HW_ICOLL_PRIORITY5_CLR 0x800000B8 91
HW_ICOLL_PRIORITY5_SET 0x800000B4 91
HW_ICOLL_PRIORITY5_TOG 0x800000BC 91
HW_ICOLL_PRIORITY6 ..... 0x800000C0 93
HW_ICOLL_PRIORITY6_CLR 0x800000C8 93
HW_ICOLL_PRIORITY6_SET 0x800000C4 93
HW_ICOLL_PRIORITY6_TOG 0x800000CC 93
HW_ICOLL_PRIORITY7 ..... 0x800000D0 95
HW_ICOLL_PRIORITY7_CLR 0x800000D8 95
HW_ICOLL_PRIORITY7_SET 0x800000D4 95
HW_ICOLL_PRIORITY7_TOG 0x800000DC 95
HW_ICOLL_PRIORITY8 ..... 0x800000E0 96
HW_ICOLL_PRIORITY8_CLR 0x800000E8 96
HW_ICOLL_PRIORITY8_SET 0x800000E4 96
HW_ICOLL_PRIORITY8_TOG 0x800000EC 96
HW_ICOLL_PRIORITY9 ..... 0x800000F0 98
HW_ICOLL_PRIORITY9_CLR 0x800000F8 98
HW_ICOLL_PRIORITY9_SET 0x800000F4 98
HW_ICOLL_PRIORITY9_TOG 0x800000FC 98
HW_ICOLL_RAW0 ..... 0x80000040 82
HW_ICOLL_RAW0_CLR 0x80000048 82
HW_ICOLL_RAW0_SET 0x80000044 82
HW_ICOLL_RAW0_TOG 0x8000004C 82
HW_ICOLL_RAW1 ..... 0x80000050 83
HW_ICOLL_RAW1_CLR 0x80000058 83
HW_ICOLL_RAW1_SET 0x80000054 83
HW_ICOLL_RAW1_TOG 0x8000005C 83
HW_ICOLL_STAT 0x80000030 81
HW_ICOLL_VBASE 0x80000160 110
HW_ICOLL_VBASE_CLR 0x80000168 110
HW_ICOLL_VBASE_SET 0x80000164 110
HW_ICOLL_VBASE_TOG 0x8000016C 110
HW_ICOLL_VECTOR 0x80000000 78
HW_ICOLL_VECTOR_CLR 0x80000008 78
HW_ICOLL_VECTOR_SET 0x80000004 78
HW_ICOLL_VECTOR_TOG 0x8000000C 78
HW_IR_CTRL 0x80078000 611
HW_IR_CTRL_CLR 0x80078008 611
HW_IR_CTRL_SET 0x80078004 611
HW_IR_CTRL_TOG 0x8007800C 611
HW_IR_DATA 0x80078050 617
HW_IR_DBGCTRL 0x80078030 614
HW_IR_DBGCTRL_CLR 0x80078038 614

HW_IR_DBGCTRL_SET 0x80078034 614
HW_IR_DBGCTRL_TOG 0x8007803C 614
HW_IR_DEBUG 0x80078090 621
HW_IR_INTR 0x80078040 615
HW_IR_INTR_CLR 0x80078048 615
HW_IR_INTR_SET 0x80078044 615
HW_IR_INTR_TOG 0x8007804C 615
HW_IR_RXDMA 0x80078020 613
HW_IR_RXDMA_CLR 0x80078028 613
HW_IR_RXDMA_SET 0x80078024 613
HW_IR_RXDMA_TOG 0x8007802C 613
HW_IR_SI_READ 0x80078080 620
HW_IR_STAT 0x80078060 618
HW_IR_TCCTRL 0x80078070 619
HW_IR_TCCTRL_CLR 0x80078078 619
HW_IR_TCCTRL_SET 0x80078074 619
HW_IR_TCCTRL_TOG 0x8007807C 619
HW_IR_TXDMA 0x80078010 612
HW_IR_TXDMA_CLR 0x80078018 612
HW_IR_TXDMA_SET 0x80078014 612
HW_IR_TXDMA_TOG 0x8007801C 612
HW_LCDIF_CTRL 0x80060000 423
HW_LCDIF_CTRL_CLR 0x80060008 423
HW_LCDIF_CTRL_SET 0x80060004 423
HW_LCDIF_CTRL_TOG 0x8006000C 423
HW_LCDIF_DATA 0x80060020 425
HW_LCDIF_DEBUG 0x80060030 426
HW_LCDIF_TIMING 0x80060010 425
HW_LRADC_CH0 ..... 0x80050050 721
HW_LRADC_CH0_CLR 0x80050058 721
HW_LRADC_CH0_SET 0x80050054 721
HW_LRADC_CH0_TOG 0x8005005C 721
HW_LRADC_CH1 ..... 0x80050060 722
HW_LRADC_CH1_CLR 0x80050068 722
HW_LRADC_CH1_SET 0x80050064 722
HW_LRADC_CH1_TOG 0x8005006C 722
HW_LRADC_CH2 ..... 0x80050070 723
HW_LRADC_CH2_CLR 0x80050078 723
HW_LRADC_CH2_SET 0x80050074 723
HW_LRADC_CH2_TOG 0x8005007C 723
HW_LRADC_CH3 ..... 0x80050080 724
HW_LRADC_CH3_CLR 0x80050088 724
HW_LRADC_CH3_SET 0x80050084 724
HW_LRADC_CH3_TOG 0x8005008C 724
HW_LRADC_CH4 ..... 0x80050090 726
HW_LRADC_CH4_CLR 0x80050098 726
HW_LRADC_CH4_SET 0x80050094 726
HW_LRADC_CH4_TOG 0x8005009C 726
HW_LRADC_CH5 ..... 0x800500A0 727
HW_LRADC_CH5_CLR 0x800500A8 727
HW_LRADC_CH5_SET 0x800500A4 727
HW_LRADC_CH5_TOG 0x800500AC 727
HW_LRADC_CH6 ..... 0x800500B0 728
HW_LRADC_CH6_CLR 0x800500B8 728
HW_LRADC_CH6_SET 0x800500B4 728
HW_LRADC_CH6_TOG 0x800500BC 728
HW_LRADC_CH7 ..... 0x800500C0 729
HW_LRADC_CH7_CLR 0x800500C8 729
HW_LRADC_CH7_SET 0x800500C4 729
HW_LRADC_CH7_TOG 0x800500CC 729
HW_LRADC_CONVERSION 0x80050130 739
HW_LRADC_CONVERSION_CLR 0x80050138 739
HW_LRADC_CONVERSION_SET 0x80050134 739

HW_LRADC_CONVERSION_TOG 0x8005013C 739
HW_LRADC_CTRL0 ..... 0x80050000 710
HW_LRADC_CTRL0_CLR 0x80050008 710
HW_LRADC_CTRL0_SET 0x80050004 710
HW_LRADC_CTRL0_TOG 0x8005000C 710
HW_LRADC_CTRL1 ..... 0x80050010 711
HW_LRADC_CTRL1_CLR 0x80050018 711
HW_LRADC_CTRL1_SET 0x80050014 711
HW_LRADC_CTRL1_TOG 0x8005001C 711
HW_LRADC_CTRL2 ..... 0x80050020 714
HW_LRADC_CTRL2_CLR 0x80050028 714
HW_LRADC_CTRL2_SET 0x80050024 714
HW_LRADC_CTRL2_TOG 0x8005002C 714
HW_LRADC_CTRL3 ..... 0x80050030 717
HW_LRADC_CTRL3_CLR 0x80050038 717
HW_LRADC_CTRL3_SET 0x80050034 717
HW_LRADC_CTRL3_TOG 0x8005003C 717
HW_LRADC_DEBUG0 ..... 0x80050110 737
HW_LRADC_DEBUG0_CLR 0x80050118 737
HW_LRADC_DEBUG0_SET 0x80050114 737
HW_LRADC_DEBUG0_TOG 0x8005011C 737
HW_LRADC_DEBUG1 ..... 0x80050120 738
HW_LRADC_DEBUG1_CLR 0x80050128 738
HW_LRADC_DEBUG1_SET 0x80050124 738
HW_LRADC_DEBUG1_TOG 0x8005012C 738
HW_LRADC_DELAY0 ..... 0x800500D0 731
HW_LRADC_DELAY0_CLR 0x800500D8 731
HW_LRADC_DELAY0_SET 0x800500D4 731
HW_LRADC_DELAY0_TOG 0x800500DC 731
HW_LRADC_DELAY1 ..... 0x800500E0 732
HW_LRADC_DELAY1_CLR 0x800500E8 732
HW_LRADC_DELAY1_SET 0x800500E4 732
HW_LRADC_DELAY1_TOG 0x800500EC 732
HW_LRADC_DELAY2 ..... 0x800500F0 734
HW_LRADC_DELAY2_CLR 0x800500F8 734
HW_LRADC_DELAY2_SET 0x800500F4 734
HW_LRADC_DELAY2_TOG 0x800500FC 734
HW_LRADC_DELAY3 ..... 0x80050100 735
HW_LRADC_DELAY3_CLR 0x80050108 735
HW_LRADC_DELAY3_SET 0x80050104 735
HW_LRADC_DELAY3_TOG 0x8005010C 735
HW_LRADC_STATUS 0x80050040 719
HW_LRADC_STATUS_CLR 0x80050048 719
HW_LRADC_STATUS_SET 0x80050044 719
HW_LRADC_STATUS_TOG 0x8005004C 719
HW_MEMCPY_CTRL 0x80014000 744
HW_MEMCPY_CTRL_CLR 0x80014008 744
HW_MEMCPY_CTRL_SET 0x80014004 744
HW_MEMCPY_CTRL_TOG 0x8001400C 744
HW_MEMCPY_DATA 0x80014010 745
HW_MEMCPY_DATA_CLR 0x80014018 745
HW_MEMCPY_DATA_SET 0x80014014 745
HW_MEMCPY_DATA_TOG 0x8001401C 745
HW_MEMCPY_DEBUG 0x80014020 746
HW_MEMCPY_DEBUG_CLR 0x80014028 746
HW_MEMCPY_DEBUG_SET 0x80014024 746
HW_MEMCPY_DEBUG_TOG 0x8001402C 746
HW_PINCTRL_CTRL 0x80018000 439
HW_PINCTRL_CTRL_CLR 0x80018008 439
HW_PINCTRL_CTRL_SET 0x80018004 439
HW_PINCTRL_CTRL_TOG 0x8001800C 439
HW_PINCTRL_DIN0 ..... 0x80018060 443
HW_PINCTRL_DIN0_CLR 0x80018068 443

HW_PINCTRL_DIN0_SET 0x80018064 443
HW_PINCTRL_DIN0_TOG 0x8001806C 443
HW_PINCTRL_DIN1 ..... 0x80018160 451
HW_PINCTRL_DIN1_CLR 0x80018168 451
HW_PINCTRL_DIN1_SET 0x80018164 451
HW_PINCTRL_DIN1_TOG 0x8001816C 451
HW_PINCTRL_DIN2 ..... 0x80018260 459
HW_PINCTRL_DIN2_CLR 0x80018268 459
HW_PINCTRL_DIN2_SET 0x80018264 459
HW_PINCTRL_DIN2_TOG 0x8001826C 459
HW_PINCTRL_DIN3 ..... 0x80018360 468
HW_PINCTRL_DIN3_CLR 0x80018368 468
HW_PINCTRL_DIN3_SET 0x80018364 468
HW_PINCTRL_DIN3_TOG 0x8001836C 468
HW_PINCTRL_DOE0 ..... 0x80018070 443
HW_PINCTRL_DOE0_CLR 0x80018078 443
HW_PINCTRL_DOE0_SET 0x80018074 443
HW_PINCTRL_DOE0_TOG 0x8001807C 443
HW_PINCTRL_DOE1 ..... 0x80018170 452
HW_PINCTRL_DOE1_CLR 0x80018178 452
HW_PINCTRL_DOE1_SET 0x80018174 452
HW_PINCTRL_DOE1_TOG 0x8001817C 452
HW_PINCTRL_DOE2 ..... 0x80018270 460
HW_PINCTRL_DOE2_CLR 0x80018278 460
HW_PINCTRL_DOE2_SET 0x80018274 460
HW_PINCTRL_DOE2_TOG 0x8001827C 460
HW_PINCTRL_DOE3 ..... 0x80018370 469
HW_PINCTRL_DOE3_CLR 0x80018378 469
HW_PINCTRL_DOE3_SET 0x80018374 469
HW_PINCTRL_DOE3_TOG 0x8001837C 469
HW_PINCTRL_DOUT0 ..... 0x80018050 442
HW_PINCTRL_DOUT0_CLR 0x80018058 442
HW_PINCTRL_DOUT0_SET 0x80018054 442
HW_PINCTRL_DOUT0_TOG 0x8001805C 442
HW_PINCTRL_DOUT1 ..... 0x80018150 450
HW_PINCTRL_DOUT1_CLR 0x80018158 450
HW_PINCTRL_DOUT1_SET 0x80018154 450
HW_PINCTRL_DOUT1_TOG 0x8001815C 450
HW_PINCTRL_DOUT2 ..... 0x80018250 459
HW_PINCTRL_DOUT2_CLR 0x80018258 459
HW_PINCTRL_DOUT2_SET 0x80018254 459
HW_PINCTRL_DOUT2_TOG 0x8001825C 459
HW_PINCTRL_DOUT3 ..... 0x80018350 467
HW_PINCTRL_DOUT3_CLR 0x80018358 467
HW_PINCTRL_DOUT3_SET 0x80018354 467
HW_PINCTRL_DOUT3_TOG 0x8001835C 467
HW_PINCTRL_DRIVE0 ..... 0x80018030 441
HW_PINCTRL_DRIVE0_CLR 0x80018038 441
HW_PINCTRL_DRIVE0_SET 0x80018034 441
HW_PINCTRL_DRIVE0_TOG 0x8001803C 442
HW_PINCTRL_DRIVE1 ..... 0x80018130 449
HW_PINCTRL_DRIVE1_CLR 0x80018138 449
HW_PINCTRL_DRIVE1_SET 0x80018134 449
HW_PINCTRL_DRIVE1_TOG 0x8001813C 449
HW_PINCTRL_DRIVE2 ..... 0x80018230 458
HW_PINCTRL_DRIVE2_CLR 0x80018238 458
HW_PINCTRL_DRIVE2_SET 0x80018234 458
HW_PINCTRL_DRIVE2_TOG 0x8001823C 458
HW_PINCTRL_DRIVE3 ..... 0x80018330 466
HW_PINCTRL_DRIVE3_CLR 0x80018338 466
HW_PINCTRL_DRIVE3_SET 0x80018334 466
HW_PINCTRL_DRIVE3_TOG 0x8001833C 466
HW_PINCTRL_IRQEN0 ..... 0x80018090 445

HW_PINCTRL_IRQEN0_CLR 0x80018098 445
HW_PINCTRL_IRQEN0_SET 0x80018094 445
HW_PINCTRL_IRQEN0_TOG 0x8001809C 445
HW_PINCTRL_IRQEN1 ..... 0x80018190 453
HW_PINCTRL_IRQEN1_CLR 0x80018198 453
HW_PINCTRL_IRQEN1_SET 0x80018194 453
HW_PINCTRL_IRQEN1_TOG 0x8001819C 453
HW_PINCTRL_IRQEN2 ..... 0x80018290 461
HW_PINCTRL_IRQEN2_CLR 0x80018298 461
HW_PINCTRL_IRQEN2_SET 0x80018294 461
HW_PINCTRL_IRQEN2_TOG 0x8001829C 461
HW_PINCTRL_IRQEN3 ..... 0x80018390 470
HW_PINCTRL_IRQEN3_CLR 0x80018398 470
HW_PINCTRL_IRQEN3_SET 0x80018394 470
HW_PINCTRL_IRQEN3_TOG 0x8001839C 470
HW_PINCTRL_IRQLEVEL0 ..... 0x800180A0 446
HW_PINCTRL_IRQLEVEL0_CLR 0x800180A8 446
HW_PINCTRL_IRQLEVEL0_SET 0x800180A4 446
HW_PINCTRL_IRQLEVEL0_TOG 0x800180AC 446
HW_PINCTRL_IRQLEVEL1 ..... 0x800181A0 454
HW_PINCTRL_IRQLEVEL1_CLR 0x800181A8 454
HW_PINCTRL_IRQLEVEL1_SET 0x800181A4 454
HW_PINCTRL_IRQLEVEL1_TOG 0x800181AC 454
HW_PINCTRL_IRQLEVEL2 ..... 0x800182A0 462
HW_PINCTRL_IRQLEVEL2_CLR 0x800182A8 462
HW_PINCTRL_IRQLEVEL2_SET 0x800182A4 462
HW_PINCTRL_IRQLEVEL2_TOG 0x800182AC 462
HW_PINCTRL_IRQLEVEL3 ..... 0x800183A0 471
HW_PINCTRL_IRQLEVEL3_CLR 0x800183A8 471
HW_PINCTRL_IRQLEVEL3_SET 0x800183A4 471
HW_PINCTRL_IRQLEVEL3_TOG 0x800183AC 471
HW_PINCTRL_IRQPOL0 ..... 0x800180B0 446
HW_PINCTRL_IRQPOL0_CLR 0x800180B8 446
HW_PINCTRL_IRQPOL0_SET 0x800180B4 446
HW_PINCTRL_IRQPOL0_TOG 0x800180BC 446
HW_PINCTRL_IRQPOL1 ..... 0x800181B0 455
HW_PINCTRL_IRQPOL1_CLR 0x800181B8 455
HW_PINCTRL_IRQPOL1_SET 0x800181B4 455
HW_PINCTRL_IRQPOL1_TOG 0x800181BC 455
HW_PINCTRL_IRQPOL2 ..... 0x800182B0 463
HW_PINCTRL_IRQPOL2_CLR 0x800182B8 463
HW_PINCTRL_IRQPOL2_SET 0x800182B4 463
HW_PINCTRL_IRQPOL2_TOG 0x800182BC 463
HW_PINCTRL_IRQPOL3 ..... 0x800183B0 472
HW_PINCTRL_IRQPOL3_CLR 0x800183B8 472
HW_PINCTRL_IRQPOL3_SET 0x800183B4 472
HW_PINCTRL_IRQPOL3_TOG 0x800183BC 472
HW_PINCTRL_IRQSTAT0 ..... 0x800180C0 447
HW_PINCTRL_IRQSTAT0_CLR 0x800180C8 447
HW_PINCTRL_IRQSTAT0_SET 0x800180C4 447
HW_PINCTRL_IRQSTAT0_TOG 0x800180CC 447
HW_PINCTRL_IRQSTAT1 ..... 0x800181C0 456
HW_PINCTRL_IRQSTAT1_CLR 0x800181C8 456
HW_PINCTRL_IRQSTAT1_SET 0x800181C4 456
HW_PINCTRL_IRQSTAT1_TOG 0x800181CC 456
HW_PINCTRL_IRQSTAT2 ..... 0x800182C0 463
HW_PINCTRL_IRQSTAT2_CLR 0x800182C8 463
HW_PINCTRL_IRQSTAT2_SET 0x800182C4 463
HW_PINCTRL_IRQSTAT2_TOG 0x800182CC 463
HW_PINCTRL_IRQSTAT3 ..... 0x800183C0 473
HW_PINCTRL_IRQSTAT3_CLR 0x800183C8 473
HW_PINCTRL_IRQSTAT3_SET 0x800183C4 473
HW_PINCTRL_IRQSTAT3_TOG 0x800183CC 473

HW_PINCTRL_MUXSEL0 ..... 0x80018010 440
HW_PINCTRL_MUXSEL0_CLR 0x80018018 440
HW_PINCTRL_MUXSEL0_SET 0x80018014 440
HW_PINCTRL_MUXSEL0_TOG 0x8001801C 440
HW_PINCTRL_MUXSEL1 ..... 0x80018020 441
HW_PINCTRL_MUXSEL1_CLR 0x80018028 441
HW_PINCTRL_MUXSEL1_SET 0x80018024 441
HW_PINCTRL_MUXSEL1_TOG 0x8001802C 441
HW_PINCTRL_MUXSEL2 ..... 0x80018110 448
HW_PINCTRL_MUXSEL2_CLR 0x80018118 448
HW_PINCTRL_MUXSEL2_SET 0x80018114 448
HW_PINCTRL_MUXSEL2_TOG 0x8001811C 448
HW_PINCTRL_MUXSEL3 ..... 0x80018120 448
HW_PINCTRL_MUXSEL3_CLR 0x80018128 449
HW_PINCTRL_MUXSEL3_SET 0x80018124 449
HW_PINCTRL_MUXSEL3_TOG 0x8001812C 449
HW_PINCTRL_MUXSEL4 ..... 0x80018210 456
HW_PINCTRL_MUXSEL4_CLR 0x80018218 456
HW_PINCTRL_MUXSEL4_SET 0x80018214 456
HW_PINCTRL_MUXSEL4_TOG 0x8001821C 457
HW_PINCTRL_MUXSEL5 ..... 0x80018220 457
HW_PINCTRL_MUXSEL5_CLR 0x80018228 457
HW_PINCTRL_MUXSEL5_SET 0x80018224 457
HW_PINCTRL_MUXSEL5_TOG 0x8001822C 457
HW_PINCTRL_MUXSEL6 ..... 0x80018310 464
HW_PINCTRL_MUXSEL6_CLR 0x80018318 464
HW_PINCTRL_MUXSEL6_SET 0x80018314 464
HW_PINCTRL_MUXSEL6_TOG 0x8001831C 464
HW_PINCTRL_MUXSEL7 ..... 0x80018320 465
HW_PINCTRL_MUXSEL7_CLR 0x80018328 465
HW_PINCTRL_MUXSEL7_SET 0x80018324 465
HW_PINCTRL_MUXSEL7_TOG 0x8001832C 465
HW_PINCTRL_PIN2IRQ0 ..... 0x80018080 444
HW_PINCTRL_PIN2IRQ0_CLR 0x80018088 444
HW_PINCTRL_PIN2IRQ0_SET 0x80018084 444
HW_PINCTRL_PIN2IRQ0_TOG 0x8001808C 444
HW_PINCTRL_PIN2IRQ1 ..... 0x80018180 452
HW_PINCTRL_PIN2IRQ1_CLR 0x80018188 453
HW_PINCTRL_PIN2IRQ1_SET 0x80018184 452
HW_PINCTRL_PIN2IRQ1_TOG 0x8001818C 453
HW_PINCTRL_PIN2IRQ2 ..... 0x80018280 461
HW_PINCTRL_PIN2IRQ2_CLR 0x80018288 461
HW_PINCTRL_PIN2IRQ2_SET 0x80018284 461
HW_PINCTRL_PIN2IRQ2_TOG 0x8001828C 461
HW_PINCTRL_PIN2IRQ3 ..... 0x80018380 469
HW_PINCTRL_PIN2IRQ3_CLR 0x80018388 469
HW_PINCTRL_PIN2IRQ3_SET 0x80018384 469
HW_PINCTRL_PIN2IRQ3_TOG 0x8001838C 469
HW_POWER_5VCTRL 0x80044010 760
HW_POWER_5VCTRL_CLR 0x80044018 760
HW_POWER_5VCTRL_SET 0x80044014 760
HW_POWER_5VCTRL_TOG 0x8004401C 760
HW_POWER_BATTCHRG 0x80044030 765
HW_POWER_BATTCHRG_CLR 0x80044038 765
HW_POWER_BATTCHRG_SET 0x80044034 765
HW_POWER_BATTCHRG_TOG 0x8004403C 765
HW_POWER_BATTMONITOR 0x800440b0 777
HW_POWER_CTRL 0x80044000 759
HW_POWER_CTRL_CLR 0x80044008 759
HW_POWER_CTRL_SET 0x80044004 759
HW_POWER_CTRL_TOG 0x8004400C 759
HW_POWER_DC1LIMITS 0x80044060 770
HW_POWER_DC1MULTOUT 0x80044050 768

HW_POWER_DC2LIMITS 0x80044070 770
HW_POWER_DEBUG 0x800440d0 779
HW_POWER_DEBUG_CLR 0x800440d8 779
HW_POWER_DEBUG_SET 0x800440d4 779
HW_POWER_DEBUG_TOG 0x800440dC 779
HW_POWER_LOOPCTRL 0x80044080 771
HW_POWER_LOOPCTRL_CLR 0x80044088 771
HW_POWER_LOOPCTRL_SET 0x80044084 771
HW_POWER_LOOPCTRL_TOG 0x8004408C 771
HW_POWER_MINPWR 0x80044020 763
HW_POWER_MINPWR_CLR 0x80044028 763
HW_POWER_MINPWR_SET 0x80044024 763
HW_POWER_MINPWR_TOG 0x8004402C 763
HW_POWER_RESET 0x800440c0 778
HW_POWER_RESET_CLR 0x800440C8 778
HW_POWER_RESET_SET 0x800440C4 778
HW_POWER_RESET_TOG 0x800440CC 778
HW_POWER_SPEEDTEMP 0x800440a0 775
HW_POWER_SPEEDTEMP_CLR 0x800440a8 775
HW_POWER_SPEEDTEMP_SET 0x800440a4 775
HW_POWER_SPEEDTEMP_TOG 0x800440aC 775
HW_POWER_STS 0x80044090 773
HW_POWER_VDDCTRL 0x80044040 766
HW_PWM_ACTIVE0 ..... 0x80064010 528
HW_PWM_ACTIVE0_CLR 0x80064018 528
HW_PWM_ACTIVE0_SET 0x80064014 528
HW_PWM_ACTIVE0_TOG 0x8006401C 528
HW_PWM_ACTIVE1 ..... 0x80064030 530
HW_PWM_ACTIVE1_CLR 0x80064038 530
HW_PWM_ACTIVE1_SET 0x80064034 530
HW_PWM_ACTIVE1_TOG 0x8006403C 530
HW_PWM_ACTIVE2 ..... 0x80064050 532
HW_PWM_ACTIVE2_CLR 0x80064058 532
HW_PWM_ACTIVE2_SET 0x80064054 532
HW_PWM_ACTIVE2_TOG 0x8006405C 532
HW_PWM_ACTIVE3 ..... 0x80064070 534
HW_PWM_ACTIVE3_CLR 0x80064078 534
HW_PWM_ACTIVE3_SET 0x80064074 534
HW_PWM_ACTIVE3_TOG 0x8006407C 534
HW_PWM_ACTIVE4 ..... 0x80064090 536
HW_PWM_ACTIVE4_CLR 0x80064098 536
HW_PWM_ACTIVE4_SET 0x80064094 536
HW_PWM_ACTIVE4_TOG 0x8006409C 536
HW_PWM_CTRL 0x80064000 527
HW_PWM_CTRL_CLR 0x80064008 527
HW_PWM_CTRL_SET 0x80064004 527
HW_PWM_CTRL_TOG 0x8006400C 527
HW_PWM_PERIOD0 ..... 0x80064020 529
HW_PWM_PERIOD0_CLR 0x80064028 529
HW_PWM_PERIOD0_SET 0x80064024 529
HW_PWM_PERIOD0_TOG 0x8006402C 529
HW_PWM_PERIOD1 ..... 0x80064040 531
HW_PWM_PERIOD1_CLR 0x80064048 531
HW_PWM_PERIOD1_SET 0x80064044 531
HW_PWM_PERIOD1_TOG 0x8006404C 531
HW_PWM_PERIOD2 ..... 0x80064060 533
HW_PWM_PERIOD2_CLR 0x80064068 533
HW_PWM_PERIOD2_SET 0x80064064 533
HW_PWM_PERIOD2_TOG 0x8006406C 533
HW_PWM_PERIOD3 ..... 0x80064080 535
HW_PWM_PERIOD3_CLR 0x80064088 535
HW_PWM_PERIOD3_SET 0x80064084 535
HW_PWM_PERIOD3_TOG 0x8006408C 535

HW_PWM_PERIOD4 ..... 0x800640A0 537
HW_PWM_PERIOD4_CLR 0x800640A8 537
HW_PWM_PERIOD4_SET 0x800640A4 537
HW_PWM_PERIOD4_TOG 0x800640AC 537
HW_RTC_ALARM 0x8005C040 508
HW_RTC_ALARM_CLR 0x8005C048 508
HW_RTC_ALARM_SET 0x8005C044 508
HW_RTC_ALARM_TOG 0x8005C04C 508
HW_RTC_CTRL 0x8005C000 503
HW_RTC_CTRL_CLR 0x8005C008 503
HW_RTC_CTRL_SET 0x8005C004 503
HW_RTC_CTRL_TOG 0x8005C00C 503
HW_RTC_DEBUG 0x8005C0A0 513
HW_RTC_DEBUG_CLR 0x8005C0A8 513
HW_RTC_DEBUG_SET 0x8005C0A4 513
HW_RTC_DEBUG_TOG 0x8005C0AC 513
HW_RTC_LASERFUSE0 ..... 0x8005C300 515
HW_RTC_LASERFUSE0_CLR 0x8005C308 515
HW_RTC_LASERFUSE0_SET 0x8005C304 515
HW_RTC_LASERFUSE0_TOG 0x8005C30C 515
HW_RTC_LASERFUSE1 ..... 0x8005C310 516
HW_RTC_LASERFUSE1_CLR 0x8005C318 516
HW_RTC_LASERFUSE1_SET 0x8005C314 516
HW_RTC_LASERFUSE1_TOG 0x8005C31C 516
HW_RTC_LASERFUSE10 ..... 0x8005C3A0 521
HW_RTC_LASERFUSE10_CLR 0x8005C3A8 521
HW_RTC_LASERFUSE10_SET 0x8005C3A4 521
HW_RTC_LASERFUSE10_TOG 0x8005C3AC 521
HW_RTC_LASERFUSE11 ..... 0x8005C3B0 521
HW_RTC_LASERFUSE11_CLR 0x8005C3B8 521
HW_RTC_LASERFUSE11_SET 0x8005C3B4 521
HW_RTC_LASERFUSE11_TOG 0x8005C3BC 522
HW_RTC_LASERFUSE2 ..... 0x8005C320 516
HW_RTC_LASERFUSE2_CLR 0x8005C328 516
HW_RTC_LASERFUSE2_SET 0x8005C324 516
HW_RTC_LASERFUSE2_TOG 0x8005C32C 516
HW_RTC_LASERFUSE3 ..... 0x8005C330 517
HW_RTC_LASERFUSE3_CLR 0x8005C338 517
HW_RTC_LASERFUSE3_SET 0x8005C334 517
HW_RTC_LASERFUSE3_TOG 0x8005C33C 517
HW_RTC_LASERFUSE4 ..... 0x8005C340 517
HW_RTC_LASERFUSE4_CLR 0x8005C348 517
HW_RTC_LASERFUSE4_SET 0x8005C344 517
HW_RTC_LASERFUSE4_TOG 0x8005C34C 517
HW_RTC_LASERFUSE5 ..... 0x8005C350 518
HW_RTC_LASERFUSE5_CLR 0x8005C358 518
HW_RTC_LASERFUSE5_SET 0x8005C354 518
HW_RTC_LASERFUSE5_TOG 0x8005C35C 518
HW_RTC_LASERFUSE6 ..... 0x8005C360 518
HW_RTC_LASERFUSE6_CLR 0x8005C368 518
HW_RTC_LASERFUSE6_SET 0x8005C364 518
HW_RTC_LASERFUSE6_TOG 0x8005C36C 518
HW_RTC_LASERFUSE7 ..... 0x8005C370 519
HW_RTC_LASERFUSE7_CLR 0x8005C378 519
HW_RTC_LASERFUSE7_SET 0x8005C374 519
HW_RTC_LASERFUSE7_TOG 0x8005C37C 519
HW_RTC_LASERFUSE8 ..... 0x8005C380 520
HW_RTC_LASERFUSE8_CLR 0x8005C388 520
HW_RTC_LASERFUSE8_SET 0x8005C384 520
HW_RTC_LASERFUSE8_TOG 0x8005C38C 520
HW_RTC_LASERFUSE9 ..... 0x8005C390 520
HW_RTC_LASERFUSE9_CLR 0x8005C398 520
HW_RTC_LASERFUSE9_SET 0x8005C394 520

HW_RTC_LASERFUSE9_TOG 0x8005C39C 520
HW_RTC_MILLISECONDS 0x8005C020 506
HW_RTC_MILLISECONDS_CLR 0x8005C028 506
HW_RTC_MILLISECONDS_SET 0x8005C024 506
HW_RTC_MILLISECONDS_TOG 0x8005C02C 507
HW_RTC_PERSISTENT0 ..... 0x8005C060 509
HW_RTC_PERSISTENT0_CLR 0x8005C068 509
HW_RTC_PERSISTENT0_SET 0x8005C064 509
HW_RTC_PERSISTENT0_TOG 0x8005C06C 509
HW_RTC_PERSISTENT1 ..... 0x8005C070 511
HW_RTC_PERSISTENT1_CLR 0x8005C078 511
HW_RTC_PERSISTENT1_SET 0x8005C074 511
HW_RTC_PERSISTENT1_TOG 0x8005C07C 511
HW_RTC_PERSISTENT2 ..... 0x8005C080 512
HW_RTC_PERSISTENT2_CLR 0x8005C088 512
HW_RTC_PERSISTENT2_SET 0x8005C084 512
HW_RTC_PERSISTENT2_TOG 0x8005C08C 512
HW_RTC_PERSISTENT3 ..... 0x8005C090 513
HW_RTC_PERSISTENT3_CLR 0x8005C098 513
HW_RTC_PERSISTENT3_SET 0x8005C094 513
HW_RTC_PERSISTENT3_TOG 0x8005C09C 513
HW_RTC_SECONDS 0x8005C030 507
HW_RTC_SECONDS_CLR 0x8005C038 507
HW_RTC_SECONDS_SET 0x8005C034 507
HW_RTC_SECONDS_TOG 0x8005C03C 507
HW_RTC_STAT 0x8005C010 505
HW_RTC_STAT_CLR 0x8005C018 505
HW_RTC_STAT_SET 0x8005C014 505
HW_RTC_STAT_TOG 0x8005C01C 505
HW_RTC_UNLOCK 0x8005C200 514
HW_RTC_UNLOCK_CLR 0x8005C208 514
HW_RTC_UNLOCK_SET 0x8005C204 514
HW_RTC_UNLOCK_TOG 0x8005C20C 514
HW_RTC_WATCHDOG 0x8005C050 508
HW_RTC_WATCHDOG_CLR 0x8005C058 508
HW_RTC_WATCHDOG_SET 0x8005C054 508
HW_RTC_WATCHDOG_TOG 0x8005C05C 508
HW_SPDIF_CTRL 0x80054000 685
HW_SPDIF_CTRL_CLR 0x80054008 685
HW_SPDIF_CTRL_SET 0x80054004 685
HW_SPDIF_CTRL_TOG 0x8005400C 685
HW_SPDIF_DATA 0x80054050 690
HW_SPDIF_DATA_CLR 0x80054058 690
HW_SPDIF_DATA_SET 0x80054054 690
HW_SPDIF_DATA_TOG 0x8005405C 690
HW_SPDIF_DEBUG 0x80054040 689
HW_SPDIF_DEBUG_CLR 0x80054048 689
HW_SPDIF_DEBUG_SET 0x80054044 689
HW_SPDIF_DEBUG_TOG 0x8005404C 689
HW_SPDIF_FRAMECTRL 0x80054020 687
HW_SPDIF_FRAMECTRL_CLR 0x80054028 687
HW_SPDIF_FRAMECTRL_SET 0x80054024 687
HW_SPDIF_FRAMECTRL_TOG 0x8005402C 687
HW_SPDIF_SRR 0x80054030 688
HW_SPDIF_SRR_CLR 0x80054038 688
HW_SPDIF_SRR_SET 0x80054034 688
HW_SPDIF_SRR_TOG 0x8005403C 689
HW_SPDIF_STAT 0x80054010 687
HW_SPDIF_STAT_CLR 0x80054018 687
HW_SPDIF_STAT_SET 0x80054014 687
HW_SPDIF_STAT_TOG 0x8005401C 687
HW_SSP_CMD0 ..... 0x80010010 405
HW_SSP_CMD0_CLR 0x80010018 405

HW_SSP_CMD0_SET 0x80010014 405
HW_SSP_CMD0_TOG 0x8001001C 405
HW_SSP_CMD1 ..... 0x80010020 407
HW_SSP_COMPMASK 0x80010040 408
HW_SSP_COMPREF 0x80010030 407
HW_SSP_CTRL0 ..... 0x80010000 403
HW_SSP_CTRL0_CLR 0x80010008 403
HW_SSP_CTRL0_SET 0x80010004 403
HW_SSP_CTRL0_TOG 0x8001000C 403
HW_SSP_CTRL1 ..... 0x80010060 409
HW_SSP_CTRL1_CLR 0x80010068 409
HW_SSP_CTRL1_SET 0x80010064 409
HW_SSP_CTRL1_TOG 0x8001006C 409
HW_SSP_DATA 0x80010070 412
HW_SSP_DEBUG 0x80010100 416
HW_SSP_SDRESP0 ..... 0x80010080 413
HW_SSP_SDRESP1 ..... 0x80010090 413
HW_SSP_SDRESP2 ..... 0x800100A0 414
HW_SSP_SDRESP3 ..... 0x800100B0 414
HW_SSP_STATUS 0x800100C0 414
HW_SSP_TIMING 0x80010050 408
HW_TIMROT_ROTCOUNT 0x80068010 485
HW_TIMROT_ROTCTRL 0x80068000 483
HW_TIMROT_ROTCTRL_CLR 0x80068008 483
HW_TIMROT_ROTCTRL_SET 0x80068004 483
HW_TIMROT_ROTCTRL_TOG 0x8006800C 483
HW_TIMROT_TIMCOUNT0 ..... 0x80068030 487
HW_TIMROT_TIMCOUNT1 ..... 0x80068050 489
HW_TIMROT_TIMCOUNT2 ..... 0x80068070 491
HW_TIMROT_TIMCOUNT3 ..... 0x80068090 494
HW_TIMROT_TIMCTRL0 ..... 0x80068020 485
HW_TIMROT_TIMCTRL0_CLR 0x80068028 485
HW_TIMROT_TIMCTRL0_SET 0x80068024 485
HW_TIMROT_TIMCTRL0_TOG 0x8006802C 485
HW_TIMROT_TIMCTRL1 ..... 0x80068040 487
HW_TIMROT_TIMCTRL1_CLR 0x80068048 487
HW_TIMROT_TIMCTRL1_SET 0x80068044 487
HW_TIMROT_TIMCTRL1_TOG 0x8006804C 487
HW_TIMROT_TIMCTRL2 ..... 0x80068060 490
HW_TIMROT_TIMCTRL2_CLR 0x80068068 490
HW_TIMROT_TIMCTRL2_SET 0x80068064 490
HW_TIMROT_TIMCTRL2_TOG 0x8006806C 490
HW_TIMROT_TIMCTRL3 ..... 0x80068080 492
HW_TIMROT_TIMCTRL3_CLR 0x80068088 492
HW_TIMROT_TIMCTRL3_SET 0x80068084 492
HW_TIMROT_TIMCTRL3_TOG 0x8006808C 492
HW_UARTAPP_CTRL0 ..... 0x8006C000 575
HW_UARTAPP_CTRL0_CLR 0x8006C008 575
HW_UARTAPP_CTRL0_SET 0x8006C004 575
HW_UARTAPP_CTRL0_TOG 0x8006C00C 575
HW_UARTAPP_CTRL1 ..... 0x8006C010 576
HW_UARTAPP_CTRL1_CLR 0x8006C018 576
HW_UARTAPP_CTRL1_SET 0x8006C014 576
HW_UARTAPP_CTRL1_TOG 0x8006C01C 576
HW_UARTAPP_CTRL2 ..... 0x8006C020 577
HW_UARTAPP_CTRL2_CLR 0x8006C028 577
HW_UARTAPP_CTRL2_SET 0x8006C024 577
HW_UARTAPP_CTRL2_TOG 0x8006C02C 577
HW_UARTAPP_DATA 0x8006C050 583
HW_UARTAPP_DEBUG 0x8006C070 586
HW_UARTAPP_INTR 0x8006C040 582
HW_UARTAPP_INTR_CLR 0x8006C048 582
HW_UARTAPP_INTR_SET 0x8006C044 582

HW_UARTAPP_INTR_TOG 0x8006C04C 582
HW_UARTAPP_LINECTRL 0x8006C030 580
HW_UARTAPP_LINECTRL_CLR 0x8006C038 580
HW_UARTAPP_LINECTRL_SET 0x8006C034 580
HW_UARTAPP_LINECTRL_TOG 0x8006C03C 580
HW_UARTAPP_STAT 0x8006C060 584
HW_UARTDBGCR 0x80070030 599
HW_UARTDBGDMACR 0x80070048 606
HW_UARTDBGDR 0x80070000 592
HW_UARTDBGFBRD 0x80070028 597
HW_UARTDBGFR 0x80070018 594
HW_UARTDBGIBRD 0x80070024 596
HW_UARTDBGICR 0x80070044 604
HW_UARTDBGIFLS 0x80070034 601
HW_UARTDBGILPR 0x80070020 595
HW_UARTDBGIMSC 0x80070038 602
HW_UARTDBGLCR_H 0x8007002C 597
HW_UARTDBGMIS 0x80070040 603
HW_UARTDBGRIS 0x8007003C 603
HW_UARTDBGRSR_ECR 0x80070004 594
HW_USBPHY_CTRL 0x8007c030 173
HW_USBPHY_CTRL_CLR 0x8007c038 174
HW_USBPHY_CTRL_SET 0x8007c034 174
HW_USBPHY_CTRL_TOG 0x8007c03C 174
HW_USBPHY_DEBUG 0x8007c050 176
HW_USBPHY_DEBUG_CLR 0x8007c058 176
HW_USBPHY_DEBUG_SET 0x8007c054 176
HW_USBPHY_DEBUG_TOG 0x8007c05C 176
HW_USBPHY_DEBUG0_STATUS 0x8007c060 177
HW_USBPHY_DEBUG1_STATUS 0x8007c070 178
HW_USBPHY_DEBUG2_STATUS 0x8007c080 179
HW_USBPHY_DEBUG3_STATUS 0x8007c090 180
HW_USBPHY_DEBUG4_STATUS 0x8007c0a0 181
HW_USBPHY_DEBUG5_STATUS 0x8007c0b0 181
HW_USBPHY_DEBUG6_STATUS 0x8007c0c0 182
HW_USBPHY_DEBUG7_STATUS 0x8007c0d0 183
HW_USBPHY_DEBUG8_STATUS 0x8007c0e0 184
HW_USBPHY_PWD 0x8007c000 169
HW_USBPHY_PWD_CLR 0x8007c008 169
HW_USBPHY_PWD_SET 0x8007c004 169
HW_USBPHY_PWD_TOG 0x8007c00C 169
HW_USBPHY_RX 0x8007c020 172
HW_USBPHY_RX_CLR 0x8007c028 172
HW_USBPHY_RX_SET 0x8007c024 172
HW_USBPHY_RX_TOG 0x8007c02C 172
HW_USBPHY_STATUS 0x8007c040 175
HW_USBPHY_TX 0x8007c010 170
HW_USBPHY_TX_CLR 0x8007c018 170
HW_USBPHY_TX_SET 0x8007c014 170
HW_USBPHY_TX_TOG 0x8007c01C 170