

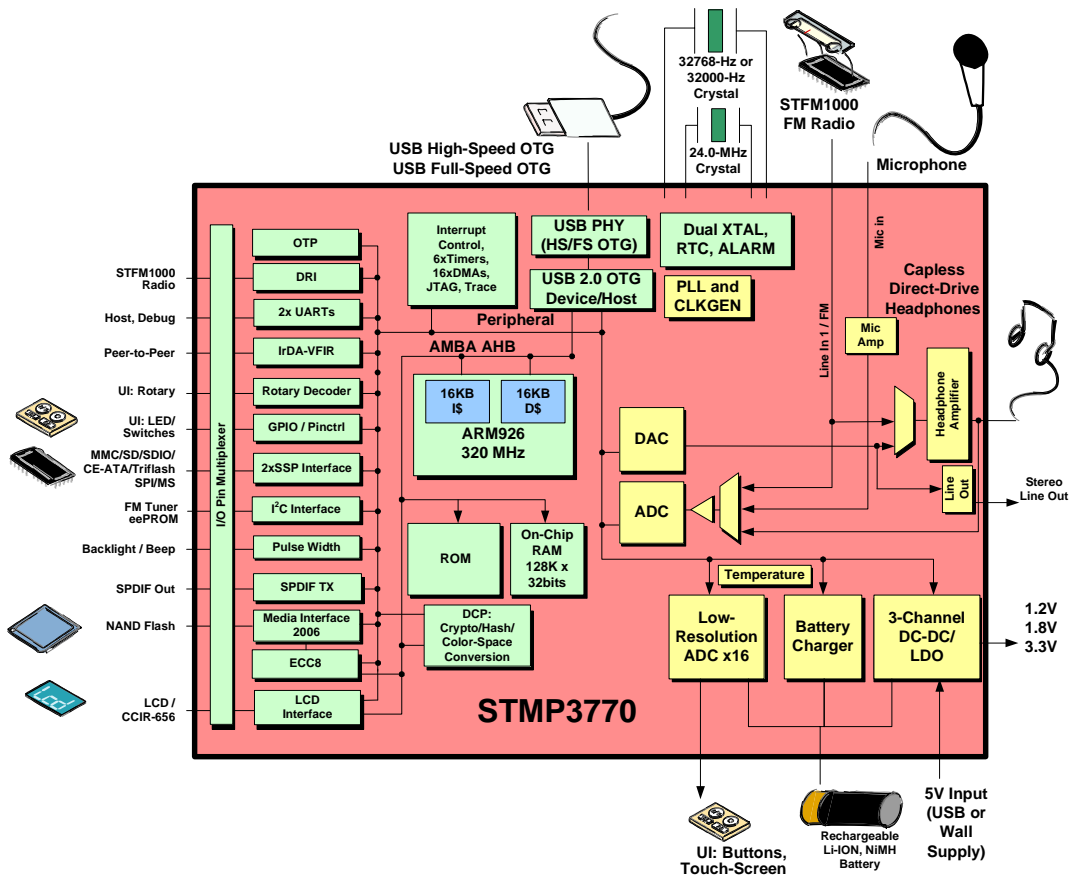
PRODUCT DATA SHEET

STMP3770

Media System on Chip

Fifth-Generation Audio/Video Decoder

Version 1.04 March 14, 2008



OFFICIAL PRODUCT DOCUMENTATION 3/14/08

5-37xx-DS2-1.04-031408

Copyright © 2007–2008 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.

SigmaTel and the SigmaTel logo are trademarks of SigmaTel, Inc. and may be used to identify SigmaTel products only. Windows Media and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries. Other product and company names contained herein may be trademarks of their respective owners.

STMP3770**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	23
1. PRODUCT OVERVIEW	25
1.1. Hardware Features	26
1.2. Application Capability	28
1.3. STMP3770 Product Features	29
1.3.1. ARM 926 Processor Core	30
1.3.2. System Buses	31
1.3.3. On-Chip RAM and ROM	32
1.3.4. On-Chip One-Time-Programmable (OCOTP) ROM	33
1.3.5. Interrupt Collector	35
1.3.6. Default First-Level Page Table	35
1.3.7. DMA Controller	35
1.3.8. Clock Generation Subsystem	36
1.3.9. Power Management Unit	36
1.3.10. USB Interface	36
1.3.11. General-Purpose Media Interface (GPMI)	37
1.3.12. Hardware Acceleration for ECC for Robust External Storage	37
1.3.13. Data Co-Processor (DCP)—Memory Copy, Crypto, and Color-Space Converter	38
1.3.14. Mixed Signal Audio Subsystem	38
1.3.15. Master Digital Control Unit (DIGCTL)	40
1.3.16. Synchronous Serial Port (SSP)	40
1.3.17. I ² C Interface	40
1.3.18. General-Purpose Input/Output (GPIO)	40
1.3.19. LCD Interface (LCDIF)	40
1.3.20. SPDIF Transmitter	41
1.3.21. Rotary Decoder	41
1.3.22. Dual UARTs	41
1.3.23. Low-Resolution ADC, Touch-Screen Interface, and Temperature Sensor	41
1.3.24. Pulse Width Modulator (PWM) Controller	42
2. CHARACTERISTICS AND SPECIFICATIONS	43
2.1. Absolute Maximum Ratings	43
2.2. Recommended Operating Conditions	44
2.2.1. Recommended Operating Conditions for Specific Clock Targets	45
2.3. DC Characteristics	46
3. ARM CPU COMPLEX	47
3.1. ARM 926 Processor Core	47
3.2. JTAG Debugger	49
3.2.1. JTAG READ ID	49
3.2.2. JTAG Hardware Reset	49
3.2.3. JTAG Interaction with CPUCLK	49
3.3. Embedded Trace Macrocell (ETM) Interface	50
4. CLOCK GENERATION AND CONTROL	51
4.1. Overview	51
4.2. Crystal Oscillators	51
4.3. Phase-Locked Loop (PLL)	52
4.4. Clock Domains	52
4.4.1. CLK_P, CLK_P_NG	52
4.4.2. CLK_H	52
4.4.3. HCLK_EN	54
4.4.4. CLK_SSP	55
4.4.5. CLK_GPMI	55
4.4.6. CLK_PCMSPDIF and CLK_SPDIF	55
4.4.7. CLK_PIX	55
4.4.8. CLK_X	55
4.4.9. CLK_UART	55
4.4.10. CLK_XTAL24M	55
4.4.11. CLK_1M	55
4.4.12. CLK_32k	55

4.4.13. CLK_ADC	56
4.5. Clock Dividers	56
4.5.1. Integer Clock Divider	56
4.5.2. Fractional Clock Divider	56
4.5.3. Phase Fractional Divider (PFD) Control	57
4.6. Programming the Clock Controller	58
4.6.1. Clock Frequency management	58
4.7. Reset	58
4.7.1. Soft Reset	58
4.8. Programmable Registers	59
4.8.1. PLL Control Register 0 Description	59
4.8.2. PLL Control Register 1 Description	61
4.8.3. CPU Clock Control Register Description	61
4.8.4. AHB and APBH Bus Clock Control Register Description	62
4.8.5. APBX Clock Control Register Description	64
4.8.6. XTAL Clock Control Register Description	65
4.8.7. PIXCLK (LCDIF) Clock Control Register Description	66
4.8.8. Synchronous Serial Port Clock Control Register Description	67
4.8.9. General-Purpose Media Interface Clock Control Register Description	68
4.8.10. SPDIF Clock Control Register Description	69
4.8.11. Fractional Clock Control Register Description	70
4.8.12. Clock Frequency Sequence Control Register Description	72
4.8.13. System Software Reset Register Description	73
4.8.14. ClkCtrl Version Description	74
5. INTERRUPT COLLECTOR	75
5.1. Overview	75
5.2. Operation	76
5.2.1. Nesting of Multi-Level IRQ Interrupts	77
5.2.2. FIQ Generation	80
5.2.3. Interrupt Sources	81
5.2.4. CPU Wait-for-Interrupt Mode	83
5.3. Behavior During Reset	83
5.4. Programmable Registers	84
5.4.1. Interrupt Collector Interrupt Vector Address Register Description	84
5.4.2. Interrupt Collector Level Acknowledge Register Description	84
5.4.3. Interrupt Collector Control Register Description	85
5.4.4. Interrupt Collector Status Register Description	88
5.4.5. Interrupt Collector Raw Interrupt Input Register 0 Description	89
5.4.6. Interrupt Collector Raw Interrupt Input Register 1 Description	89
5.4.7. Interrupt Collector Priority Register 0 Description	90
5.4.8. Interrupt Collector Priority Register 1 Description	92
5.4.9. Interrupt Collector Priority Register 2 Description	93
5.4.10. Interrupt Collector Priority Register 3 Description	95
5.4.11. Interrupt Collector Priority Register 4 Description	97
5.4.12. Interrupt Collector Priority Register 5 Description	98
5.4.13. Interrupt Collector Priority Register 6 Description	101
5.4.14. Interrupt Collector Priority Register 7 Description	102
5.4.15. Interrupt Collector Priority Register 8 Description	104
5.4.16. Interrupt Collector Priority Register 9 Description	106
5.4.17. Interrupt Collector Priority Register 10 Description	107
5.4.18. Interrupt Collector Priority Register 11 Description	109
5.4.19. Interrupt Collector Priority Register 12 Description	111
5.4.20. Interrupt Collector Priority Register 13 Description	112
5.4.21. Interrupt Collector Priority Register 14 Description	114
5.4.22. Interrupt Collector Priority Register 15 Description	116
5.4.23. Interrupt Collector Interrupt Vector Base Address Register Description	118
5.4.24. Interrupt Collector Debug Register 0 Description	118
5.4.25. Interrupt Collector Debug Read Register 0 Description	120
5.4.26. Interrupt Collector Debug Read Register 1 Description	120
5.4.27. Interrupt Collector Debug Flag Register Description	121
5.4.28. Interrupt Collector Debug Read Request Register 0 Description	121
5.4.29. Interrupt Collector Debug Read Request Register 1 Description	122
5.4.30. Interrupt Collector Version Register Description	122

6. DEFAULT FIRST-LEVEL PAGE TABLE (DFLPT)	125
6.1. Overview	125
6.2. Operation	125
6.2.1. Memory Map	126
6.2.2. Default First-Level Page Table PIO Register Map Entry 2048	128
7. DIGITAL CONTROL AND ON-CHIP RAM	129
7.1. Overview	129
7.2. SRAM Controls	130
7.3. Miscellaneous Controls	131
7.3.1. Performance Monitoring	131
7.3.2. High-Entropy PRN Seed	131
7.3.3. Write-Once Register	131
7.3.4. Microseconds Counter	131
7.4. Programmable Registers	131
7.4.1. DIGCTL Control Register Description	131
7.4.2. DIGCTL Status Register Description	135
7.4.3. Free-Running HCLK Counter Register Description	137
7.4.4. On-Chip RAM Control Register Description	137
7.4.5. On-Chip RAM Repair Address Register Description	138
7.4.6. On-Chip ROM Control Register Description	138
7.4.7. Software Write-Once Register Description	139
7.4.8. Entropy Register Description	139
7.4.9. Entropy Latched Register Description	140
7.4.10. SJTAG Debug Register Description	140
7.4.11. Digital Control Microseconds Counter Register Description	142
7.4.12. Digital Control Debug Read Test Register Description	142
7.4.13. Digital Control Debug Register Description	143
7.4.14. SRAM BIST Control and Status Register Description	143
7.4.15. SRAM Status Register 0 Description	144
7.4.16. SRAM Status Register 1 Description	145
7.4.17. SRAM Status Register 2 Description	145
7.4.18. SRAM Status Register 3 Description	146
7.4.19. SRAM Status Register 4 Description	146
7.4.20. SRAM Status Register 5 Description	147
7.4.21. SRAM Status Register 6 Description	147
7.4.22. SRAM Status Register 7 Description	148
7.4.23. SRAM Status Register 8 Description	148
7.4.24. SRAM Status Register 9 Description	149
7.4.25. SRAM Status Register 10 Description	150
7.4.26. SRAM Status Register 11 Description	150
7.4.27. SRAM Status Register 12 Description	151
7.4.28. SRAM Status Register 13 Description	152
7.4.29. Digital Control Scratch Register 0 Description	153
7.4.30. Digital Control Scratch Register 1 Description	153
7.4.31. Digital Control ARM Cache Register Description	153
7.4.32. Debug Trap Range Low Address Description	154
7.4.33. Debug Trap Range High Address Description	155
7.4.34. Digital Control Chip Revision Register Description	155
7.4.35. AHB Statistics Control Register Description	156
7.4.36. AHB Layer 0 Transfer Count Register Description	156
7.4.37. AHB Layer 0 Performance Metric for Stalled Bus Cycles Register Description	157
7.4.38. AHB Layer 0 Performance Metric for Valid Bus Cycles Register Description	158
7.4.39. AHB Layer 1 Transfer Count Register Description	158
7.4.40. AHB Layer 1 Performance Metric for Stalled Bus Cycles Register Description	159
7.4.41. AHB Layer 1 Performance Metric for Valid Bus Cycles Register Description	159
7.4.42. AHB Layer 2 Transfer Count Register Description	160
7.4.43. AHB Layer 2 Performance Metric for Stalled Bus Cycles Register Description	160
7.4.44. AHB Layer 2 Performance Metric for Valid Bus Cycles Register Description	161
7.4.45. AHB Layer 3 Transfer Count Register Description	162
7.4.46. AHB Layer 3 Performance Metric for Stalled Bus Cycles Register Description	162
7.4.47. AHB Layer 3 Performance Metric for Valid Bus Cycles Register Description	163
7.4.48. Default First-Level Page Table Movable PTE Locator 0 Description	163
7.4.49. Default First-Level Page Table Movable PTE Locator 1 Description	164

7.4.50. Default First-Level Page Table Movable PTE Locator 2 Description	164
7.4.51. Default First-Level Page Table Movable PTE Locator 3 Description	165
7.4.52. Default First-Level Page Table Movable PTE Locator 4 Description	166
7.4.53. Default First-Level Page Table Movable PTE Locator 5 Description	166
7.4.54. Default First-Level Page Table Movable PTE Locator 6 Description	167
7.4.55. Default First-Level Page Table Movable PTE Locator 7 Description	167
8. ON-CHIP OTP (OCOTP) CONTROLLER	169
8.1. Overview	169
8.2. Operation	170
8.2.1. Software Read Sequence	171
8.2.2. Software Write Sequence	172
8.2.3. Write Postamble	173
8.2.4. Shadow Registers and Hardware Capability Bus	173
8.3. Behavior During Reset	174
8.4. Programmable Registers	174
8.4.1. OTP Controller Control Register Description	174
8.4.2. OTP Controller Write Data Register Description	176
8.4.3. Value of OTP Bank 0 Word 0 (Customer) Description	176
8.4.4. Value of OTP Bank 0 Word 1 (Customer) Description	177
8.4.5. Value of OTP Bank 0 Word 2 (Customer) Description	177
8.4.6. Value of OTP Bank 0 Word 3 (Customer) Description	177
8.4.7. Value of OTP Bank 0 Word 4 (Crypto Key) Description	178
8.4.8. Value of OTP Bank 0 Word 5 (Crypto Key) Description	178
8.4.9. Value of OTP Bank 0 Word 6 (Crypto Key) Description	179
8.4.10. Value of OTP Bank 0 Word 7 (Crypto Key) Description	179
8.4.11. Customer Capability Shadow Register Description	180
8.4.12. LOCK Shadow Register OTP Bank 2 Word 0 Description	180
8.4.13. Value of OTP Bank 3 Word 0 (ROM Use 0) Description	181
8.4.14. Value of OTP Bank 3 Word 1 (ROM Use 1) Description	182
8.4.15. Value of OTP Bank 3 Word 2 (ROM Use 2) Description	185
8.4.16. OTP Controller Version Register Description	185
9. USB HIGH-SPEED ON-THE-GO (HOST/DEVICE) CONTROLLER	187
9.1. Overview	187
9.2. USB Programmed I/O (PIO) Target Interface	189
9.3. USB DMA Interface	189
9.4. STMP3770USB UTM Interface	189
9.4.1. Digital/Analog Loopback Test Mode	189
9.5. USB Controller Flowcharts	190
9.5.1. References	193
9.6. Programmable Registers	193
9.6.1. Identification Register Description	193
9.6.2. General Hardware Parameters Register Description	194
9.6.3. Host Hardware Parameters Register Description	194
9.6.4. Device Hardware Parameters Register Description	195
9.6.5. TX Buffer Hardware Parameters Register Description	195
9.6.6. RX Buffer Hardware Parameters Register Description	196
9.6.7. General-Purpose Timer 0 Load (Non-EHCI-Compliant) Register Description	196
9.6.8. General-Purpose Timer 0 Control (Non-EHCI-Compliant) Register Description	197
9.6.9. General-Purpose Timer 1 Load (Non-EHCI-Compliant) Register Description	198
9.6.10. General-Purpose Timer 1 Control (Non-EHCI-Compliant) Register Description	198
9.6.11. Capability Register Length (EHCI-Compliant) Register Description	198
9.6.12. Host Interface Version Number (EHCI-Compliant) Register Description	199
9.6.13. Host Control Structural Parameters (EHCI-Compliant with Extensions) Register Description	199
9.6.14. Host Control Capability Parameters (EHCI-Compliant) Register Description	201
9.6.15. Device Interface Version Number (Non-EHCI-Compliant) Register Description	202
9.6.16. Device Control Capability Parameters (Non-EHCI-Compliant) Register Description	203
9.6.17. USB Command Register Description	203
9.6.18. USB Status Register Description	208
9.6.19. USB Interrupt Enable Register Description	212
9.6.20. USB Frame Index Register Description	214
9.6.21. PERIODICLISTBASE and DEVADDR Register Descriptions	215

9.6.22. ASYNCLISTADDR and ENDPTLISTADDR Register Descriptions	217
9.6.23. Host Controller Embedded TT Asynchronous Buffer Status and Control Register Description	218
9.6.24. Programmable Burst Size Register Description	219
9.6.25. Host Transmit Pre-Buffer Packet Timing Register Description	220
9.6.26. ULPI Viewport Register Description	222
9.6.27. Endpoint NAK Register Description	223
9.6.28. Endpoint NAK Enable Register Description	224
9.6.29. Port Status and Control 1 Register Description	224
9.6.30. OTG Status and Control Register Description	232
9.6.31. USB Device Mode Register Description	235
9.6.32. Endpoint Setup Status Register Description	238
9.6.33. Endpoint Initialization Register Description	238
9.6.34. Endpoint De-Initialize Register Description	239
9.6.35. Endpoint Status Register Description	240
9.6.36. Endpoint Compete Register Description	241
9.6.37. Endpoint Control 0 Register Description	242
9.6.38. Endpoint Control 1–4 Registers Description	244
10. INTEGRATED USB 2.0 PHY	249
10.1. Overview	249
10.2. Operation	249
10.2.1. UTMI	250
10.2.2. Digital Transmitter	250
10.2.3. Digital Receiver	250
10.2.4. Analog Receiver	250
10.2.5. Analog Transmitter	252
10.2.6. Recommended Register Configuration for USB Certification	255
10.3. Behavior During Reset	255
10.4. Programmable Registers	256
10.4.1. USB PHY Power-Down Register Description	256
10.4.2. USB PHY Transmitter Control Register Description	257
10.4.3. USB PHY Receiver Control Register Description	258
10.4.4. USB PHY General Control Register Description	259
10.4.5. USB PHY Status Register Description	261
10.4.6. USB PHY Debug Register Description	262
10.4.7. UTMI Debug Status Register 0 Description	264
10.4.8. UTMI Debug Status Register 1 Description	264
10.4.9. UTMI RTL Version Description	265
11. AHB-TO-APBH BRIDGE WITH DMA	267
11.1. Overview	267
11.2. AHBH DMA	268
11.3. Implementation Examples	272
11.3.1. NAND Read Status Polling Example	272
11.4. Behavior During Reset	274
11.5. Programmable Registers	274
11.5.1. AHB-to-APBH Bridge Control and Status Register 0 Description	274
11.5.2. AHB-to-APBH Bridge Control and Status Register 1 Description	275
11.5.3. AHB-to-APBH DMA Device Assignment Register Description	278
11.5.4. APBH DMA Channel 0 Current Command Address Register Description	279
11.5.5. APBH DMA Channel 0 Next Command Address Register Description	279
11.5.6. APBH DMA Channel 0 Command Register Description	280
11.5.7. APBH DMA Channel 0 Buffer Address Register Description	282
11.5.8. APBH DMA Channel 0 Semaphore Register Description	282
11.5.9. AHB-to-APBH DMA Channel 0 Debug Information Register Description	283
11.5.10. AHB-to-APBH DMA Channel 0 Debug Information Register Description	286
11.5.11. APBH DMA Channel 1 Current Command Address Register Description	286
11.5.12. APBH DMA Channel 1 Next Command Address Register Description	287
11.5.13. APBH DMA Channel 1 Command Register Description	287
11.5.14. APBH DMA Channel 1 Buffer Address Register Description	289
11.5.15. APBH DMA Channel 1 Semaphore Register Description	290
11.5.16. AHB-to-APBH DMA Channel 1 Debug Information Register Description	291
11.5.17. AHB-to-APBH DMA Channel 1 Debug Information Register Description	293

11.5.18. APBH DMA Channel 2 Current Command Address Register Description	293
11.5.19. APBH DMA Channel 2 Next Command Address Register Description	294
11.5.20. APBH DMA Channel 2 Command Register Description	294
11.5.21. APBH DMA Channel 2 Buffer Address Register Description	296
11.5.22. APBH DMA Channel 2 Semaphore Register Description	297
11.5.23. AHB-to-APBH DMA Channel 2 Debug Information Register Description	297
11.5.24. AHB-to-APBH DMA Channel 2 Debug Information Register Description	299
11.5.25. APBH DMA Channel 3 Current Command Address Register Description	300
11.5.26. APBH DMA Channel 3 Next Command Address Register Description	300
11.5.27. APBH DMA Channel 3 Command Register Description	301
11.5.28. APBH DMA Channel 3 Buffer Address Register Description	302
11.5.29. APBH DMA Channel 3 Semaphore Register Description	302
11.5.30. AHB-to-APBH DMA Channel 3 Debug Information Register Description	303
11.5.31. AHB-to-APBH DMA Channel 3 Debug Information Register Description	305
11.5.32. APBH DMA Channel 4 Current Command Address Register Description	306
11.5.33. APBH DMA Channel 4 Next Command Address Register Description	306
11.5.34. APBH DMA Channel 4 Command Register Description	307
11.5.35. APBH DMA Channel 4 Buffer Address Register Description	309
11.5.36. APBH DMA Channel 4 Semaphore Register Description	309
11.5.37. AHB-to-APBH DMA Channel 4 Debug Information Register Description	310
11.5.38. AHB-to-APBH DMA Channel 4 Debug Information Register Description	312
11.5.39. APBH DMA Channel 5 Current Command Address Register Description	313
11.5.40. APBH DMA Channel 5 Next Command Address Register Description	313
11.5.41. APBH DMA Channel 5 Command Register Description	314
11.5.42. APBH DMA Channel 5 Buffer Address Register Description	316
11.5.43. APBH DMA Channel 5 Semaphore Register Description	317
11.5.44. AHB-to-APBH DMA Channel 5 Debug Information Register Description	317
11.5.45. AHB-to-APBH DMA Channel 5 Debug Information Register Description	319
11.5.46. APBH DMA Channel 6 Current Command Address Register Description	320
11.5.47. APBH DMA Channel 6 Next Command Address Register Description	320
11.5.48. APBH DMA Channel 6 Command Register Description	321
11.5.49. APBH DMA Channel 6 Buffer Address Register Description	323
11.5.50. APBH DMA Channel 6 Semaphore Register Description	324
11.5.51. AHB-to-APBH DMA Channel 6 Debug Information Register Description	324
11.5.52. AHB-to-APBH DMA Channel 6 Debug Information Register Description	326
11.5.53. APBH DMA Channel 7 Current Command Address Register Description	327
11.5.54. APBH DMA Channel 7 Next Command Address Register Description	327
11.5.55. APBH DMA Channel 7 Command Register Description	328
11.5.56. APBH DMA Channel 7 Buffer Address Register Description	330
11.5.57. APBH DMA Channel 7 Semaphore Register Description	331
11.5.58. AHB-to-APBH DMA Channel 7 Debug Information Register Description	331
11.5.59. AHB-to-APBH DMA Channel 7 Debug Information Register Description	333
11.5.60. APBH Bridge Version Register Description	334
12. AHB-TO-APBX BRIDGE WITH DMA	335
12.1. Overview	335
12.2. APBX DMA	336
12.3. DMA Chain Example	339
12.4. Behavior During Reset	340
12.5. Programmable Registers	341
12.5.1. AHB-to-APBX Bridge Control and Status Register 0 Description	341
12.5.2. AHB-to-APBX Bridge Control and Status Register 1 Description	342
12.5.3. AHB-to-APBX DMA Device Assignment Register Description	344
12.5.4. APBX DMA Channel 0 Current Command Address Register Description	345
12.5.5. APBX DMA Channel 0 Next Command Address Register Description	346
12.5.6. APBX DMA Channel 0 Command Register Description	346
12.5.7. APBX DMA Channel 0 Buffer Address Register Description	348
12.5.8. APBX DMA Channel 0 Semaphore Register Description	349
12.5.9. AHB-to-APBX DMA Channel 0 Debug Information Register Description	350
12.5.10. AHB-to-APBX DMA Channel 0 Debug Information Register Description	351
12.5.11. APBX DMA Channel 1 Current Command Address Register Description	352
12.5.12. APBX DMA Channel 1 Next Command Address Register Description	353
12.5.13. APBX DMA Channel 1 Command Register Description	353
12.5.14. APBX DMA Channel 1 Buffer Address Register Description	355

12.5.15. APBX DMA Channel 1 Semaphore Register Description	355
12.5.16. AHB-to-APBX DMA Channel 1 Debug Information Register Description	356
12.5.17. AHB-to-APBX DMA Channel 1 Debug Information Register Description	358
12.5.18. APBX DMA Channel 2 Current Command Address Register Description	359
12.5.19. APBX DMA Channel 2 Next Command Address Register Description	359
12.5.20. APBX DMA Channel 2 Command Register Description	360
12.5.21. APBX DMA Channel 2 Buffer Address Register Description	361
12.5.22. APBX DMA Channel 2 Semaphore Register Description	362
12.5.23. AHB-to-APBX DMA Channel 2 Debug Information Register Description	363
12.5.24. AHB-to-APBX DMA Channel 2 Debug Information Register Description	364
12.5.25. APBX DMA Channel 3 Current Command Address Register Description	365
12.5.26. APBX DMA Channel 3 Next Command Address Register Description	365
12.5.27. APBX DMA Channel 3 Command Register Description	366
12.5.28. APBX DMA Channel 3 Buffer Address Register Description	367
12.5.29. APBX DMA Channel 3 Semaphore Register Description	368
12.5.30. AHB-to-APBX DMA Channel 3 Debug Information Register Description	368
12.5.31. AHB-to-APBX DMA Channel 3 Debug Information Register Description	370
12.5.32. APBX DMA Channel 4 Current Command Address Register Description	371
12.5.33. APBX DMA Channel 4 Next Command Address Register Description	371
12.5.34. APBX DMA Channel 4 Command Register Description	372
12.5.35. APBX DMA Channel 4 Buffer Address Register Description	373
12.5.36. APBX DMA Channel 4 Semaphore Register Description	374
12.5.37. AHB-to-APBX DMA Channel 4 Debug Information Register Description	374
12.5.38. AHB-to-APBX DMA Channel 4 Debug Information Register Description	376
12.5.39. APBX DMA Channel 5 Current Command Address Register Description	377
12.5.40. APBX DMA Channel 5 Next Command Address Register Description	377
12.5.41. APBX DMA Channel 5 Command Register Description	378
12.5.42. APBX DMA Channel 5 Buffer Address Register Description	379
12.5.43. APBX DMA Channel 5 Semaphore Register Description	380
12.5.44. AHB-to-APBX DMA Channel 5 Debug Information Register Description	380
12.5.45. AHB-to-APBX DMA Channel 5 Debug Information Register Description	382
12.5.46. APBX DMA Channel 6 Current Command Address Register Description	383
12.5.47. APBX DMA Channel 6 Next Command Address Register Description	383
12.5.48. APBX DMA Channel 6 Command Register Description	384
12.5.49. APBX DMA Channel 6 Buffer Address Register Description	385
12.5.50. APBX DMA Channel 6 Semaphore Register Description	386
12.5.51. AHB-to-APBX DMA Channel 6 Debug Information Register Description	386
12.5.52. AHB-to-APBX DMA Channel 6 Debug Information Register Description	388
12.5.53. APBX DMA Channel 7 Current Command Address Register Description	389
12.5.54. APBX DMA Channel 7 Next Command Address Register Description	389
12.5.55. APBX DMA Channel 7 Command Register Description	390
12.5.56. APBX DMA Channel 7 Buffer Address Register Description	391
12.5.57. APBX DMA Channel 7 Semaphore Register Description	392
12.5.58. AHB-to-APBX DMA Channel 7 Debug Information Register Description	392
12.5.59. AHB-to-APBX DMA Channel 7 Debug Information Register Description	394
12.5.60. APBX Bridge Version Register Description	395
13. GENERAL-PURPOSE MEDIA INTERFACE (GPMI)	397
13.1. Overview	397
13.2. GPMI NAND Mode	398
13.2.1. Multiple NAND Support	399
13.2.2. GPMI NAND Timing and Clocking	399
13.2.3. Basic NAND Timing	399
13.2.4. High-Speed NAND Timing	400
13.2.5. NAND Command and Address Timing Example	402
13.2.6. Hardware ECC (ECC8) Interface	402
13.3. Behavior During Reset	403
13.4. Programmable Registers	403
13.4.1. GPMI Control Register 0 Description	403
13.4.2. GPMI Compare Register Description	405
13.4.3. GPMI Integrated ECC Control Register Description	405
13.4.4. GPMI Integrated ECC Transfer Count Register Description	407
13.4.5. GPMI Payload Address Register Description	408
13.4.6. GPMI Auxiliary Address Register Description	408

13.4.7. GPIM Control Register 1 Description	409
13.4.8. GPIM Timing Register 0 Description	410
13.4.9. GPIM Timing Register 1 Description	411
13.4.10. GPIM Timing Register 2 Description	412
13.4.11. GPIM DMA Data Transfer Register Description	413
13.4.12. GPIM Status Register Description	413
13.4.13. GPIM Debug Information Register Description	415
13.4.14. GPIM Version Register Description	416
14. 8-SYMBOL CORRECTING ECC ACCELERATOR (ECC8)	419
14.1. Overview	419
14.2. Operation	421
14.2.1. Reed-Solomon ECC Accelerator	426
14.2.2. Reed-Solomon ECC Encoding for NAND Writes	428
14.2.3. Reed-Solomon ECC Decoding for NAND Reads	436
14.2.4. Interrupts	444
14.3. Behavior During Reset	445
14.4. Programmable Registers	445
14.4.1. Hardware ECC Accelerator Control Register Description	445
14.4.2. Hardware ECC Accelerator Status Register 0 Description	447
14.4.3. Hardware ECC Accelerator Status Register 1 Description	448
14.4.4. Hardware ECC Accelerator Debug Register 0	450
14.4.5. KES Debug Read Register Description	453
14.4.6. Chien Search Forney Evaluator Debug Read Register Description	453
14.4.7. Syndrome Generator Debug Read Register Description	453
14.4.8. AHB Master and ECC8 Controller Debug Read Register Description	454
14.4.9. ECC8 Block Name Register Description	454
14.4.10. ECC8 Version Register Description	455
15. DATA CO-PROCESSOR (DCP)	457
15.1. Overview	457
15.1.1. DCP Limitations for Software	460
15.2. Operation	460
15.2.1. Memory Copy, Blit, and Fill Functionality	460
15.2.2. Advanced Encryption Standard (AES)	461
15.2.3. Hashing	463
15.2.4. Color-Space Conversion (YUV/YCbCr to RGB)	463
15.2.5. Managing DCP Channel/CSC Arbitration and Performance	465
15.2.6. Programming Channel Operations	466
15.2.7. Programming the Color-Space Converter	475
15.2.8. Programming Other DCP Functions	478
15.3. Programmable Registers	484
15.3.1. DCP Control Register 0 Description	484
15.3.2. DCP Status Register Description	485
15.3.3. DCP Channel Control Register Description	487
15.3.4. DCP Capability 0 Register Description	488
15.3.5. DCP Capability 1 Register Description	489
15.3.6. DCP Context Buffer Pointer Description	489
15.3.7. DCP Key Index Register Description	490
15.3.8. DCP Key Data Description	491
15.3.9. DCP Work Packet 0 Status Register Description	491
15.3.10. DCP Work Packet 1 Status Register Description	492
15.3.11. DCP Work Packet 2 Status Register Description	493
15.3.12. DCP Work Packet 3 Status Register Description	494
15.3.13. DCP Work Packet 4 Status Register Description	495
15.3.14. DCP Work Packet 5 Status Register Description	495
15.3.15. DCP Work Packet 6 Status Register Description	496
15.3.16. DCP Channel 0 Command Pointer Address Register Description	496
15.3.17. DCP Channel 0 Semaphore Register Description	497
15.3.18. DCP Channel 0 Status Register Description	498
15.3.19. DCP Channel 0 Options Register Description	499
15.3.20. DCP Channel 1 Command Pointer Address Register Description	500
15.3.21. DCP Channel 1 Semaphore Register Description	501
15.3.22. DCP Channel 1 Status Register Description	502

15.3.23. DCP Channel 1 Options Register Description	504
15.3.24. DCP Channel 2 Command Pointer Address Register Description	504
15.3.25. DCP Channel 2 Semaphore Register Description	505
15.3.26. DCP Channel 2 Status Register Description	506
15.3.27. DCP Channel 2 Options Register Description	508
15.3.28. DCP Channel 3 Command Pointer Address Register Description	508
15.3.29. DCP Channel 3 Semaphore Register Description	509
15.3.30. DCP Channel 3 Status Register Description	510
15.3.31. DCP Channel 3 Options Register Description	512
15.3.32. Color-Space Conversion Control Register 0 Description	512
15.3.33. Color-Space Conversion Status Register Description	514
15.3.34. Color-Space Conversion Output Buffer Parameters Register Description	515
15.3.35. Color-Space Conversion Input Buffer Parameters Register Description	515
15.3.36. Color-Space RGB Frame Buffer Pointer Description	516
15.3.37. Color-Space Luma (Y) Buffer Pointer Description	517
15.3.38. Color-Space Chroma (U/Cb) Buffer Pointer Description	517
15.3.39. Color-Space Chroma (V/Cr) Buffer Pointer Description	518
15.3.40. Color-Space Conversion Coefficient Register 0 Description	518
15.3.41. Color-Space Conversion Coefficient Register 1 Description	519
15.3.42. Color-Space Conversion Coefficient Register 2 Description	520
15.3.43. Color-Space Conversion X-Scaling Register Description	520
15.3.44. Color-Space Conversion Y-Scaling Register Description	521
15.3.45. DCP Debug Select Register Description	522
15.3.46. DCP Debug Data Register Description	522
15.3.47. DCP Version Register Description	523
16. SYNCHRONOUS SERIAL PORTS (SSP)	525
16.1. Overview	525
16.2. External Pins	526
16.3. Bit Rate Generation	526
16.4. Frame Format for SPI and SSI	526
16.5. Motorola SPI Mode	527
16.5.1. SPI DMA Mode	527
16.5.2. Motorola SPI Frame Format	527
16.5.3. Motorola SPI Format with Polarity=0, Phase=0	527
16.5.4. Motorola SPI Format with Polarity=0, Phase=1	529
16.5.5. Motorola SPI Format with Polarity=1, Phase=0	529
16.5.6. Motorola SPI Format with Polarity=1, Phase=1	531
16.6. Texas Instruments Synchronous Serial Interface (SSI) Mode	532
16.7. SD/SDIO/MMC Mode	533
16.7.1. SD/MMC Command/Response Transfer	533
16.7.2. SD/MMC Data Block Transfer	534
16.7.3. SDIO Interrupts	536
16.7.4. SD/MMC Mode Error Handling	536
16.7.5. SD/MMC Clock Control	538
16.8. CE-ATA Mode	538
16.9. MS Mode	539
16.9.1. MS Mode I/O Pins	539
16.9.2. Basic MS Mode Protocol	539
16.9.3. MS Mode High-Level Operation	540
16.9.4. MS Mode Four-State Bus Protocol	540
16.9.5. Wait for Card IRQ	540
16.9.6. Checking Card Status	541
16.9.7. MS Mode Error Conditions	541
16.9.8. MS Mode Details	543
16.10. Behavior During Reset	543
16.11. Programmable Registers	544
16.11.1. SSP Control Register 0 Description	544
16.11.2. SD/MMC and MS Command Register 0 Description	546
16.11.3. SD/MMC Command Register 1 Description	550
16.11.4. SD/MMC and MS Compare Reference Register Description	550
16.11.5. SD/MMC and MS Compare Mask Register Description	550
16.11.6. SSP Timing Register Description	551
16.11.7. SSP Control Register 1 Description	551

16.11.8. SSP Data Register Description	554
16.11.9. SD/MMC Card Response Register 0 Description	555
16.11.10. SD/MMC Card Response Register 1 Description	555
16.11.11. SD/MMC Card Response Register 2 Description	556
16.11.12. SD/MMC Card Response Register 3 Description	556
16.11.13. SSP Status Register Description	556
16.11.14. SSP Debug Register Description	558
16.11.15. SSP Version Register Description	559
17. LCD INTERFACE (LCDIF)	561
17.1. Overview	561
17.2. Operation	561
17.2.1. DMA and FIFO Control	562
17.2.2. Write and Read Data Paths	563
17.2.3. System Interface	567
17.2.4. VSYNC Interface	569
17.2.5. DOTCLK Interface	571
17.2.6. Controlling VSYNC, HSYNC, and DOTCLK Signal Generation	573
17.2.7. ITU-R BT.656 Digital Video Interface (DVI)	573
17.2.8. LCDIF Pin Usage by Interface Mode	574
17.3. Behavior During Reset	575
17.4. Programmable Registers	576
17.4.1. LCDIF General Control Register Description	576
17.4.2. LCDIF General Control 1 Register Description	578
17.4.3. LCDIF Timing Register Description	581
17.4.4. LCDIF VSYNC Mode and DOTCLK Mode Control 0 Register Description	582
17.4.5. LCDIF VSYNC Mode and DOTCLK Mode Control 1 Register Description	584
17.4.6. LCDIF VSYNC Mode and DOTCLK Mode Control 2 Register Description	585
17.4.7. LCDIF VSYNC Mode and DOTCLK Mode Control 3 Register Description	586
17.4.8. Digital Video Interface Control 0 Register Description	587
17.4.9. Digital Video Interface Control 1 Register Description	588
17.4.10. Digital Video Interface Control 2 Register Description	589
17.4.11. Digital Video Interface Control 3 Register Description	590
17.4.12. LCDIF Data Register Description	591
17.4.13. LCDIF Status Register Description	592
17.4.14. LCDIF Version Register Description	593
17.4.15. LCDIF Debug 0 Register Description	593
18. TIMERS AND ROTARY DECODER	595
18.1. Overview	595
18.2. Timers	596
18.2.1. Using External Signals as Inputs	597
18.2.2. Timer 3 and Duty Cycle Mode	598
18.2.3. Testing Timer 3 Duty Cycle Modes	599
18.3. Rotary Decoder	600
18.3.1. Testing the Rotary Decoder	602
18.3.2. Behavior During Reset	602
18.4. Programmable Registers	603
18.4.1. Rotary Decoder Control Register Description	603
18.4.2. Rotary Decoder Up/Down Counter Register Description	604
18.4.3. Timer 0 Control and Status Register Description	605
18.4.4. Timer 0 Count Register Description	606
18.4.5. Timer 1 Control and Status Register Description	607
18.4.6. Timer 1 Count Register Description	609
18.4.7. Timer 2 Control and Status Register Description	609
18.4.8. Timer 2 Count Register Description	611
18.4.9. Timer 3 Control and Status Register Description	612
18.4.10. Timer 3 Count Register Description	614
18.4.11. TIMROT Version Register Description	615
19. REAL-TIME CLOCK, ALARM, WATCHDOG, PERSISTENT BITS	617
19.1. Overview	617
19.2. Programming and Enabling the RTC Clock	619
19.3. RTC Persistent Register Copy Control	619

19.4. Real-Time Clock Function	621
19.4.1. Behavior During Reset	621
19.5. Millisecond Resolution Timing Function	621
19.6. Alarm Clock Function	621
19.7. Watchdog Reset Function	622
19.8. Programmable Registers	622
19.8.1. Real-Time Clock Control Register Description	622
19.8.2. Real-Time Clock Status Register Description	624
19.8.3. Real-Time Clock Milliseconds Counter Register Description	625
19.8.4. Real-Time Clock Seconds Counter Register Description	626
19.8.5. Real-Time Clock Alarm Register Description	627
19.8.6. Watchdog Timer Register Description	627
19.8.7. Persistent State Register 0 Description	628
19.8.8. Persistent State Register 1 Description	630
19.8.9. Persistent State Register 2 Description	631
19.8.10. Persistent State Register 3 Description	631
19.8.11. Persistent State Register 4 Description	632
19.8.12. Persistent State Register 5 Description	632
19.8.13. Real-Time Clock Debug Register Description	633
19.8.14. Real-Time Clock Version Register Description	634
20. PULSE-WIDTH MODULATOR (PWM) CONTROLLER	635
20.1. Overview	635
20.2. Operation	635
20.2.1. Multi-Chip Attachment Mode	638
20.2.2. Channel 2 Analog Enable Function	639
20.2.3. Analog Feedback for Backlight Control Using PWM Channel 2	639
20.3. Behavior During Reset	640
20.4. Programmable Registers	640
20.4.1. PWM Control and Status Register Description	640
20.4.2. PWM Channel 0 Active Register Description	642
20.4.3. PWM Channel 0 Period Register Description	643
20.4.4. PWM Channel 1 Active Register Description	644
20.4.5. PWM Channel 1 Period Register Description	644
20.4.6. PWM Channel 2 Active Register Description	646
20.4.7. PWM Channel 2 Period Register Description	646
20.4.8. PWM Channel 3 Active Register Description	647
20.4.9. PWM Channel 3 Period Register Description	648
20.4.10. PWM Channel 4 Active Register Description	649
20.4.11. PWM Channel 4 Period Register Description	650
20.4.12. PWM Version Register Description	651
21. I²C INTERFACE	653
21.1. Overview	653
21.2. Operation	654
21.2.1. I ² C Interrupt Sources	654
21.2.2. I ² C Bus Protocol	655
21.2.3. Programming Examples	666
21.3. Behavior During Reset	669
21.4. Programmable Registers	669
21.4.1. I ² C Control Register 0 Description	669
21.4.2. I ² C Timing Register 0 Description	672
21.4.3. I ² C Timing Register 1 Description	672
21.4.4. I ² C Timing Register 2 Description	673
21.4.5. I ² C Control Register 1 Description	674
21.4.6. I ² C Status Register Description	677
21.4.7. I ² C Controller DMA Read and Write Data Register Description	680
21.4.8. I ² C Device Debug Register 0 Description	681
21.4.9. I ² C Device Debug Register 1 Description	682
21.4.10. I ² C Version Register Description	684
22. APPLICATION UART	685
22.1. Overview	685
22.2. Operation	686

STMP3770

22.2.1. Fractional Baud Rate Divider	687
22.2.2. UART Character Frame	687
22.2.3. DMA Operation	687
22.2.4. Data Transmission or Reception	687
22.2.5. Error Bits	688
22.2.6. Overrun Bit	688
22.2.7. Disabling the FIFOs	688
22.3. Behavior During Reset	689
22.4. Programmable Registers	689
22.4.1. UART Receive DMA Control Register Description	689
22.4.2. UART Transmit DMA Control Register Description	690
22.4.3. UART Control Register Description	691
22.4.4. UART Line Control Register Description	693
22.4.5. UART Line Control 2 Register Description	695
22.4.6. UART Interrupt Register Description	696
22.4.7. UART Data Register Description	697
22.4.8. UART Status Register Description	698
22.4.9. UART Debug Register Description	700
22.4.10. UART Version Register Description	701
23. DEBUG UART	703
23.1. Overview	703
23.2. Operation	704
23.2.1. Fractional Baud Rate Divider	704
23.2.2. UART Character Frame	705
23.2.3. Data Transmission or Reception	705
23.2.4. Error Bits	705
23.2.5. Overrun Bit	706
23.2.6. Disabling the FIFOs	706
23.3. Programmable Registers	706
23.3.1. UART Data Register Description	706
23.3.2. UART Receive Status (Read) and Error Clear (Write) Register Description	708
23.3.3. UART Flag Register Description	709
23.3.4. UART Integer Baud Rate Divisor Register Description	709
23.3.5. UART Fractional Baud Rate Divisor Register Description	710
23.3.6. UART Line Control Register, HIGH Byte Description	710
23.3.7. UART Control Register Description	712
23.3.8. UART Interrupt FIFO Level Select Register Description	713
23.3.9. UART Interrupt Mask Set/Clear Register Description	714
23.3.10. UART Raw Interrupt Status Register Description	715
23.3.11. UART Masked Interrupt Status Register Description	716
23.3.12. UART Interrupt Clear Register Description	717
23.3.13. UART DMA Control Register Description	718
24. AUDIOIN/ADC	719
24.1. Overview	719
24.2. Operation	720
24.2.1. AUDIOIN DMA	721
24.2.2. ADC Sample Rate Converter and Internal Operation	722
24.2.3. Microphone	725
24.3. Behavior During Reset	726
24.4. Programmable Registers	726
24.4.1. AUDIOIN Control Register Description	726
24.4.2. AUDIOIN Status Register Description	729
24.4.3. AUDIOIN Sample Rate Register Description	730
24.4.4. AUDIOIN Volume Register Description	731
24.4.5. AUDIOIN Debug Register Description	733
24.4.6. ADC Mux Volume and Select Control Register Description	735
24.4.7. Microphone and Line Control Register Description	737
24.4.8. Analog Clock Control Register Description	738
24.4.9. AUDIOIN Read Data Register Description	739
25. AUDIOOUT/DAC	741
25.1. Overview	741

25.2. Operation	742
25.2.1. AUDIOOUT DMA	743
25.2.2. DAC Sample Rate Converter and Internal Operation	744
25.2.3. Reference Control Settings	747
25.2.4. Headphone	748
25.3. Behavior During Reset	750
25.4. Programmable Registers	751
25.4.1. AUDIOOUT Control Register Description	751
25.4.2. AUDIOOUT Status Register Description	753
25.4.3. AUDIOOUT Sample Rate Register Description	754
25.4.4. AUDIOOUT Volume Register Description	756
25.4.5. AUDIOOUT Debug Register Description	758
25.4.6. Headphone Volume and Select Control Register Description	759
25.4.7. Reserved Register Description	761
25.4.8. Audio Power-Down Control Register Description	761
25.4.9. AUDIOOUT Reference Control Register Description	762
25.4.10. Miscellaneous Audio Controls Register Description	765
25.4.11. Miscellaneous Test Audio Controls Register Description	767
25.4.12. BIST Control and Status Register Description	769
25.4.13. Hardware BIST Status 0 Register Description	769
25.4.14. Hardware AUDIOOUT BIST Status 1 Register Description	770
25.4.15. Analog Clock Control Register Description	770
25.4.16. AUDIOOUT Write Data Register Description	771
25.4.17. AUDIOOUT Line Out Control Register Description	772
25.4.18. AUDIOOUT Version Register Description	774
26. SPDIF TRANSMITTER	775
26.1. Overview	775
26.2. Operation	775
26.2.1. Interrupts	778
26.2.2. Clocking	778
26.2.3. DMA Operation	778
26.2.4. PIO Debug Mode	780
26.3. Programmable Registers	781
26.3.1. SPDIF Control Register Description	781
26.3.2. SPDIF Status Register Description	783
26.3.3. SPDIF Frame Control Register Description	784
26.3.4. SPDIF Sample Rate Register Description	785
26.3.5. SPDIF Debug Register Description	786
26.3.6. SPDIF Write Data Register Description	786
26.3.7. SPDIF Version Register Description	787
27. DIGITAL RADIO INTERFACE (DRI)	789
27.1. Overview	789
27.2. Operation	790
27.3. Behavior During Reset	792
27.4. Programmable Registers	792
27.4.1. DRI Control Register Description	792
27.4.2. DRI Timing Register Description	795
27.4.3. DRI Status Register Description	796
27.4.4. DRI Controller DMA Read Data Register Description	797
27.4.5. DRI Device Debug Register 0 Description	797
27.4.6. DRI Device Debug Register 1 Description	799
27.4.7. DRI Version Register Description	800
28. LOW-RESOLUTION ADC AND TOUCH-SCREEN INTERFACE	801
28.1. Overview	801
28.2. Operation	802
28.2.1. External Temperature Sensing with a Diode	803
28.2.2. Internal Die Temperature Sensing	803
28.2.3. Scheduling Conversions	804
28.2.4. Delay Channels	804
28.3. Behavior During Reset	805
28.4. Programmable Registers	807

28.4.1. LRADC Control Register 0 Description	807
28.4.2. LRADC Control Register 1 Description	809
28.4.3. LRADC Control Register 2 Description	812
28.4.4. LRADC Control Register 3 Description	815
28.4.5. LRADC Status Register Description	817
28.4.6. LRADC 0 Result Register Description	818
28.4.7. LRADC 1 Result Register Description	819
28.4.8. LRADC 2 Result Register Description	821
28.4.9. LRADC 3 Result Register Description	822
28.4.10. LRADC 4 Result Register Description	823
28.4.11. LRADC 5 Result Register Description	824
28.4.12. LRADC 6 (VDDIO) Result Register Description	826
28.4.13. LRADC 7 (BATT) Result Register Description	827
28.4.14. LRADC Scheduling Delay Register 0 Description	828
28.4.15. LRADC Scheduling Delay Register 1 Description	830
28.4.16. LRADC Scheduling Delay Register 2 Description	831
28.4.17. LRADC Scheduling Delay Register 3 Description	833
28.4.18. LRADC Debug Register 0 Description	834
28.4.19. LRADC Debug Register 1 Description	835
28.4.20. LRADC Battery Conversion Register Description	836
28.4.21. LRADC Control Register 4 Description	838
28.4.22. LRADC Version Register Description	841
29. POWER SUPPLY	843
29.1. Overview	843
29.2. DC-DC Converters	846
29.2.1. DC-DC Operating Modes	846
29.2.2. DC-DC Operation	847
29.3. Linear Regulators	850
29.3.1. USB Compliance Features	850
29.3.2. 5V to Battery Power Interaction	850
29.4. Power-Up and Power-Down	852
29.4.1. Power-Up Sequence	852
29.4.2. Power-Down Sequence	852
29.4.3. Reset Sequence	853
29.5. PSWITCH Pin Functions	853
29.5.1. Power On	853
29.5.2. Power Down	853
29.5.3. Software Functions/Recovery Mode	853
29.6. Battery Monitor	856
29.7. Battery Charger	856
29.8. Silicon Speed Sensor	857
29.9. Interrupts	857
29.10. DC-DC Converter Efficiency	858
29.11. Programmable Registers	859
29.11.1. Power Control Register Description	859
29.11.2. DC-DC 5V Control Register Description	861
29.11.3. DC-DC Minimum Power and Miscellaneous Control Register Description	862
29.11.4. Battery Charge Control Register Description	864
29.11.5. VDDD Supply Targets and Brownouts Control Register Description	865
29.11.6. VDDA Supply Targets and Brownouts Control Register Description	867
29.11.7. VDDIO Supply Targets and Brownouts Control Register Description	869
29.11.8. DC-DC Multi-Output Converter Modes Control Register Description	870
29.11.9. DC-DC Miscellaneous Register Description	872
29.11.10. DC-DC Duty Cycle Limits Control Register Description	872
29.11.11. Converter Loop Behavior Control Register Description	873
29.11.12. Power Subsystem Status Register Description	875
29.11.13. Transistor Speed Control and Status Register Description	877
29.11.14. Battery-Level Monitor Register Description	878
29.11.15. Power Module Reset Register Description	879
29.11.16. Power Module Debug Register Description	880
29.11.17. Power Module Special Register Description	881
29.11.18. Power Module Version Register Description	881

30. SERIAL JTAG (SJTAG)	883
30.1. Overview	883
30.2. Operation	884
30.2.1. Debugger Async Start Phase	885
30.2.2. STMP3770 Timing Mark Phase	885
30.2.3. Debugger Send TDI, Mode Phase	885
30.2.4. STMP3770 Wait For Return Clock Phase	886
30.2.5. STMP3770 Sends TDO and Return Clock Timing Phase	886
30.2.6. STMP3770 Terminate Phase	886
30.2.7. SJTAG External Pin	887
30.2.8. Selecting Serial JTAG or Six-Wire JTAG Mode	887
31. BOOT MODES	889
31.1. Boot Modes	889
31.1.1. Boot Pins Definition and Mode Selection	889
31.1.2. Boot Mode Selection Map	890
31.2. OTP eFuse and Persistent Bit Definitions	891
31.2.1. OTP eFuse	891
31.2.2. Persistent Bits	893
31.3. Memory Map	893
31.4. General Boot Procedure	894
31.4.1. Preparing Bootable Images	894
31.4.2. Constructing Image to Be Loaded by Boot Loader	895
31.5. NOR Boot Mode	895
31.6. I ² C Boot Mode	895
31.7. SPI Boot Mode	896
31.7.1. Media Format	896
31.7.2. SSP	896
31.8. SD/MMC Boot Mode	897
31.9. NAND Boot Mode	899
31.9.1. NAND Control Block (NCB)	899
31.9.2. Expected NAND Layout	902
31.9.3. Typical NAND Page Organization	911
31.10. USB Boot Driver	913
31.10.1. Boot Loader Transaction Controller (BLTC)	913
31.10.2. Plug-in Transaction Controller (PITC)	914
31.10.3. USB IDs and Serial Number	914
31.10.4. USB Recovery Mode	914
32. PIN DESCRIPTIONS	915
32.1. Pin Definitions for 100-Pin BGA	916
32.2. Pin Definitions for 100-Pin LQFP	922
33. PIN CONTROL AND GPIO	929
33.1. Overview	929
33.2. Operation	929
33.2.1. Reset Configuration	930
33.2.2. Pin Interface Multiplexing	930
33.2.3. GPIO Interface	933
33.3. Behavior During Reset	937
33.4. Programmable Registers	938
33.4.1. PINCTRL Block Control Register Description	938
33.4.2. PINCTRL Pin Mux Select Register 0 Description	939
33.4.3. PINCTRL Pin Mux Select Register 1 Description	941
33.4.4. PINCTRL Pin Mux Select Register 2 Description	943
33.4.5. PINCTRL Pin Mux Select Register 3 Description	945
33.4.6. PINCTRL Pin Mux Select Register 4 Description	947
33.4.7. PINCTRL Pin Mux Select Register 5 Description	950
33.4.8. PINCTRL Pin Mux Select Register 6 Description	952
33.4.9. PINCTRL Pin Mux Select Register 7 Description	954
33.4.10. PINCTRL Drive Strength and Voltage Register 0 Description	955
33.4.11. PINCTRL Drive Strength and Voltage Register 1 Description	958
33.4.12. PINCTRL Drive Strength and Voltage Register 2 Description	960
33.4.13. PINCTRL Drive Strength and Voltage Register 3 Description	962

33.4.14. PINCTRL Drive Strength and Voltage Register 4 Description	964
33.4.15. PINCTRL Drive Strength and Voltage Register 5 Description	966
33.4.16. PINCTRL Drive Strength and Voltage Register 6 Description	968
33.4.17. PINCTRL Drive Strength and Voltage Register 7 Description	970
33.4.18. PINCTRL Drive Strength and Voltage Register 8 Description	972
33.4.19. PINCTRL Drive Strength and Voltage Register 9 Description	974
33.4.20. PINCTRL Drive Strength and Voltage Register 10 Description	976
33.4.21. PINCTRL Drive Strength and Voltage Register 11 Description	979
33.4.22. PINCTRL Drive Strength and Voltage Register 12 Description	981
33.4.23. PINCTRL Drive Strength and Voltage Register 13 Description	983
33.4.24. PINCTRL Drive Strength and Voltage Register 14 Description	985
33.4.25. PINCTRL Bank 0 Pullup Resistor Enable Register Description	987
33.4.26. PINCTRL Bank 1 Pullup Resistor Enable Register Description	988
33.4.27. PINCTRL Bank 2 Pullup Resistor Enable Register Description	989
33.4.28. PINCTRL Bank 3 Pad Keeper Disable Register Description	990
33.4.29. PINCTRL Bank 0 Data Output Register Description	991
33.4.30. PINCTRL Bank 1 Data Output Register Description	992
33.4.31. PINCTRL Bank 2 Data Output Register Description	992
33.4.32. PINCTRL Bank 0 Data Input Register Description	993
33.4.33. PINCTRL Bank 1 Data Input Register Description	993
33.4.34. PINCTRL Bank 2 Data Input Register Description	994
33.4.35. PINCTRL Bank 0 Data Output Enable Register Description	995
33.4.36. PINCTRL Bank 1 Data Output Enable Register Description	995
33.4.37. PINCTRL Bank 2 Data Output Enable Register Description	996
33.4.38. PINCTRL Bank 0 Interrupt Select Register Description	997
33.4.39. PINCTRL Bank 1 Interrupt Select Register Description	997
33.4.40. PINCTRL Bank 2 Interrupt Select Register Description	998
33.4.41. PINCTRL Bank 0 Interrupt Mask Register Description	999
33.4.42. PINCTRL Bank 1 Interrupt Mask Register Description	1000
33.4.43. PINCTRL Bank 2 Interrupt Mask Register Description	1000
33.4.44. PINCTRL Bank 0 Interrupt Level/Edge Register Description	1001
33.4.45. PINCTRL Bank 1 Interrupt Level/Edge Register Description	1002
33.4.46. PINCTRL Bank 2 Interrupt Level/Edge Register Description	1003
33.4.47. PINCTRL Bank 0 Interrupt Polarity Register Description	1003
33.4.48. PINCTRL Bank 1 Interrupt Polarity Register Description	1004
33.4.49. PINCTRL Bank 2 Interrupt Polarity Register Description	1005
33.4.50. PINCTRL Bank 0 Interrupt Status Register Description	1005
33.4.51. PINCTRL Bank 1 Interrupt Status Register Description	1006
33.4.52. PINCTRL Bank 2 Interrupt Status Register Description	1007
34. REGISTER MACRO USAGE	1009
34.1. Definitions	1009
34.2. Background	1009
34.3. Naming Convention	1010
34.4. Examples	1011
34.4.1. Setting 1-Bit Wide Field	1011
34.4.2. Clearing 1-Bit Wide Field	1011
34.4.3. Toggling 1-Bit Wide Field	1012
34.4.4. Modifying n-Bit Wide Field	1012
34.4.5. Modifying Multiple Fields	1012
34.4.6. Writing Entire Register (All Fields Updated at Once)	1012
34.4.7. Reading a Bit Field	1012
34.4.8. Reading Entire Register	1013
34.4.9. Accessing Multiple Instance Register	1013
34.4.10. Correct Way to Soft Reset a Block	1013
34.5. Summary Preferred	1013
34.6. Summary Alternate Syntax	1014
34.7. Assembly Example	1014
35. MEMORY MAP	1017
36. PACKAGE DRAWINGS	1019
36.1. 100-Pin Low-Profile Quad Flat Pack (LQFP)	1019
36.2. 100-Pin Ball Grid Array (BGA)	1020

37. STMP3770 PART NUMBERS AND ORDERING INFORMATION	1023
38. ACRONYMS AND ABBREVIATIONS	1025
INDEX: REGISTER NAMES	1029

LIST OF FIGURES

Figure 1.	System Block Diagram	30
Figure 2.	Physical Memory Map	33
Figure 3.	STMP3770 SOC Block Diagram	34
Figure 4.	Mixed Signal Audio Elements	39
Figure 5.	ARM926 RISC Processor Core	48
Figure 6.	ARM Programmable Registers	49
Figure 7.	Logical Diagram of STMP3770 Clock Domains	53
Figure 8.	Divide Range $2 < \text{DIV} < 2^n$, 3/8 Example Waveform	57
Figure 9.	Divide Range $1 < \text{DIV} < 2$, 3/4 Example Waveform	57
Figure 10.	Soft Reset Logic	59
Figure 11.	Interrupt Collector Diagram for IRQ Generation	75
Figure 12.	Interrupt Collector Bit “37” Logic	76
Figure 13.	IRQ Control Flow	78
Figure 14.	Nesting of Multi-Level IRQ Interrupts	79
Figure 15.	FIQ Generation Logic	80
Figure 16.	Default First-Level Page Table (DFLPT) Block Diagram	125
Figure 17.	DFLPT Virtual Memory Map	127
Figure 18.	Digital Control (DIGCTL) Block Diagram	129
Figure 19.	On-Chip RAM Partitioning	130
Figure 20.	On-Chip OTP (OCOTP) Controller Block Diagram	169
Figure 21.	OCOTP Allocation—Customer View	170
Figure 22.	USB 2.0 Device Controller Block Diagram	188
Figure 23.	USB 2.0 Check_USB_Plugged_In Flowchart	190
Figure 24.	USB 2.0 USB PHY Startup Flowchart	191
Figure 25.	USB 2.0 PHY PLL Suspend Flowchart	192
Figure 26.	UTMI Powerdown	192
Figure 27.	USB 2.0 PHY Block Diagram	249
Figure 28.	USB 2.0 PHY Analog Transceiver Block Diagram	251
Figure 29.	USB 2.0 PHY Transmitter Block Diagram	254
Figure 30.	AHB-to-APBH Bridge DMA Block Diagram	267
Figure 31.	AHB-to-APBH Bridge DMA Channel Command Structure	269
Figure 32.	AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command	273
Figure 33.	AHB-to-APBX Bridge DMA Block Diagram	335
Figure 34.	AHB-to-APBX Bridge DMA Channel Command Structure	337
Figure 35.	AHB-to-APBX Bridge DMA AUDIOOUT (DAC) Example Command Chain	340
Figure 36.	General-Purpose Media Interface Controller Block Diagram	398
Figure 37.	BASIC NAND Timing	399
Figure 38.	NAND Read Path Timing	401
Figure 39.	NAND Command and Address Timing Example	402
Figure 40.	Hardware 8-Symbol Correcting ECC Accelerator (ECC8) Block Diagram	420
Figure 41.	ECC-Protected 2K NAND Page Data—NAND Memory Footprint	422
Figure 42.	ECC-Protected 2K NAND Page Data—System Memory Footprint	423
Figure 43.	ECC-Protected 4K NAND Page Data—NAND Memory Footprint	424
Figure 44.	ECC-Protected 4K NAND Page Data—System Memory Footprint	425
Figure 45.	ECC8 Reed-Solomon Encode Flowchart	429
Figure 46.	ECC8 DMA Descriptor Legend	429
Figure 47.	ECC8 Reed-Solomon Encode DMA Descriptor Chain	430
Figure 48.	ECC8 Reed-Solomon Decode Flowchart	436
Figure 49.	ECC8 Reed-Solomon Block Coding—Decoder for $t=8$	437
Figure 50.	ECC8 Reed-Solomon Decode DMA Descriptor Chain	438
Figure 51.	Data Co-Processor (DCP) Block Diagram	457
Figure 52.	Cipher Block Chaining (CBC) Mode Encryption	462
Figure 53.	Cipher Block Chaining (CBC) Mode Decryption	463
Figure 54.	Supported Chroma Subsampling Modes	464
Figure 55.	DCP Arbitration	466

Figure 56.	DCP Work Packet Structure	469
Figure 57.	Basic Memory Copy Operation	478
Figure 58.	Basic Hash Operation	479
Figure 59.	Basic Cipher Operation	480
Figure 60.	Multi-Buffer Scatter/Gather Cipher and Hash Operation	482
Figure 61.	Synchronous Serial Port Block Diagram	525
Figure 62.	Motorola SPI Frame Format (Single Transfer) with POLARITY=0 and PHASE=0	528
Figure 63.	Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=0	528
Figure 64.	Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=1	529
Figure 65.	Motorola SPI Frame Format (Single Transfer) with POLARITY=1 and PHASE=0	530
Figure 66.	Motorola SPI Frame Format (Continuous Transfer) with POLARITY=1 and PHASE=0	530
Figure 67.	Motorola SPI Frame Format with POLARITY=1 and PHASE=1	531
Figure 68.	Texas Instruments Synchronous Serial Frame Format (Single Transfer)	532
Figure 69.	Texas Instruments Synchronous Serial Frame Format (Continuous Transfer)	532
Figure 70.	SD/MMC Block Transfer Flowchart	537
Figure 71.	Basic MS Protocols	539
Figure 72.	MS Operation Flowchart	542
Figure 73.	MS Four-State Read and Write	543
Figure 74.	Top-Level LCDIF Block Diagram	562
Figure 75.	LCDIF Write Path	564
Figure 76.	LCDIF Read Path	564
Figure 77.	8-Bit LCDIF Register Programming—Example A	565
Figure 78.	8-Bit LCDIF Register Programming—Example B	565
Figure 79.	16-Bit LCDIF Register Programming—Example A	566
Figure 80.	16-Bit LCDIF Register Programming—Example B	566
Figure 81.	LCD Interface Signals in Write Mode	567
Figure 82.	LCD Interface Signals in Read Mode	568
Figure 83.	LCD Interface Signals in DOTCLK Mode	572
Figure 84.	LCDIF Interface Signals in ITU-R BT.656 Digital Video Interface Mode	574
Figure 85.	Timers and Rotary Decoder Block Diagram	595
Figure 86.	Timer 0, Timer 1, or Timer 2 Detail	596
Figure 87.	Timer 3 Detail	598
Figure 88.	Pulse-Width Measurement Mode	599
Figure 89.	Detail of Rotary Decoder	600
Figure 90.	Rotary Decoding Mode—Debouncing Rotary A and B Inputs	601
Figure 91.	Rotary Decoding Mode—Input Transitions	602
Figure 92.	RTC, Watchdog, Alarm, and Persistent Bits Block Diagram	618
Figure 93.	RTC Initialization Sequence	619
Figure 94.	RTC Writing to a Master Register from CPU	620
Figure 95.	Pulse-Width Modulation Controller (PWM) Block Diagram	636
Figure 96.	PWM Output Example	637
Figure 97.	PWM Differential Output Pair Example	638
Figure 98.	PWM Output Driver	639
Figure 98.	PWM Output Driver	639
Figure 99.	Backlight Current Control	640
Figure 100.	I ² C Interface Block Diagram	654
Figure 101.	I ² C Data and Clock Timing	656
Figure 102.	I ² C Data and Clock Timing Generation	657
Figure 103.	I ² C Master Mode Flow Chart—Initial States	660
Figure 104.	I ² C Master Mode Flow Chart—Receive States	661
Figure 105.	I ² C Master Mode Flow Chart—Transmit States	662
Figure 106.	I ² C Master Mode Flow Chart—Send Stop States	663
Figure 107.	I ² C Slave Mode Flow Chart	665
Figure 108.	I ² C Writing Five Bytes	666
Figure 109.	I ² C Reading 256 Bytes from an EEPROM	667
Figure 110.	Application UART Block Diagram	686
Figure 111.	Application UART Character Frame	687

STMP3770

Figure 112. Debug UART Block Diagram	704
Figure 113. Debug UART Character Frame	705
Figure 114. AUDIOIN/ADC Block Diagram	720
Figure 115. Variable-Rate A/D Converter	724
Figure 116. External Microphone Bias Generation	725
Figure 117. Internal Microphone Bias Generation	726
Figure 118. AUDIOOUT/DAC Block Diagram	742
Figure 119. Stereo Sigma Delta D/A Converter	746
Figure 120. Conventional Stereo Headphone Application Circuit	748
Figure 121. Stereo Headphone Application Circuit with Common Node	748
Figure 122. Stereo Headphone Common Short Detection and Powerdown Circuit	749
Figure 123. Stereo Headphone L/R/ Short Detection and Powerdown Circuit	749
Figure 124. SPDIF Transmitter Block Diagram	776
Figure 125. SPDIF Flow Chart	777
Figure 126. SPDIF DMA Two-Block Transmit Example	779
Figure 127. Digital Radio Interface (DRI) Block Diagram	789
Figure 128. DRI Synchronization and Data Recovery	790
Figure 129. Digital Radio Interface (DRI) Framing	790
Figure 130. Digital Radio Interface (DRI) Digital Signals into Analog Line In	792
Figure 131. Low-Resolution ADC and Touch-Screen Interface Block Diagram	802
Figure 132. Low-Resolution ADC Successive Approximation Unit	806
Figure 133. Using Delay Channels to Oversample a Touch-Screen	807
Figure 134. 5-V Power Supply Block Diagram	844
Figure 135. Li-Ion DC-DC Supply Block Diagram	845
Figure 136. 1-Cell Alkaline/NIMH DC-DC Supply Block Diagram	846
Figure 137. Brownout Detection Flowchart	849
Figure 138. Power-Up, Power-Down and Reset Flow Chart	855
Figure 139. DC-DC Converter Efficiency	858
Figure 140. Serial JTAG (SJTAG) Block Diagram	883
Figure 141. SJTAG Clock Relationships	884
Figure 142. SJTAG Phases of Operation for One JTAG Clock	885
Figure 143. SJTAG Drivers	887
Figure 144. Boot Loader Memory Map	894
Figure 145. Creating a Boot Loader Image	895
Figure 146. FindBootControlBlocks Flowchart	900
Figure 147. Block Search Flowchart	901
Figure 148. Expected NAND Layout	902
Figure 149. NAND Layout—Multiple NANDs	904
Figure 150. Boot Image Recovery	905
Figure 151. Bad Block Search	906
Figure 152. DBBT Layout	907
Figure 153. 2K Page Layout in NAND	911
Figure 154. 2K Page Layout in On-Chip Memory	911
Figure 155. Redundant Area—2K	912
Figure 156. 4K Page in NAND	913
Figure 157. 4K Page Layout in On-Chip Memory	913
Figure 158. Pad Diagram	930
Figure 159. GPIO Output Setup Flowchart	934
Figure 160. GPIO Input Setup Flowchart	935
Figure 161. GPIO Interrupt Flowchart	936
Figure 162. GPIO Interrupt Generation	937
Figure 163. STMP3770 Detailed Memory Map	1017
Figure 164. 100-Pin Low-Profile Quad Flat Pack (LQFP) Package Drawing	1019
Figure 165. 100-Pin Ball Grid Array (BGA) Package Drawing, Part 1	1020
Figure 166. 100-Pin BGA Package Drawing, Part 2	1021

REVISION HISTORY

REVISION	DESCRIPTION
1.00	First Release - 18 January 2008.
1.01	Added 100-pin LQFP package information.
1.02	Updated pinouts for the 100-pin LQFP package.
1.03	Corrected default chip ID information.
1.04	Added details to GPMI drive strengths.

STMP3770



1. PRODUCT OVERVIEW

The STMP3770 is SigmaTel's fifth-generation system on chip (SOC) designed for portable consumer electronic products. The STMP3770 surpasses its predecessor, the STMP36xx, in performance, power savings, and integration. It is an ideal processor for applications such as MP3 players, portable media players, portable navigation devices and various other handheld multimedia devices that require low power, high performance, high integration and quality audio and video playback.

This chapter provides an general overview of the STMP3770 product and describes hardware features, application capability, design support, and additional documentation. See [Table 1](#), the pinout information in [Chapter 32, 'Pin Descriptions' on page 915](#), and [Table 1261, "Part Numbers for STMP3770 Family Members," on page 1023](#) for more detailed information about which functions described later in this document are supported in which package and part number.

Table 1. STMP3770 Functions

FUNCTION	LQFP100	BGA100
General-Purpose Media Interface (GPMI): • NAND data width • Parallel ATA	8-bit data No	8-bit data No
LCD Interface (LCDIF): • Data width • 8-bit serial DOTCLOCK mode	8-bit data Yes	8-bit data Yes
Serial Audio Interface (I ² S): • Interfaces supported	0	0
Low-Resolution ADC (LRADC): • Number supported • Touch-screen supported	2 No	2 No
Application UART: • Supported via dedicated pins? • Supported via multiplexed pins?	No Yes	No Yes
Synchronous Serial Port 1 (SSP1): • Data width	4-bit data	4-bit data
Stereo LineOut Interface	Yes	Yes

1.1. Hardware Features

- **ARM926 CPU Running at 205 MHz at 1.2 V and 320 MHz at 1.45 V**
 - ◆ Integrated ARM926EJ-S CPU
 - ◆ 16-Kbyte data cache and 16-Kbyte instruction cache
 - ◆ ARM Embedded Trace Macrocell (ETM) version 9-medium-plus
 - ◆ One-wire JTAG interface, cheap enough to include even on product boards
 - ◆ Resistor-less boot mode selection using integrated OTP e-fuse block
- **512 Kbytes of Integrated Low-Power On-Chip RAM**
- **64 Kbytes of Integrated Mask-Programmable On-Chip ROM**
- **1 Kbit of On-Chip One-Time-Programmable (OCOTP) ROM**
- **Universal Serial Bus (USB) High-Speed On-The-Go (OTG)—Up to 480 Mb/s**
 - ◆ High-speed USB device and host functions
 - ◆ Fully integrated high-speed OTG Physical Layer Protocol (PHY)
 - ◆ Complete OTG support
- **Power Management Unit**
 - ◆ Single DC-DC switched converter supports all common battery configurations.
 - ◆ Features multi-channel outputs for VDDIO (3.3 V), VDDD (1.2 V), VDDA (1.8 V)
 - ◆ Direct power from 5-V source (USB, wall power, or other source)
 - ◆ Silicon speed and temperature sensors enable adaptive power management over temperature and silicon process.
- **Optimized for Very Long Battery Life**
 - ◆ <25 mW system power consumption while playing 128-kbps MP3.
- **Audio Codec**
 - ◆ Stereo headphone DAC with 99 dB SNR
 - ◆ Stereo ADC with 85 dB SNR
 - ◆ Stereo headphone amplifier with direct drive to eliminate bulky capacitors
 - ◆ Amplifiers are designed for click/pop free operation and have short-circuit protection.
 - ◆ Two stereo line inputs
 - ◆ Microphone input
 - ◆ SPDIF digital out
 - ◆ Stereo line out (BGA100 package only) with 98 dB SNR (includes line-out and idle DAC SNR)
- **16-Channel Low-Resolution ADC**
 - ◆ 6 independent channels and 10 dedicated channels
 - ◆ Temperature sensor controller
 - ◆ Absolute accuracy of 1.3%
 - ◆ Up to 0.5% with bandgap calibration
- **Security Features**
 - ◆ Read-only unique ID for digital rights management algorithms
 - ◆ Secure boot using 128-bit AES hardware decryption
 - ◆ SHA-1 hashing hardware
 - ◆ Customer-programmed (OTP) 128 bit AES key is never visible to software.
- **Wide Assortment of External Media Interfaces**
 - ◆ Up to four NAND flash memories (BGA100) with hardware management of device interleaving
 - ◆ High-speed MMC, secure digital (SD), MS

- ◆ Hardware Reed-Solomon Error Correction Code (ECC) engine offers industry-leading protection and performance for NANDs.
- **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 Standard LCD Modules**
 - ◆ 8-bit bus (100-pin packages)
 - ◆ Read or write frame buffers in external LCD controller chips
 - ◆ Support for streaming video using either VSYNC or dot-clock display modes
 - ◆ Efficient support for 8-bit delta pixel mode LCD displays
 - ◆ Optional ITU-BT656 8-bit digital video output to an external NTSC/PAL decoder
- **Power-Efficient Direct-Drive LCD Backlight Controller with Voltage or Current Feedback**
 - ◆ 5-bit brightness adjustment
- **Data Co-Processor (DCP) Supports Memory Copy, Crypto, and Color-Space Conversion**
 - ◆ High-speed memory copy
 - ◆ Simple blit capability
 - ◆ Color-space conversion in hardware
 - ◆ Rotation in hardware
 - ◆ Scaling in hardware
- **Two Universal Asynchronous Receiver-Transmitters (UARTs)**
 - ◆ High-speed application UART operates up to 3.25 Mb/s with hardware flow control and dual DMA.
 - ◆ Debug UART operates at up to 115Kb/s using programmed I/O.
- **I²C Master/Slave**
 - ◆ DMA control of an entire EEPROM or other device read/write transaction without CPU intervention
- **Dual Synchronous Serial Ports (for SPI, MMC, SDIO, MS, CE-ATA, Triflash)**
 - ◆ 1-bit, 4-bit and 8-bit MMC/SD/SDIO/MSPPro modes
 - ◆ Compliant with SDIO Rev. 2.0
 - ◆ 4-bit and 8-bit CE-ATA hard drive connection
 - ◆ SPI up to 16 Mbits/s
 - ◆ Maximum SSP_CLK frequency is 80 MHz
 - ◆ Maximum SSP_CLK frequency in MMC/SD/SDIO/MS/CE-ATA modes is 40 MHz
 - ◆ Maximum SSP_CLK frequency in SPI/SSI modes with pF loads is 16 MHz.
- **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 optional low power crystal (32.768 kHz or 32.0 kHz), customer-selectable via OTP.
- **SPDIF Transmit**
- **Digital Radio Interface (DRI) with Clock and Data Inputs for Connection to SigmaTel's STFM1000 FM Stereo Receiver**
- **Customer-Programmable One-Time-Programmable (OTP) ROM via Integrated eFuse Block**
 - ◆ Resistorless boot mode selection

- ◆ 128-bit boot mode crypto key
- ◆ Boot mode specification of NAND characteristics for device that the customer is soldering to the board. This means no more costly delays waiting for new device support in the boot ROM.
- ◆ Fully software-programmable and accessible

■ Flexible I/O Pins

- ◆ All digital pins have drive-strength controls as described in [Section 33.2.2.1 on page 932](#).
- ◆ Almost all digital pins have general-purpose input/output (GPIO) mode.

■ Offered in 100-Pin Low-Profile Quad Flat Pack (LQFP), and 100-Pin Ball Grid Array (BGA)

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, MP3, or Nearly Any Other Format

■ Graphical Equalizer

■ Sound Effects and Spatialization

■ JPEG Image Decode and Encode

- ◆ Simultaneous JPEG decoding and compressed audio playback

■ Flexible USB Connectivity

- ◆ Mass storage device
- ◆ Media transfer protocol (MTP) 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 or USB Host

■ Ready for Bluetooth Using SDIO or UART

1.3. STMP3770 Product Features

The STMP3770 is SigmaTel's fifth generation single-chip digital media system for applications such as digital audio players, voice recorders, and cell phones. The STMP3770 offers long battery life, minimal external components, high processing performance, and excellent software development and debug support.

[Figure 1](#) shows a block diagram of a typical system based on the STMP3770.

The STMP3770 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, 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. The DC-DC converters and the clock generator can be reprogrammed on-the-fly to trade off power versus performance dynamically.

To provide the maximum application flexibility, the STMP3770 integrates a wide range of I/O ports. It can efficiently interface to nearly any type of flash memory, 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 (3.25 Mb/s) UART.

As with previous STMP3xxx products, the STMP3770 integrates the entire suite of analog components needed for a portable audio player. This includes a high-resolution audio codec with headphone amplifier, 16-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 512 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 video decoding. These and other advanced features are integrated into software development kits (SDKs) that support the STMP3770. Contact your local SigmaTel representative for more information on the software development kits available for the STMP3770.

Execution always begins in on-chip ROM after reset, unless overridden by the debugger. 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.

STMP3770

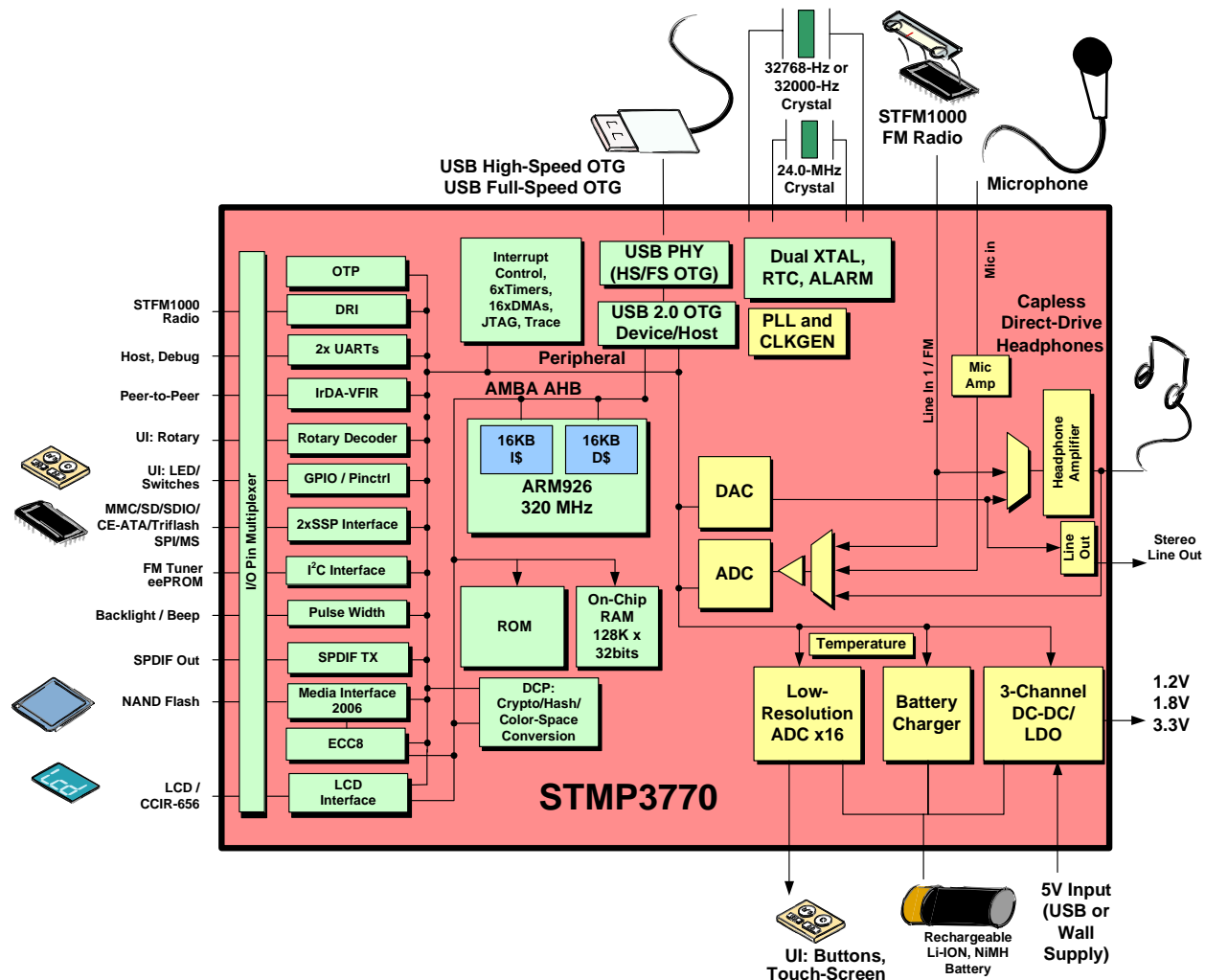


Figure 1. System Block Diagram

1.3.1. 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 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. The ARM CPU is described in [Chapter 3, “ARM CPU Complex”](#) on page 47.

The ARM RISC CPU is the central controller for the entire STMP3770 SOC, as shown in [Figure 3](#). The ARM 926 core includes two AHB masters:

- AHB1—Used for instruction fetches
- AHB2—Used for data load/stores, page table accesses, etc.

1.3.2. System Buses

The STMP3770 uses buses based on ARM's Advanced Microcontroller Bus Architecture (AMBA) for the on-chip peripherals. The AMBA specification (http://www.arm.com/products/solutions/AMBA_Spec.html) outlines two bus types: AHB and APB.

- AHB is a higher-performance bus that supports multiple masters such as the CPU and DMA controllers.
- The APB is a lower-speed peripheral bus.

As shown in [Figure 3](#), the STMP3770 uses a four-layer AHB and two APBs: APBH and APBX. The APB buses are enhanced to include byte-write capability.

1.3.2.1. AHB Bus

The AHB is the main high-performance system bus and is implemented in four layers, as follows:

- Layer 0 (AHB0)—Hardware ECC (ECC8) and DCP crypto/memory-copy masters
- Layer 1 (AHB1)—CPU instruction access to OCRAM, OCROM
- Layer 2 (AHB2)—CPU data access to OCRAM, OCROM, all bridges, USB and DFLPT slaves
- Layer 3 (AHB3)—APBH DMA, APBX DMA, and USB masters

The ARM926 instruction bus (AHB1) is a single-master layer, as is the ARM926 data bus (AHB2). The other two layers have multiple masters, as shown in [Figure 3](#).

The ARM926 data bus connects to the all slaves in the system, including RAMs, ROMs, bridge slaves, and USB slaves. The APB peripherals can act as AHB slaves through the AHB-APB bridge. The AHB has eight slaves:

- USB slave
- On-chip RAM
- On-chip ROM
- Default first-level page table
- Two APB bridges

Each layer of the AHB bus allows one active transaction at a time. A transaction is initiated by a master, controlled by an arbiter, and serviced by the slave at the corresponding address. A transaction can be as short as a single byte, or as long as a CPU cache line (32 bytes). For the USB, a transaction can be much longer, up to 512 bytes on its AHB layer. For more information, refer to the AMBA 2.0 specification.

1.3.2.2. APB Buses

There are two APB peripheral buses on the STMP3770:

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

The "H" in APBH denotes that the APBH is synchronous to HCLK, as compared to APBX, which runs on the crystal-derived XCLK. [Figure 3](#) shows which blocks are controlled by which bus. See [Section 1.3.7, "DMA Controller" on page 35](#) for more information about these peripheral buses and their DMA bridges.

1.3.3. On-Chip RAM and ROM

The STMP3770 includes 128Kx32-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 129](#).

The STMP3770 also includes 16Kx32-bit words of on-chip mask-programmable 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.

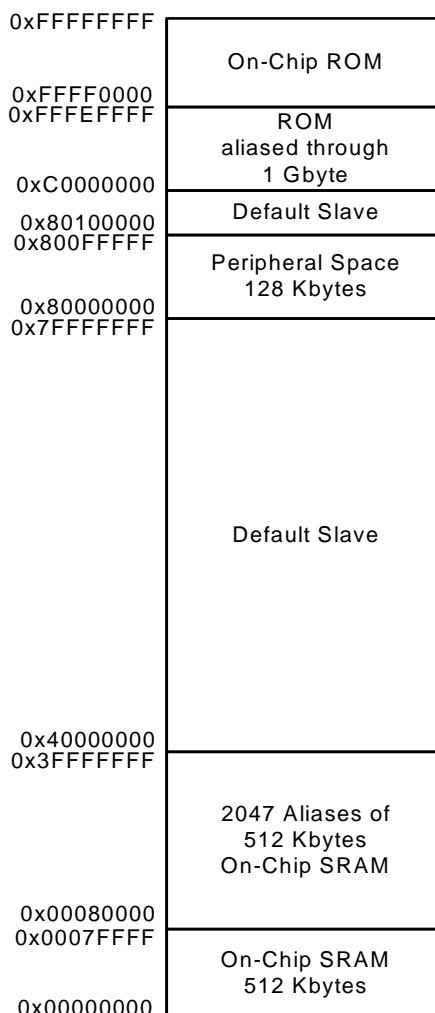
Other boot modes are responsible for loading application code from off-chip into the on-chip RAM. It supports initial program loading from a number of sources:

- NAND flash devices
- I²C 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.

[Figure 2](#) shows the memory map for the AHB2 devices.

**Figure 2. Physical Memory Map****1.3.4. On-Chip One-Time-Programmable (OCOTP) ROM**

The STMP3770 contains 1024 bits (1Kb) of OTP ROM. The OTP is segmented into four distinct physical banks. Each bank is further divided logically into eight 32-bit words. The OTP serves several functions:

- Housing of hardware and software capability bits (copied into shadow registers)
- Housing of SigmaTel operations and unique-ID fields.
- Housing the customer-programmable cryptography key
- Four words for customer general use
- A 32-bit word is dedicated to controller read and write locking of the various OTP regions (copied into a shadow register)
- Storage of various ROM configuration bits

Access to the OTP is done through a memory-mapped APBH slave interface. Each of the 32 words is memory-mapped on APBH for the purposes of reading (requires a bank-opening sequence). Writing to the OTP is done through an address and data interface, where software provides the OTP word number (one of 32) and a programming mask. For more information, see [Chapter 8, 'On-Chip OTP \(OCOTP\) Controller'](#) on page 169.

STMP3770

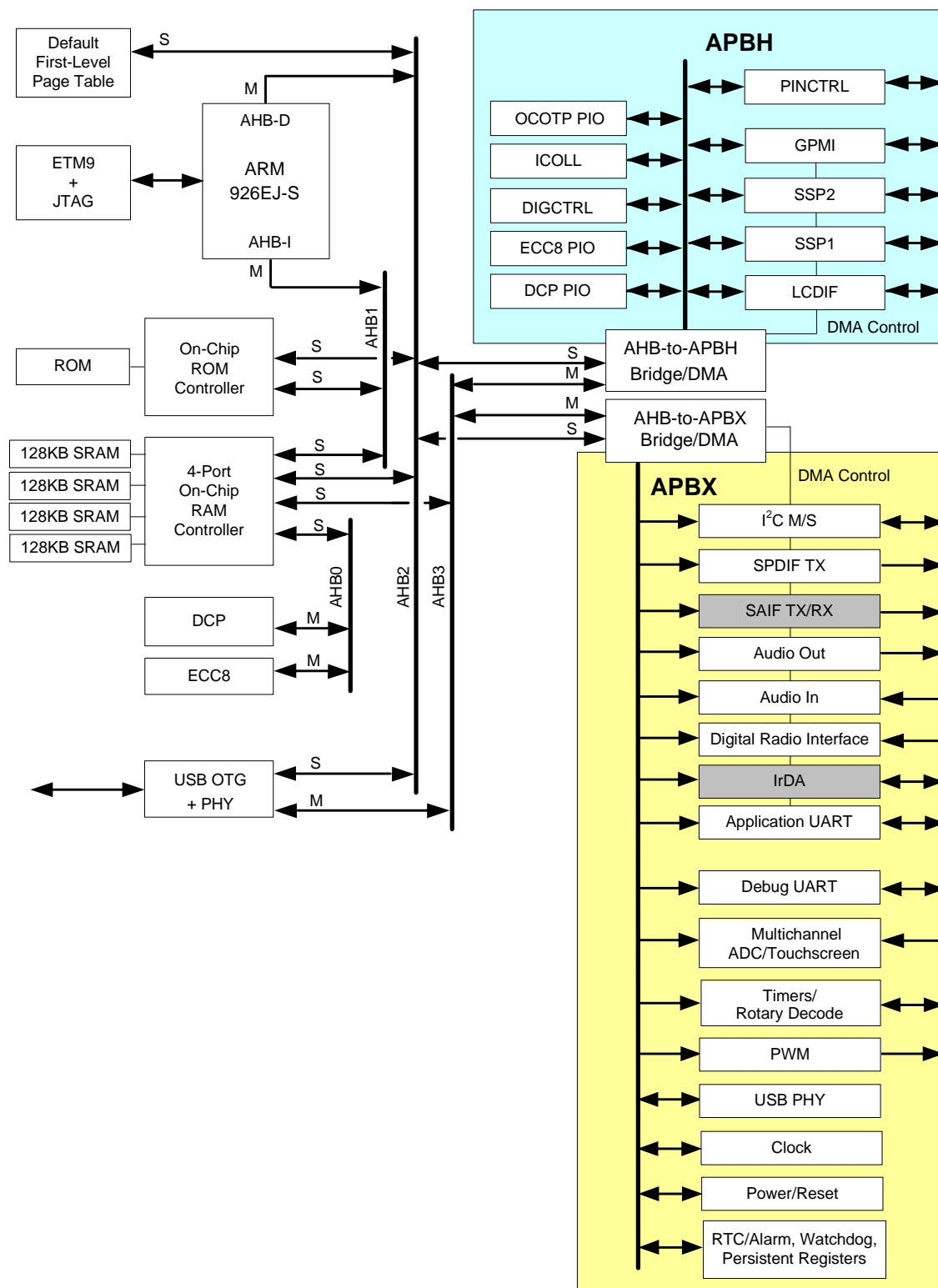


Figure 3. STMP3770 SOC Block Diagram

1.3.5. *Interrupt Collector*

The STMP3770 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 75](#).

1.3.6. *Default First-Level Page Table*

The STMP3770 contains a default first-level page table implemented as an AHB slave. This device provides an economical way to present 16 Kbytes of L1 page table entry data to the ARM CPU's MMU. The default first-level page table provides semi-configurable access to the PIO block at 0x80000000, as well as eight 32-bit fully programmable L1 PTE descriptors that can be allocated to any of the 4K available entries (except for the PIO entry).

This feature is described more completely in [Chapter 6, "Default First-level Page Table \(DFLPT\)" on page 125](#).

1.3.7. *DMA Controller*

Many peripherals on the STMP3770 use 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.

Similarly, both the ECC8 error correction engine and the DCP (crypto/memcpy) devices contain their own AHB bus masters to allow for more random accesses to system memory.

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.

As mentioned previously, there are two AMBA peripheral buses on the STMP3770:

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

See [Chapter 11, "AHB-to-APBH Bridge with DMA" on page 267](#), and [Chapter 12, "AHB-to-APBX Bridge with DMA" on page 335](#), for more detailed information. Note that the AHB HCLK can run up to 133 MHz.

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.

1.3.8. Clock Generation Subsystem

The STMP3770 uses several different clock domains to provide clocks to the various subsystems, as shown in [Figure 7 on page 53](#). These clocks are either derived from the 24-MHz crystal or from the integrated high-speed PLL. The PLL output is fixed at 480 MHz.

More details about the system clock architecture can be found in [Chapter 4, “Clock Generation and Control” on page 51](#).

The system includes a real-time clock that can use either the 24-MHz system crystal or an optional low power crystal oscillator running at either 32.768 kHz or 32.0 kHz (customer-configurable via OTP). An integrated watchdog reset timer is also available for automatic recovery from errant code execution. See [Chapter 19, “Real-Time Clock, Alarm, Watchdog, Persistent Bits” on page 617](#) for more information about these features.

1.3.9. Power Management Unit

The STMP3770 contains a sophisticated power management unit (PMU), including an integrated DC-DC converter and three linear regulators. The PMU can operate from a battery (1-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 brownout detection, as well as on-chip temperature, digital speed, and process monitoring.

The chip includes an integrated DC-DC converter that can be used to provide programmable power for the device as well as the entire application on up to three rails, Vddio, Vddd and Vdda. The converter 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 29, “Power Supply” on page 843](#).

1.3.10. 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 STMP3770 device interface can be attached to USB 2.0 hosts and hubs running in the USB 2.0 high-speed mode at 480 Mbits per second. It can be attached to USB 2.0 full-speed interfaces at 12 Mbits per second.

The USB controller and integrated PHY support high-speed OTG modes for peer-to-peer file interchange. The STMP3770 has a high-current PWM channel that can be used with low-cost external components to generate up to 8 mA 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 STMP3770. 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 five on-chip physical endpoints.

It is a dynamically configured port that can support up to five 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 9, “USB High-Speed On-the-Go \(Host/Device\) Controller” on page 187](#) and [Chapter 10, “Integrated USB 2.0 PHY” on page 249](#) for more information.

1.3.11. General-Purpose Media Interface (GPMI)

The chip includes a general-purpose media interface (GPMI) controller that supports NAND devices (all packages).

The NAND flash interface provides a state machine that provides all of the logic necessary to perform DMA functions between on-chip 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 data bus among four NAND devices without detailed CPU intervention. This allows the STMP3770 to provide unprecedented levels of NAND performance.

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. For NAND MP3 player applications, these might be all NAND flash devices.

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 397](#) for more information.

1.3.12. Hardware Acceleration for ECC for Robust External Storage

The forward error correction circuit (ECC) is used to provide STMP3770 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:

- Four-symbol error correcting (9 bits/symbol) Reed-Solomon encoder/decoder
- Eight-symbol error correcting (9 bits/symbol) Reed-Solomon encoder/decoder

The Reed-Solomon modes are used for storage elements 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 in a 2-Kbyte paged device or up to eight 9-bit symbols over a 512-byte block in a 4-Kbyte paged device.

In addition, the STMP3770 ECC8 engine provides significantly higher levels of performance and software convenience than was available in the STMP36xx. Both of these error correction encoder/decoders use DMA transfers to move data from system memory (on-chip RAM) completely in parallel with the CPU performing other useful work.

For storage read transfers, the ECC8 controller uses its AHB bus master to transfer data directly to system memory at up to four times faster than the performance seen on the STMP36xx HWECC engine. In addition, the ECC8 automatically corrects errors in the read data buffers in system memory without CPU assistance.

The ECC8 includes one more significant enhancement, namely, it provides four-symbol error correction for the 9 or 16 byte metadata stored in the redundant area of the NAND device. On the STMP36xx, this metadata was protected as part of the last 512-byte block in the page. This yielded long correction times for the tiny metadata area.

See [Chapter 14, “8-Symbol Correcting ECC Accelerator \(ECC8\)” on page 419](#) for more information.

1.3.13. **Data Co-Processor (DCP)—Memory Copy, Crypto, and Color-Space Converter**

The STMP3770 SOC contains a data co-processor consisting of four virtual channels. Each channel is essentially a memory-to-memory copy engine. The linked list control structure can be used to move byte-aligned blocks of data from a source to a destination. In the process of copying from one place to another, the DCP can be programmed to encrypt or decrypt the block using AES-128 in one of several chaining modes. An SHA-1 hash can be calculated as part of the memory-copy operation.

In addition, the DCP can be programmed to perform a color-space conversion from one of several formats in the source buffer to one of several formats in the destination buffer. In particular, the source buffer can be 4:2:0 YCbCr/YUV in planar format or a 4:4:4 16 bit RGB frame buffer format for display on an LCD panel. Another interesting output format can produce a destination frame buffer suitable for 8-bit delta pixel LCD displays. The DCP also supports scaling and rotation.

See [Chapter 15, “Data Co-Processor \(DCP\)” on page 457](#) for more information.

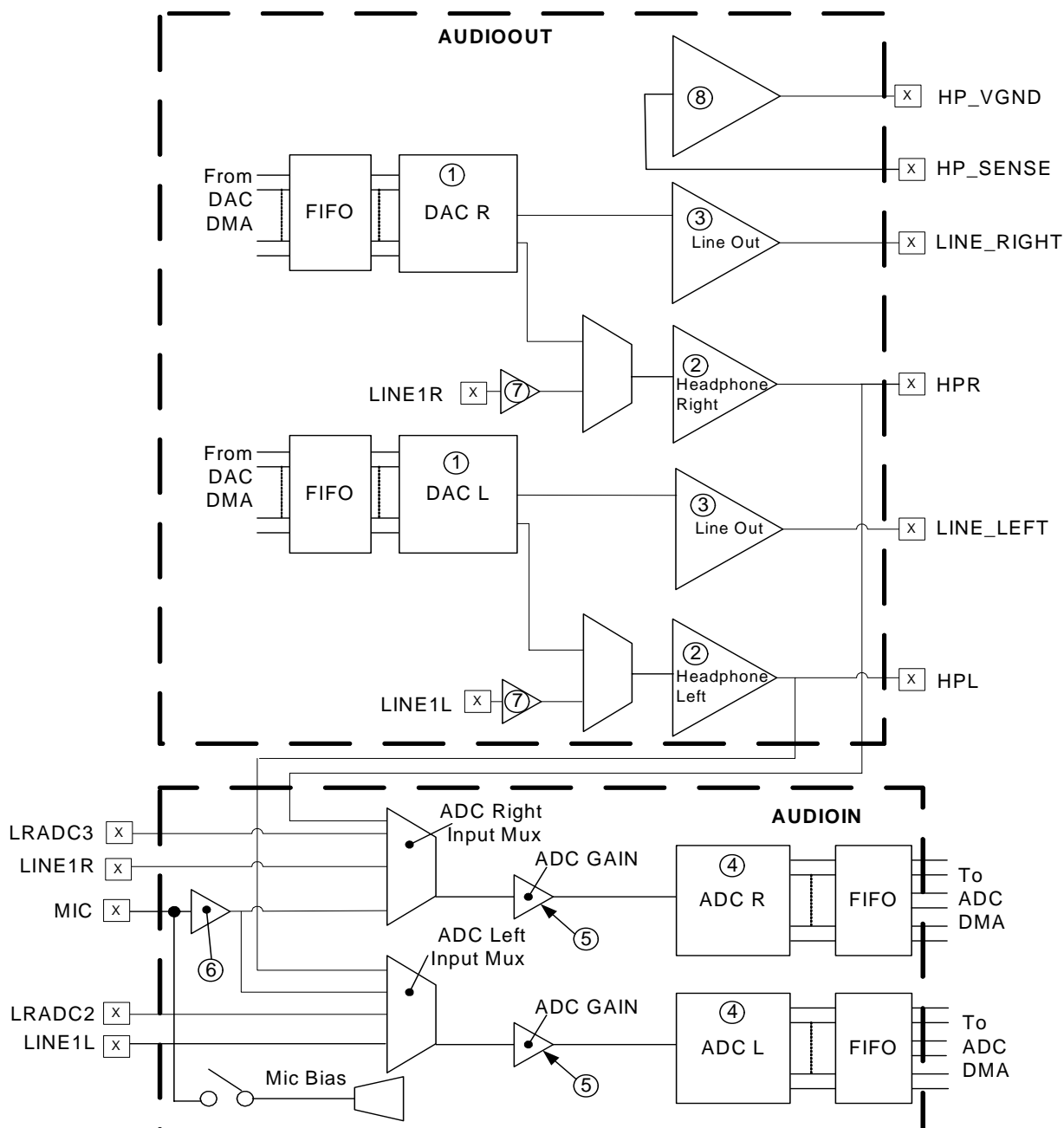
1.3.14. **Mixed Signal Audio Subsystem**

The STMP3770 contains an integrated high-quality mixed signal audio subsystem, including high-quality sigma delta D/A and A/D converters, as shown in [Figure 4](#). 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 (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 interrupt collector. 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 24, “AUDIOIN/ADC” on page 719](#), and [Chapter 25, “AUDIOOUT/DAC” on page 741](#).

**Notes:**

1. HW_AUDIOOUT_DACVOLUME: Digital volume control. 0 dB to -100 dB in 0.5 dB steps.
2. HW_AUDIOOUT_HPVOL: Analog volume control. 6 dB to -57.5 dB in 0.5 dB steps.
3. HW_AUDIOOUT_LINEOUTCTRL: Analog control for LineOut block. 0 dB to -4 dB in 25 mV steps. -4 dB to -34 dB in 2 dB steps.
4. HW_AUDIOIN_ADCVOLUME: Digital volume control. 0 dB to -100 dB in 0.5 dB steps.
5. HW_AUDIOIN_ADCVOL: Analog volume control that controls the ADC gain block. 0 dB to 22.5 dB gain in 1.5 dB steps.
6. HW_AUDIOIN_MICLINE_MICGAIN: Analog volume control that controls the microphone amplifier. 0, 20, 30, 40 dB gain.
7. HW_AUDIOIN_MICLINE_ATTEN_LINE
8. HW_AUDIOOUT_PWDN: Enable capless headphone common amplifier.

Figure 4. Mixed Signal Audio Elements

1.3.15. 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. It also provides several security features, including an entropy register, as well as the JTAG shield trust zone controls. See [Chapter 7, 'Digital Control and On-Chip RAM'](#) on page 129 for more information.

1.3.16. Synchronous Serial Port (SSP)

The STMP3770 SOC contains two integrated synchronous serial ports, SSPs. Each SSP supports a wide range of synchronous serial interfaces, including:

- 1-bit, 4-bit, or 8-bit high-speed MMC/SD/SDIO
- 4-bit or 8-bit CE-ATA mode.
- SPI
- 1-bit MS mode
- TI SSI

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

See [Chapter 16, "Synchronous Serial Ports \(SSP\)"](#) on page 525 for more information about these features.

1.3.17. I²C Interface

The chip contains a two-wire SMB/I²C bus interface. It can act as either a slave or master on the SMB interface. The on-chip ROM supports boot operations from I²C mastered EEPROMs, as well as slave I²C boot mode. See [Chapter 21, "I²C Interface"](#) on page 653 for more information.

1.3.18. General-Purpose Input/Output (GPIO)

The STMP3770 contains 51 GPIO pins in the 100-pin package. Most digital pins that are available for specific functions are also available as GPIO pins if they are not otherwise used in a particular application. See [Chapter 33, 'Pin Control and GPIO'](#) on page 929, for more information

1.3.19. LCD Interface (LCDIF)

The LCD interface has a dedicated DMA channel and can be used to transfer data directly to 8-bit LCD modules (100-pin package). It has programmable pin timing, support for 8080 and 6800 modes and automatically handshakes transfers to the LCD.

New features for the STMP3770 SOC include the ability to read the frame buffer contents back under DMA control. In addition, LCD with VSYNC or dot clock video streaming modes can be supported. Finally, the LCD interface can be used to output an 8-bit, 27-MHz ITU-BT656 digital video interface to an external NTSC/PAL encoder. See [Chapter 17, "LCD Interface \(LCDIF\)"](#) on page 561 for more information.

1.3.20. *SPDIF Transmitter*

The STMP3770 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 26, “SPDIF Transmitter” on page 775](#) for more information.

1.3.21. *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 595](#) for more information.

1.3.22. *Dual UARTs*

Each of two UARTs, similar to a 16550 UART, are provided—one for application use and one for debug use. The application UART is a high-speed device capable of running up to 3.25 Mbits per second with 16-byte receive and transmit FIFOs. The application UART supports DMA and flow control (CTS/RTS). The debug UART does not use DMA channels.

See [Chapter 22, “Application UART” on page 685](#), and [Chapter 23, “Debug UART” on page 703](#) for more information.

1.3.23. *Low-Resolution ADC, Touch-Screen Interface, and Temperature Sensor*

The LRADC provides 16 “physical” channels of 12-bit resolution analog-to-digital conversion. Only 8 “virtual” channels can be used at one time, but those 8 channels can be mapped to any of the 16 physical channels. Some physical channels have dedicated inputs:

- Channel 15—VDD5V
- Channel 14—Bandgap reference
- Channel 13—USB_DN
- Channel 12—USB_DP
- Channel 10 and 11—Reserved
- Channel 8 and 9—Internal temperature sensing
- Channel 7—Battery
- Channel 6—VDDIO

The USB_DN/DP inputs can only be sampled with the LRADC in non-USB mode (see HW_USBPHY_CTRL_DATA_ON_LRADC).

The remaining six channels are available for other uses (only channels 0 and 1 are available in the 100-pin packages) 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. The LRADC provides typical performance of 12-bit no-missing-codes, 9-bit SNR, and 1% absolute accuracy (limited by the bandgap reference).

See [Chapter 28, “Low-Resolution ADC and Touch-Screen Interface” on page 801](#) for more information.

STMP3770



1.3.24. *Pulse Width Modulator (PWM) Controller*

The STMP3770 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 (EL) display backlights. Independent output control of each phase allows 0, 1, or high impedance 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 635](#) for more information.

2. CHARACTERISTICS AND SPECIFICATIONS

This chapter describes the characteristics and specifications of the STMP3770 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)	−0.3	4.242	V
5-Volt Source Pin (VDD5V)	−0.3	5.25	V
PSWITCH—DC-DC Mode 1 (Note 1)	−0.3	BATT	V
PSWITCH—DC-DC Mode 0 (Notes 1 and 2)	−0.3	BATT/2	V
Analog Supply Voltage—VDDA1/VDDIO18	−0.3	2.10	V
Digital Supply Voltage	−0.3	1.45	V
I/O Supply—VDDIO1, VDDIO2, VDDIO3, VDDIO4	−0.3	3.63	V
DC-DC Converter—DCDC_VDDD	−0.3	1.45	V
DC-DC Converter—DCDC_VDDA	−0.3	1.98	V
DC-DC Converter—DCDC_VDDIO	−0.3	3.63	V
DC-DC Converter—DCDC1_BATT	−0.3	max (VDDIO, BATT)	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_DP and USB_DN 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. VDDIO can be applied to PSWITCH through a 10kΩ resistor. This is necessary in order to enter the chip's firmware recovery mode. (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.

2.2. Recommended Operating Conditions

Table 3. Recommended Operating Conditions

PARAMETER	MIN	TYP	MAX	UNITS
Ambient Operating Temperature (Note 1)	-10		70	°C
Analog Core Supply Voltage—VDDA1/VDDIO18	1.62	-	2.10	V
Digital Core Supply Voltage—VDDD1, VDDD2, VDDD3. Specification dependent on frequency. (Note 2)	1.075	-	1.45	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	-	2.9	-	V
DC-DC Mode 1	-	0.9	-	V
Standby Current (Note 3):				
DC-DC Mode 0 (32-kHz RTC off), BATT = 4.2 V	-	22		μA
DC-DC Mode 0 (32-kHz RTC on), BATT = 4.2 V	-	24		μA
DC-DC Mode 1 (32-kHz RTC off), BATT = 1.6 V	-	10		μA
DC-DC Mode 1 (32-kHz RTC on), BATT = 1.6 V	-	12		μA
Microphone:				
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 (16 Ω load)	-	-75	-	dB
Input Resistance (Line-to-Headphone mode) (Note 5)	-	50	-	kΩ
Input Resistance (Line-to-ADC mode)		30		kΩ
LineIn-to-HP SNR Idle Channel (Note 6)	96	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, 16 Ω load)	-	0.54	-	Vrms
Output Resistance	-	-	<1	Ω
THD+N (16 Ω load)	-	-	-66	dB
THD+N (10 kΩ load)	-	-84	-	dB
DAC SNR Idle Channel (Note 6)	-	99	-	dB
DAC -60 dB Dynamic Range (Note 6)	96	99	-	dB
Stereo LineOut:				
Full-Scale Output Voltage (VDDIO = 3.3 V, 10 kΩ load)	-	1.0	-	Vrms
THD+N (10 kΩ load, -3 dB signal) (Note 6)		-88		dB
SNR Idle Channel (10 kΩ load) (Note 6)		98		dB

Notes:

- Contact SigmaTel for extended temperature range options. In most systems 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 STMP3770.
- For optimum USB jitter performance, VDDD=1.3 V.
- 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.
- 1 Vrms requires external resistor divider.
- Input resistance changes with volume setting: 20KΩ at +12 dB, 50KΩ at 0 dB, 100KΩ at -34.5 dB.
- Measured "A weighted" over a 20-Hz to a 20-kHz bandwidth, relative to full scale output voltage (when VDDIO = 3.3 V and VDDA = 2.1 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](#), and [Table 6](#) for the specific clock targets. Measured supply voltage may not match the programmed value; see [Table 1111](#), “HW_POWER_VDDDCTRL Bit Field Descriptions,” on page 866 for more information.

Table 4. Recommended Operating Conditions for Specific CPUCLK Targets

CPUCLK Range (MHz)	HW_CLKCTRL_FRAC_CPUFRAC Allowable Range	Min VDDD Target Voltage (V)	HW_POWER_VDDCTRL_TRG (Using DC-DC Converters)	Corresponding VDDD Brownout Voltage	HW_POWER_VDDCTRL_BO_OFFSET
Up to 99.31	0x1B–0x23	1.050	0x0A	0.975	0x03
Up to 205.71	0x12–0x23 (full range)	1.200	0x10	1.100	0x04
Up to 261.82	0x12–0x23 (full range)	1.300	0x14	1.200	0x04
Up to 320	0x12–0x23 (full range)	1.450	0x1A	1.350	0x04

Table 5. Recommended Operating Conditions for Specific HCLK Targets

HCLK Range (MHz)	HW_CLKCTRL_FRAC_CPUFRAC Allowable Range	Min VDDD Target Voltage (V)	HW_POWER_VDDCTRL_TRG (Using DC-DC Converters)	Corresponding VDDD Brownout Voltage	HW_POWER_VDDCTRL_BO_OFFSET
Up to 99.31	0x1B–0x23	1.050	0x0A	0.975	0x03
Up to 144	0x12–0x23 (full range)	1.200	0x10	1.100	0x04
Up to 196.36	0x12–0x23 (full range)	1.300	0x14	1.200	0x04
Up to 205.71	0x12–0x23 (full range)	1.450	0x1A	1.350	0x04

Table 6. Recommended Operating Conditions for PLL Fractional Divider Clock Targets

Min VDDD Target Voltage (V)	HW_POWER_VDDCTRL_TRG	Corresponding VDDD Brownout Voltage	HW_POWER_VDDCTRL_BO_OFFSET	HW_CLKCTRL_FRA_C_PIXFRAC Allowable Range for ref_pix Target	HW_CLKCTRL_FRAC_IOFRAC Allowable Range for ref_io Target	HW_CLKCTRL_FRAC_CPUFRAC Allowable Range for ref_cpu Target
1.050	0x0A	0.975	0x03	0x1D–0x23 (246.86–297.93 MHz)	0x1B–0x23 (246.86–320 MHz)	0x1B–0x23 (246.86–320 MHz)
1.200	0x10	1.100	0x04	0x12–0x23 (246.86–480 MHz)	0x12–0x23 (246.86–480 MHz)	0x12–0x23 (246.86–480 MHz)

Notes:

- When SAIF is used in conjunction with the PLL as the clock source, HW_POWER_VDDCTRL_TRG must be set at 0x10 (1.2 V) and BO_OFFSET at 0x4 (1.1 V).

STMP3770

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 7. DC Characteristics

PARAMETER	MIN	TYP	MAX	UNITS
Power Dissipation: VDDD = 1.08 V, VDDA = 2.1 V, VDDIO = 2.8 V, DC-DC_MODE = 00 (Li-Ion H), VDDD brownout = NA, CPUCLK = 24 MHz, HCLK = 24 MHz, PLL off, USB off, LCDIF off, LRADC on, DCP on, TimRot on, DFLPT on, I\$/D\$ on, DAC on, ADC off, Application = AAC Playback from NAND, minimum power configuration selected.		20		mW
V _{iH} (DIO3)—Input high voltage for DIO3 digital I/O pin set except DEBUG, GPML_RDY2, I2C_SCL, I2C_SDA, ROTARYB, UART2_RX in 3.3-V mode.	2.0			V
V _{iH} (DIO3)—Input high voltage for DEBUG, GPML_RDY2, I2C_SCL, I2C_SDA, ROTARYB, UART2_RX in 3.3-V mode.	2.2			V
V _{IL} (DIO3)—Input low voltage for DIO3 digital I/O pin set in 1.8-V and 3.3-V modes.			0.8	V
V _{OH} (DIO3)—Output high voltage for DIO3 digital I/O pin set in 1.8-V and 3.3-V modes (I _{OH} = -0.1 mA).	0.8*VDDIO			V
V _{OL} (DIO3)—Output low voltage for DIO3 digital I/O pin set in 1.8-V and 3.3-V modes (I _{OL} = 0.1 mA).			0.2*VDDIO	V

Table 8. Drive Strengths Per Pad Type

PAD TYPE ¹	DRIVE STRENGTH ²	3.3-V PULLUP	3.3-V PULLDOWN	1.8-V PULLUP	1.8-V PULLDOWN
Regular I/O Address I/O	4 mA	3.6 mA	3.6 mA	3.0 mA	3.2 mA
	8 mA	7.2 mA	7.2 mA	7.8 mA	7.2 mA
	12 mA	10.8 mA	10.8 mA	11.7 mA	10.8 mA
High-Drive I/O: PWM4 ³	4 mA	3.6 mA	3.6 mA	4.1 mA	3.6 mA
	16 mA	14.4 mA	14.4 mA	16.4 mA	14.4 mA
	20 mA	18.0 mA	18.0 mA	20.5 mA	18.0 mA
SSP1_SCK, GPML_WRN, GPML_RDN, GPML_IRQ/SSP2_SCK	4 mA				
	8 mA	8.0 mA	7.8 mA	7.6 mA	7.8 mA
	12 mA				
	16 mA	15.9 mA	15.6 mA	15.2 mA	15.6 mA

Notes:

1. This table contains preliminary information. It shows minimum current sink/source capacity, measured as follows: Worst case—Process = SS, VDDIO33 = 3.15 V, VDDIO18 = 1.7 V, TEMP = 100 @V_{ds} = 0.4 V
2. The stronger the driver mode, the noisier the on-chip power supply. The use of a stronger drive mode must be limited to only a few pins. The majority of GPIO drivers must be set in 4-mA mode.
3. High-drive I/O has a high current source/sink capability. However, it is not meant as high-speed I/O. For AC, the driver turns on slowly to reduce L*di/dt power supply noise.

3. ARM CPU COMPLEX

This chapter describes the ARM CPU included on the STMP3770 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 5](#).

See http://www.arm.com/documentation/ARMProcessor_Cores/index.html to download the following ARM documentation on the ARM926EJ-S core:

- 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 6](#). 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.

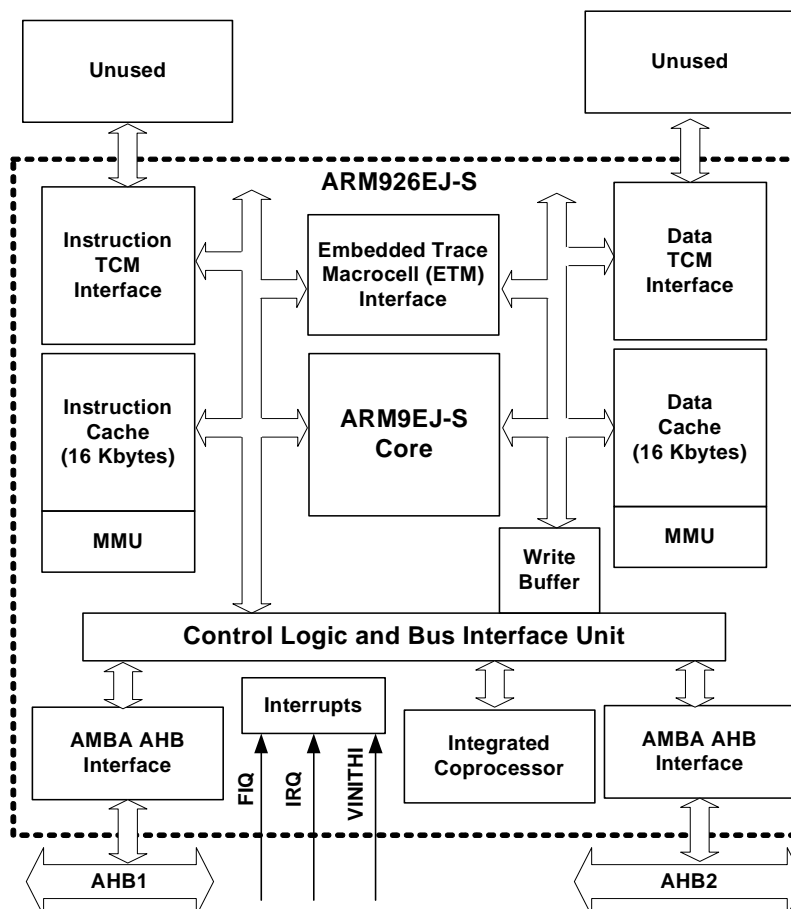


Figure 5. ARM926 RISC Processor Core

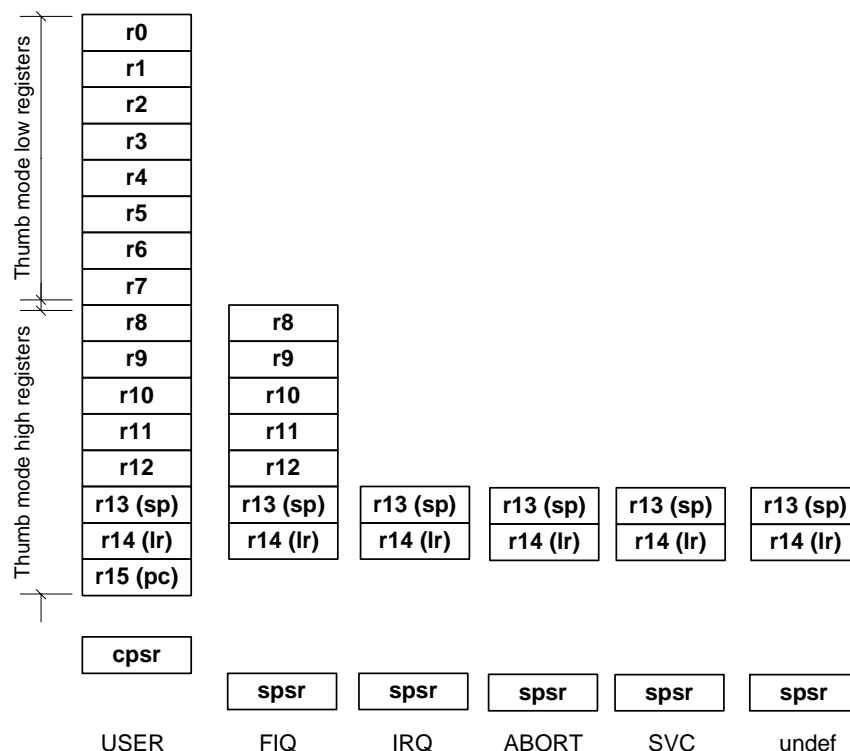
The following exceptions are recognized by the core:

- 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 STMP3770 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 a 16-Kbyte instruction cache and 16-Kbyte data cache and has two master interfaces to the AMBA AHB, as shown in [Figure 5](#).

The STMP3770 always operates in little endian mode.

**Figure 6. ARM Programmable Registers**

3.2. JTAG Debugger

The TAP controller of the ARM core in the STMP3770 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 bitstream 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

STMP3770



3.3. Embedded Trace Macrocell (ETM) Interface

The STMP3770 includes an ARM ETM-9 trace module implementing a small mode trace buffer. See the pin list in [Chapter 32](#) for the pinout of trace information.

4. CLOCK GENERATION AND CONTROL

This chapter describes the clock generation and control features of the STMP3770. The programmable registers are described in [Section 4.8](#).

Note: Clocks names with the format CLK_<NAME> in this chapter are equivalent to <NAME>CLK references throughout the rest of this data sheet. For example, CLK_H is equivalent to HCLK.

4.1. Overview

The STMP3770 clock architecture is designed to offer high performance, low power, and efficient software power management. The STMP3770 has up to three clock sources (two crystals and a phase-locked loop (PLL)) that are distributed to all 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.

The clock architecture is similar to that used on the STMP36xx. The ARM CPU is provided a processor clock (CPUCLK, also called P_CLK) and a synchronization signal that is edge-synchronous with HCLK, the APB clock. HCLK always runs at some fraction of the CPUCLK. The lower HCLK rates help reduce power consumption and allow the AHB peripherals to meet timing even when the CPU and cache are running at a higher frequency (e.g., the CPU can run at 200 MHz with the AHB at 100 MHz). Additional features include the following:

- Five individual phase fractional dividers (PFD) produce independent reference clock domains.
- Each PFD can be selected to divide the PLL by $PLL_freq * (18/n)$ where:
 $18 \leq n \leq 35$
- Programmable divide values adjust system performance to achieve application requirements with minimal power consumption.
- An on-chip ring oscillator running at approximately 24 MHz allows for low-power operation when minimum system power is important and clock frequency precision is more tolerant.

4.2. Crystal Oscillators

The STMP3770 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-kHz crystal oscillator (which can be either exactly 32.0 kHz or 32.768 kHz) can optionally be used as a clock reference for the real-time clock (RTC). The 32-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 STMP3770 is powered off.
- Real-time clock circuit can use either crystal.
- Bias current adjustment.

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

4.3. Phase-Locked Loop (PLL)

The STMP3770 includes a multi-output 480-MHz PLL. The PLL is used at high frequency by the USB, and other digital clock domains divide the PLL clock to lower frequencies, as shown in [Figure 7](#). The PLL is designed for low power and low jitter.

The PLL takes the 24-MHz crystal frequency and multiplies it to produce 480 MHz. The PLL output is always 480 MHz. Variable clock frequencies can be obtained by using the dividers discussed in [Section 4.5](#). The lock time of the PLL is less than 10 μ s.

4.4. Clock Domains

To offer the best combination of performance, power consumption, and ease of use, the STMP3770 provides the clock domains illustrated in [Figure 7](#) and listed in [Table 9](#). All major functional clock domains/branches have trunk-level clock-gating for power management. The intent is to gate off clock domains when modules for certain applications are not necessary. Software will have to enable the clock domain that drives on-chip devices where trunk level clock-gating is implemented. All clock domains are asynchronous unless noted otherwise.

4.4.1. CLK_P, CLK_P_NG

The reference for the CLK_P domain can be either the ref_xtal or ref_cpu. These references drive a 10-bit clock divider to provide a maximum divide down of the selected reference clock by 2^{10} . All of the ARM core and SOC components on the CLK_H branch are considered to be on the CLK_P domain. clk_p_ng (not gated) is the free-running version of CLK_P. It does not stop running when the wait-for-interrupt function is active. It is necessary to allow the circuits to “wake up” from the wait for interrupt state.

Note: CLK_P is also called CPUCLK in this document. They are equivalent terms.

4.4.2. CLK_H

The CLK_H domain is a branch of the CLK_P domain. The CLK_H branch can be at the same CLK_P frequency or have a divided value up to 2^5 . Two divide modes exist for the CLK_H branch:

- Integer divide—In this mode, the value programmed in the HW_CLKCTRL_HBUS_DIV field represents an integer divide value.
- Fractional divide—In this mode, the value programmed in the HW_CLKCTRL_HBUS_DIV field represents a binary fraction. When the accumulation of the current count and the programmed divide value carry out of the most significant bit, a CLK_H pulse is generated. For example, to achieve an 8:3 CLK_P-to-CLK_H clock ratio, set the DIV field to 0.01100, which represents $(0*1/2) + (1*1/4) + (1*1/8) + (0*1/16) + (0*1/32)$.

The CLK_H branch can be further divided by the dynamic power adjustment logic. When all the system CLK_H components are not busy and their respective busy signals are inactive, the CLK_H branch is further divided down by the value in the HW_CLKCTRL_HBUS register. The frequency reduction of the CLK_H branch saves overall power consumption.

Since CLK_H is shared across multiple modules, module-level clock-gating allows for power reduction when the respective modules are in the soft reset state.

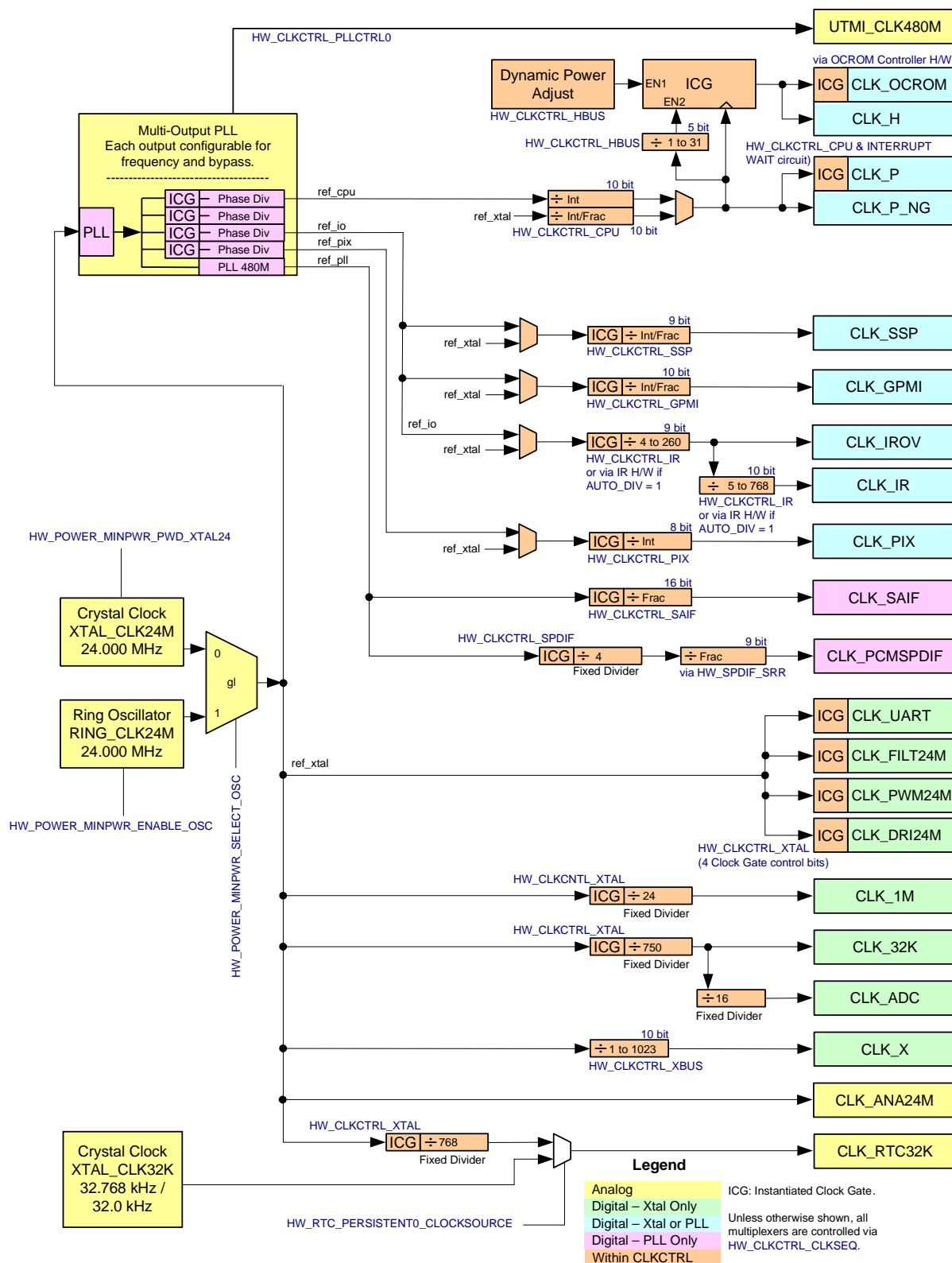


Figure 7. Logical Diagram of STMP3770 Clock Domains

Table 9. STMP3770 System Clocks

NAME	REFERENCE	DIVIDE/ FREQ	DESCRIPTION
Reference Clocks			
ref_xtal	xtal_24m/ring_24m	1	Muxed select between internal ring oscillator and external crystal.
ref_cpu	PLL	9 phase	9-phase fractional divider output used as the reference for the CPU clock divider.
ref_io	PLL	9 phase	9-phase fractional divider output used as the reference for the GPPI, and SSP clock dividers.
ref_pix	PLL	9 phase	9-phase fractional divider output used as the reference for the PIX clock divider.
ref_pll	PLL	1	Raw PLL output used as the reference for the SAIF clock divider.
Divided Clock Domains Referenced from PLL or Crystal Clock			
CLK_P	ref_xtal/ref_cpu	7 bits	ARM core clock.
CLK_H	CLK_P	5 bits	AHB/APBH clock domain. CLK_H is a gated branch of the CLK_P domain.
CLK_SSP	ref_xtal/ref_io	8 bits	SSP interface clock.
CLK_GPPI	ref_xtal/ref_io	8 bits	General-purpose memory interface clock domain.
CLK_IROV/IR	ref_io/clk_irov	9/10 bits	Oversample IR clock and IR data bit clock. The IROV clock has the ref_io as its reference. The IR clock domain uses the CLK_IROV domain as its reference.
CLK_SPDIF /CLK_PCMSPDIF	ref_xtal/ref_io		Clk_spdif is an intermediate clock that drives the CLK_PCMSPDIF fractional clock divider.
CLK_PIX	ref_xtal/ref_pix	DFD	External display interface clock. Its reference is the crystal or fractional divider output that drives a digital fractional divider (DFD).
Divided Clock Domains Referenced from Crystal Clock			
CLK_X	ref_xtal	10 bits	APBX clock domain.
CLK_UART	ref_xtal	2 bits	UART clock domain.
Fixed Clock Domains			
CLK_XTAL24M	ref_xtal	24 MHz	Used for the DRI, filter, and analog 24-MHz clock domains.
CLK_1M	ref_xtal	1 MHz	Fixed 1-MHz clock domain.
CLK_32K	ref_xtal32k/ref_xtal	32 kHz	Fixed 32-kHz clock domain. The reference is either the 32-kHz crystal or the 24-MHz crystal and divides by 768 to produce 32 kHz.
CLK_ADC	ref_xtal	2 kHz	Fixed 2-kHz clock domain.

4.4.3. HCLK_EN

The HCLK_EN control signal produces an enable to the ARM core indicating that the next rising edge of CLK_P will be coincident with the CLK_H branch rising edge. The HCLK_EN control signal is generated by the same counter that generates the CLK_H branch enable signal. The HCLK_EN signal is a synchronous control signal to the CLK_P domain.

The CLK_H branch is not a separate domain driven by a separate divider that uses CLK_P as its reference. Also, the CLK_H branch is not a 50% duty-cycle domain. The clock high time is the same as for the selected CLK_P frequency that is produced by the CLK_P divider.

4.4.4. CLK_SSP

This clock drives the SSP interface. The reference clock can be either the ref_xtal or ref_io, or the 24-MHz oscillator or 9-phase fractional divider, respectfully.

4.4.5. CLK_GPMI

This clock drives the GPMI interface. The reference clock can be either the ref_xtal or ref_io, or the 24-MHz oscillator or 9-phase fractional divider, respectfully.

4.4.6. CLK_PCMSPDIF and CLK_SPDIF

CLK_PCMSPDIF is the SPDIF interface clock. CLK_SPDIF is an internal reference clock for the PCMSPDIF fractional clock divider.

4.4.7. CLK_PIX

This clock drives the external display interface. The reference clock can be either the ref_xtal or ref_pix, or the 24-MHz oscillator or 9-phase fractional divider, respectfully.

Clock source selection is not deglitched. The LCD (or display) interface should be disabled, or in a soft reset state, when the clock selection is made. Once the divider is programmed and the reference clock source for the divider is selected, the LCD operation can be enabled with a stable clock source.

4.4.8. CLK_X

CLK_X drives all components connected to the APBX clock domain. The reference for this domain is the 24-MHz oscillator source. This reference drives a 10-bit integer clock divider.

4.4.9. CLK_UART

The CLK_UART clock drives the UART interface. The reference for this domain is the 24-MHz oscillator source.

4.4.10. CLK_XTAL24M

This is a fixed 24-MHz clock domain. This domain has multiple branches that are used by different sub modules in the system, including the following:

- CLK_FILT24M
- CLK_DRI24M
- CLK_PWM24M

4.4.11. CLK_1M

This is a fixed 1-MHz clock domain. It divides the 24-MHz crystal clock domain (/24).

4.4.12. CLK_32k

This is a fixed 32-kHz clock domain. It divides the 24-MHz crystal clock domain (/750).

4.4.13. CLK_ADC

This is a fixed 2-kHz clock domain. It divides the 24-MHz crystal clock domain (/12000).

4.5. Clock Dividers

4.5.1. Integer Clock Divider

Each divider has the capability to divide an input reference frequency by a fixed integer value. The mode is selected when the FRAC_EN field in the selected clock control register is logic 0. Odd divide values produce a 50% duty cycle. Also, divide-by-1 is a valid value and simply passes the reference clock waveform. Divide-by-zero is also valid and essentially gates the output clock to the off state. It is recommended that the respective clock gate bit is set to gate a domain off for power savings instead of setting the DIV field to 0.

4.5.2. Fractional Clock Divider

The fractional clock divider uses a fractional counter to approximate a divided clock with respect to the selected reference frequency. The accuracy of the output clock is dependent on the extent of the bits used to implement the fractional counter. The reference clock frequency and the fractional divide value must both be selected to achieve the desired output frequency.

There are two modes of operation for the fractional clock dividers. The modes refer to the range of the divide value programmed:

- Range of $2 < \text{DIV} < 2^n$
- Range of $1 < \text{DIV} < 2$

These modes are selected based on the fractional value programmed in the DIV field of the selected clock control register. If the most significant bit of the DIV field is a logic 1, then the fractional divider operates in the range $1 < \text{DIV} < 2$. Otherwise, the divider operates in the range $2 < \text{DIV} < 2^n$.

4.5.2.1. Divide Range $2 < \text{DIV} < 2^n$

As an example, if the desired divide value is 3.5, the digital approximation of $1/3.5$ is 0.01001001 using an 8-bit fractional approximation. The most significant bit of the DIV field in this case is logic 0, so the mode $2 < \text{DIV} < 2^n$ is selected. The following sequence indicates the first eight values of the fractional clock divider. Note that the accumulated count is simply the current value incremented by the value programmed in the DIV field on each cycle.

1. 0.01001001
2. 0.10010010
3. 0.11011011
4. 1.00100100 (carry-out of MSB initiates an output clock edge)
5. 0.01101101
6. 0.10110110
7. 0.11111111
8. 1.01001000 (output edge initiated)

When the carry-out of the fractional count is 1, a rising-edge output pulse is initiated, and the remainder of the accumulator is preserved. The subfractional accumulated

value is considered to determine if the output edge should occur on the falling edge of the reference clock or the rising edge of the reference clock for accuracy.

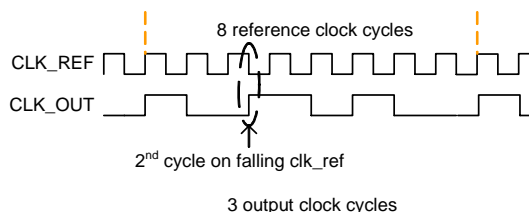


Figure 8. Divide Range $2 < \text{DIV} < 2^n$, 3/8 Example Waveform

4.5.2.2. Divide Range $1 < \text{DIV} < 2$

For the $1 < \text{DIV} < 2$ case, the most significant bit of the DIV field is logic 1. In this case, the divide-by-one circuit is enabled or disabled based on the carry out of the digital fractional divider (DFD). This option is useful to divide the 24-MHz clock to a range between 12 to 24 MHz. The effective period is equal to the reference period, because the output clock is a gated version of the reference clock. For example, a divide value of $4/3$ will allow 3 consecutive pulses of the reference clock to propagate and will then gate off a single reference clock cycle. The edge-to-edge timing is effectively equal to the reference clock.

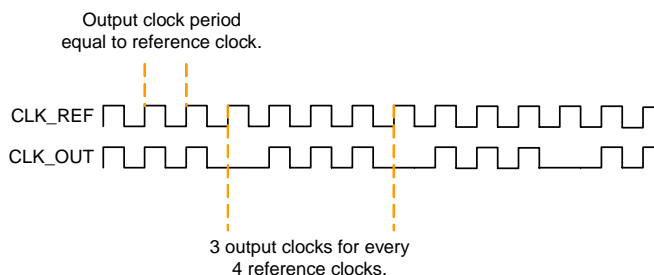


Figure 9. Divide Range $1 < \text{DIV} < 2$, 3/4 Example Waveform

4.5.3. Phase Fractional Divider (PFD) Control

A simple control interface is implemented to manage updates to the PFDs. The interface includes the signals UPDATE_***, STATUS_***, CLKGATE_***, and FRAC_***, where *** stands for CPU, IO, or PIX, since there are four instances of this interface, one for each PFD. The definition of these interface signals is as follows:

- UPDATE_***: Toggle control signals from the digital to the PFD modules in the analog section that indicate a new HW_CLKCTRL_FRAC_*** FRAC value has been loaded. Any transition on these signals, either 1 to 0 or 0 to 1 indicates a new value is loaded. This signal should be synchronized in the PFD to indicate when the FRAC field can be sampled to drive the PFD in its reference clock domain.

- STATUS_***: Toggle status bits from the PFD to the clock controller module in the digital section. When the new FRAC field has been sampled in the PFD and the new clock frequency is stable, the respective STATUS bit should toggle.
- CLKGATE_***: Active high control outputs from the clock controller to the PFD in the analog section. A high state indicates the respective PFD should be gated off and in the low-power state.
- FRAC_***: Outputs from the clock controller to the PFDs. Indicate the desired divide value that reduces the PLL output frequency.

4.6. Programming the Clock Controller

The programmer should note the following points about the clock controller:

- Change either the DIV_FRAC_EN/divider or the CLKGATE bit in the clock controller registers on a single register write. Changing these two values simultaneously will result in unexpected behavior.
- Do not switch from PLL to XTAL or XTAL to PLL in the HW_CLKCTRL_CLKSEQ register if the clock being changed is gated in any way (i.e., either by gating the reference clock or the CLKGATE bit in the register).
- Do not change the divider in any clock while either the clock is gated or the upstream reference clock is gated.

4.6.1. Clock Frequency Management

Clock frequency selection for some domains can be a function of multiple reference clock sources and divide parameters that are set in the clock controller PIO control registers. The most extreme case is using a programmable fractional PLL clock divider, a multiplexer that selects either the crystal clock or fractional PLL clock as a source to drive the clock controller divider, and the divide value for the clock controller divider itself. When programming a selected frequency, the sequence of events to achieve a given frequency must maintain the integrity of the system as a whole. During a clock system context switch, intermediate clock frequencies for selected domains cannot be faster than the subsystem or I/O interface is designed to support.

It is expected that the sequence of events when a clock domain is tuned to a desired frequency be managed by software using a hardware-status polling mechanism. Each parameter has an associated enable bit, so that all the divide parameters can be programmed in advance of the parameters taking effect. A single register, HW_CLKCTRL_CLKSEQ, contains all the enable bits that cause the divide parameters to take effect. The enable bits can be set, the busy bits can be polled for each parameter, and thus the enable/busy sequencing via software control can manage the tuning of clock frequencies throughout the system.

4.7. Reset

4.7.1. Soft Reset

In addition to the power-up reset cycle, two soft resets bits also exist to establish the initial state of the device. These bits are called HW_CLKCTRL_RESET_DIG and HW_CLKCTRL_RESET_CHIP. Setting these bits will result in a chip-wide reset cycle.

- When setting the DIG software reset bit, the chip_reset_n signal and cycle is initiated, but the power module and DC-DC converter control logic is NOT reset.

STMP3770



Table 11. HW_CLKCTRL_PLLCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29:28	LFR_SEL	RW	0x0	TEST MODE FOR SIGMATEL USE ONLY. Adjusts loop filter resistor. DEFAULT = 0x0 Default loop filter resistor. TIMES_2 = 0x1 Doubles the loop filter resistor. TIMES_05 = 0x2 Halves the loop filter resistor. UNDEFINED = 0x3 Undefined.
27:26	RSVD	RO	0x0	Reserved.
25:24	CP_SEL	RW	0x0	TEST MODE FOR SIGMATEL USE ONLY. Adjusts charge pump current. DEFAULT = 0x0 Default charge pump current. TIMES_2 = 0x1 Doubles charge pump current. TIMES_05 = 0x2 Halves the charge pump current. UNDEFINED = 0x3 Undefined.
23:22	RSVD	RO	0x0	Reserved.
21:20	DIV_SEL	RW	0x0	TEST MODE FOR SIGMATEL USE ONLY. This field is currently NOT supported. DEFAULT = 0x0 PLL frequency is 480 MHz. LOWER = 0x1 Lower the PLL frequency from 480 MHz to 384 MHz. LOWEST = 0x2 Lower the PLL frequency from 480 MHz to 288 MHz. UNDEFINED = 0x3 Undefined.
19	RSVD	RO	0x0	Reserved.
18	EN_USB_CLKS	RW	0x0	0 = 8-phase PLL outputs for USB PHY are powered down. 1 = 8-phase PLL outputs for USB PHY are powered up. Additionally, the UTMICLK120_GATE and UTMICLK30_GATE must be deasserted in the UTMI PHY to enable USB operation.
17	RSVD	RO	0x0	Reserved.
16	POWER	RW	0x0	PLL Power On (0 = PLL off; 1 = PLL On). Allow 10 us after turning the PLL on before using the PLL as a clock source. This is the time the PLL takes to lock to 480 MHz.
15:0	RSVD	RO	0x0	Reserved.

EXAMPLE:

```
HW_CLKCTRL_PLLCTRL0_WR(BF_CLKCTRL_PLLCTRL0_POWER(1)); // enable PLL
wait_10us; // Wait 10 us to let PLL lock before using it
```


Table 15. HW_CLKCTRL_CPU Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29	BUSY_REF_XTAL	RO	0x0	This read-only bit field returns a 1 when the clock divider is busy transferring a new divider value across clock domains.
28	BUSY_REF_CPU	RO	0x0	This read-only bit field returns a 1 when the clock divider is busy transferring a new divider value across clock domains.
27	RSVD	RO	0x0	Reserved.
26	DIV_XTAL_FRAC_EN	RW	0x0	0 = Enable integer divide. 1 = Enable fractional divide.
25:16	DIV_XTAL	RW	0x001	This field controls the divider connected to the crystal reference clock that drives the CLK_P domain when bypass is selected. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.
15:13	RSVD	RO	0x0	Reserved.
12	INTERRUPT_WAIT	RW	0x0	Gate off CLK_P while waiting for an interrupt.
11	RSVD	RO	0x0	Reserved.
10	DIV_CPU_FRAC_EN	RW	0x0	Reserved.
9:0	DIV_CPU	RW	0x001	This field controls the divider connected to the ref_cpu reference clock that drives the CLK_P domain when bypass is NOT selected. For changes to this field to take effect, the ref_cpu reference clock must be running. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

DESCRIPTION:

This register controls the ARM 926 clock divider.

EXAMPLE:

```
HW_CLKCTRL_CPU_WR(BF_CLKCTRL_DIV_CPU(12)); // 480 MHz / 12 = 40 MHz
```

4.8.4. AHB and APBH Bus Clock Control Register Description

The AHB and APBH Bus Clock Control Register provides controls for CLK_H (also known as HCLK) generation.

HW_CLKCTRL_HBUS	0x80040030
HW_CLKCTRL_HBUS_SET	0x80040034
HW_CLKCTRL_HBUS_CLR	0x80040038
HW_CLKCTRL_HBUS_TOG	0x8004003C

Table 16. HW_CLKCTRL_HBUS

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0										
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0							
RSVD		BUSY		RSVD		APBHDMA_AS_ENABLE	APBXDMA_AS_ENABLE	TRAFFIC_JAM_AS_ENABLE	TRAFFIC_AS_ENABLE	CPU_DATA_AS_ENABLE	CPU_INSTR_AS_ENABLE	AUTO_SLOW_MODE	RSVD		SLOW_DIV		RSVD										DIV_FRAC_EN		DIV									

Table 17. HW_CLKCTRL_HBUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29	BUSY	RO	0x0	This read-only bit field returns a 1 when the clock divider is busy transferring a new divider value across clock domains.
28:27	RSVD	RO	0x0	Reserved.
26	APBHDMA_AS_ENABLE	RW	0x0	0 = Run at the programmed CLK_H frequency. 1 = Enable auto-slow mode based on APBH DMA activity.
25	APBXDMA_AS_ENABLE	RW	0x0	0 = Run at the programmed CLK_H frequency. 1 = Enable auto-slow mode based on APBX DMA activity.
24	TRAFFIC_JAM_AS_ENABLE	RW	0x0	0 = Run at the programmed CLK_H frequency. 1 = Enable auto-slow mode when less than three masters are trying to use the AHB. More than three active masters will engage the default mode.
23	TRAFFIC_AS_ENABLE	RW	0x0	0 = Run at the programmed CLK_H frequency. 1 = Enable auto-slow mode based on AHB master activity.
22	CPU_DATA_AS_ENABLE	RW	0x0	0 = Run at the programmed CLK_H frequency. 1 = Enable auto-slow mode based on with CPU Data access to AHB.
21	CPU_INSTR_AS_ENABLE	RW	0x0	0 = Run at the programmed CLK_H frequency. 1 = Enable auto-slow mode based on with CPU Instruction access to AHB.
20	AUTO_SLOW_MODE	RW	0x0	1 = Enable CLK_H auto-slow mode. When this is set, then CLK_H will run at the slow rate until one of the fast mode events has occurred. Note that the color-space conversion (CSC) function will have poor performance if this bit is set.
19	RSVD	RO	0x0	Reserved.

Table 25. HW_CLKCTRL_SSP Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9	DIV_FRAC_EN	RW	0x0	Reserved.
8:0	DIV	RW	0x1	<p>The synchronous serial port clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0.</p> <p>NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.</p>

EXAMPLE:

```
HW CLKCTRL SSP WR(BF CLKCTRL SSP DIV(40));
```

4.8.9. General-Purpose Media Interface Clock Control Register Description

This register controls the divider that generates the general-purpose media interface (GPMI) clock, CLK_GPMI.

HW	CLKCTRL	GPMI	0x80040080
----	---------	------	------------

Table 26. HW CLKCTRL GPMI

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0				
CLKGATE	RSVD	BUSY	RSVD																		DIV_FRAC_EN	DIV													

Table 27. HW_CLKCTRL_GPMI Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	CLKGATE	RW	0x1	CLK_GPMI Gate. 0 = CLK_GPMI is not gated. 1 = CLK_GPMI is gated off. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30	RSVD	RO	0x0	Reserved.
29	BUSY	RO	0x0	This read-only bit field returns a 1 when the clock divider is busy transferring a new divider value across clock domains.
28:11	RSVD	RO	0x0	Reserved.

Table 31. HW_CLKCTRL_FRAC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	PIX_STABLE	RO	0x0	This read-only bit field is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21:16	PIXFRAC	RW	0x12	This field controls the pixel clock fractional divider. The resulting frequency will be $480 * (18/\text{PIXFRAC})$ where $\text{PIXFRAC} = 18\text{--}35$.
15:8	RSVD			Reserved
7	CLKGATECPU	RW	0x1	CPU Clock Gate. 0 = CPU fractional divider clock is enabled. 1 = CPU fractional divider clock (reference PLL ref_cpu) is off (power savings).
6	CPU_STABLE	RO	0x0	This read-only bit field is for DIAGNOSTIC PURPOSES ONLY since the fractional divide should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
5:0	CPUFRAC	RW	0x12	This field controls the CPU clock fractional divider. The resulting frequency will be $480 * (18/\text{CPUFRAC})$ where $\text{CPUFRAC} = 18\text{--}35$.

EXAMPLE:

```
HW_CLKCTRL_FRAC_WR ( BF_CLKCTRL_FRAC_CPUFRAC ( 4 ) );
```


Table 33. HW_CLKCTRL_CLKSEQ Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	BYPASS_PIX	RW	0x1	PIX Bypass Select. 0 = Select ref_pix path to generate the PIX clock domain. PLL and 9-phase fractional divider must be configured when this bit is cleared. 1 = Select ref_xtal path to generate the PIX clock domain.
0	BYPASS_SAIF	RW	0x1	Reserved—Always clear to 0. Note that this is not the reset value.

EXAMPLE:

```
HW_CLKCTRL_CLKSEQ_WR(BF_CLKCTRL_CLKSEQ_BYPASS_IR(1));
```

4.8.13. System Software Reset Register Description

This register controls full chip reset generation.

```
HW_CLKCTRL_RESET          0x800400F0
```

Table 34. HW_CLKCTRL_RESET

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																												CHIP	DIG		

Table 35. HW_CLKCTRL_RESET Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:2	RSVD	RO	0x0	Reserved.
1	CHIP	RW	0x0	Setting this bit to a logic 1 will reset the ENTIRE chip, no exceptions. This bit will also be reset after the full chip reset cycle completes.
0	DIG	RW	0x0	Setting this bit to a logic 1 will reset the digital sections of the chip. The DC-DC and power module will not be reset. This bit will also be reset after the reset cycle completes.

EXAMPLE:

```
HW_CLKCTRL_RESET_WR(BF_CLKCTRL_RESET_ALL(1));
```

0x80040100

Table 36. HW_CLKCTRL_VERSION

Table 37. HW_CLKCTRL_VERSION Bit Field Descriptions

CLKCTRL Block v2.1

5. INTERRUPT COLLECTOR

This chapter describes the interrupt control features of the STMP3770 and includes sections on interrupt nesting, FIQ generation, and CPU wait-for-interrupt mode. [Table 38](#) lists all of the interrupt sources available on the STMP3770. Programmable registers for interrupt generation and control are described in [Section 5.4](#).

5.1. Overview

The ARM926 CPU core has two interrupt input lines, IRQ and FIQ. As shown in [Figure 11](#), the Interrupt Collector (ICOLL) steers 64 interrupt sources to the two interrupt input signals on the ARM core: IRQ and FIQ.

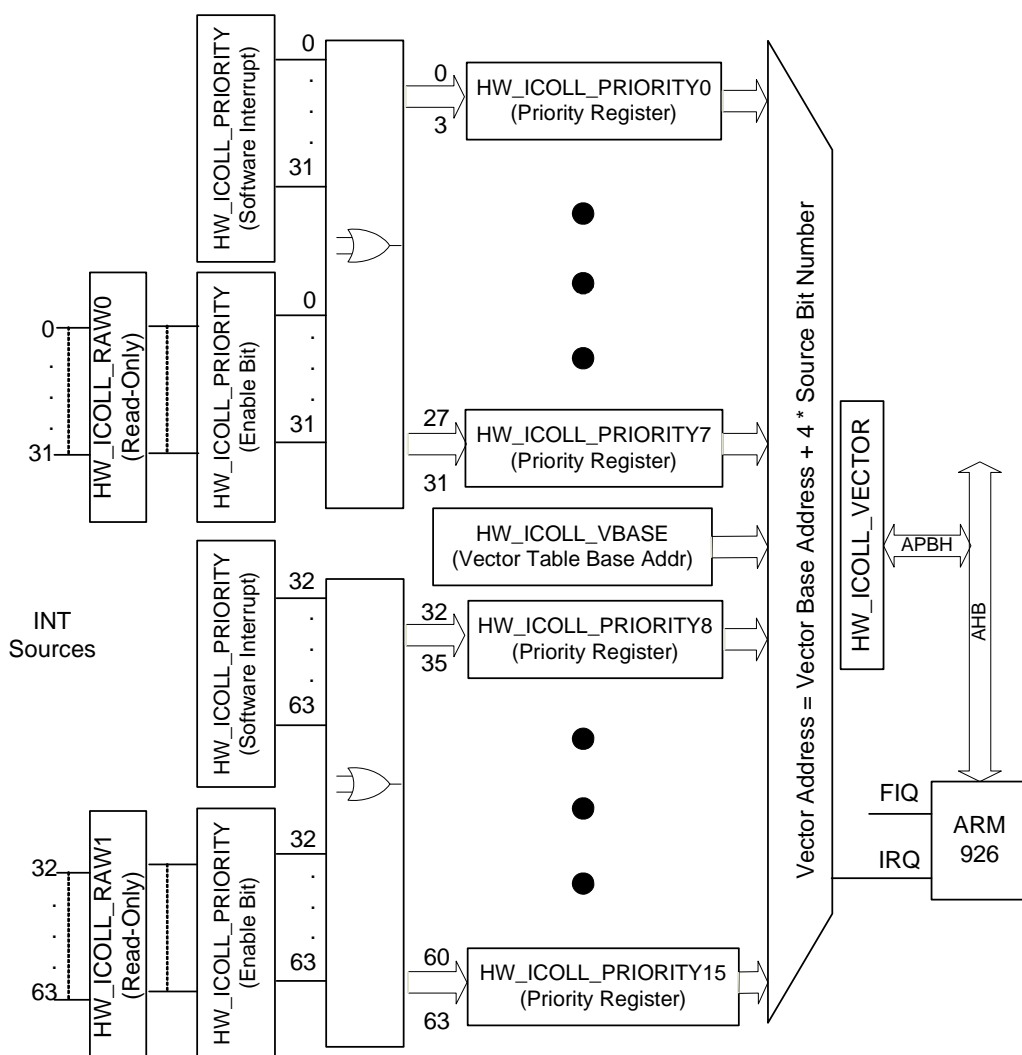


Figure 11. Interrupt Collector Diagram for IRQ Generation

5.2. Operation

Within an individual interrupt request line (IRQ only), 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. FIQ interrupts are not prioritized, nor are they vectorized. Up to eight of the interrupt sources can be selected to generate the FIQ interrupt, source bits 28–35. If more than one is routed to the FIQ, then they must be discriminated by software.

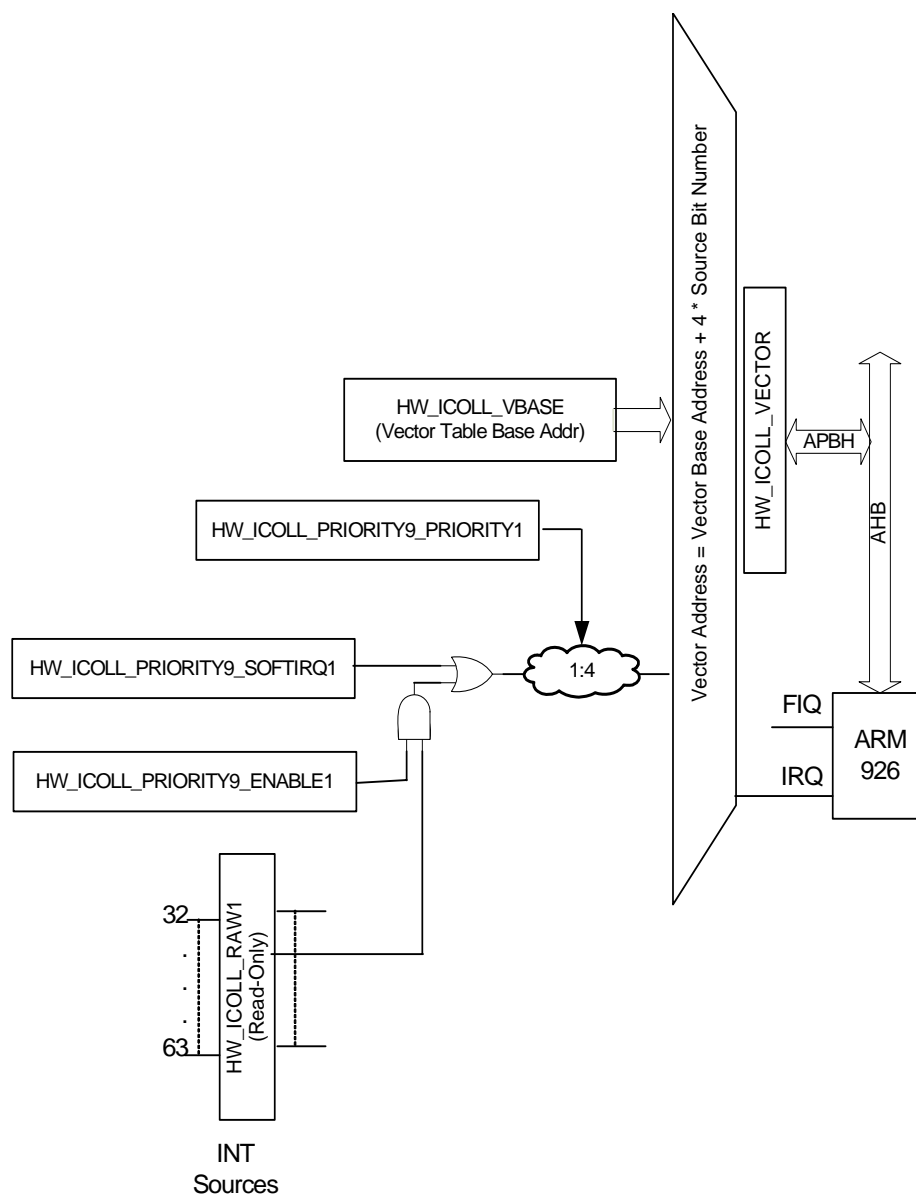


Figure 12. Interrupt Collector Bit “37” Logic

Generally, the FIQ is reserved for the exclusive use of brownout interrupts, however, one can direct any of the timer interrupts to the FIQ instead of the IRQ, if desired.

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 12](#).

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 13](#). 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:

$$\text{VectorAddress} = \text{VectorBase} + (\text{Pitch} * \text{SourceBitNumber})$$

Pitch = 4, 8, 12, 16, 20, 24, or 28 as desired. (See HW_ICOLL_CTRL_VECTOR_PITCH.)

Note: All IRQ bits on the STMP3770 must be cleared with an SCT clear-address operation only, and not by a general write.

5.2.1. 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 14](#) for the following discussion.

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 14](#). If a higher priority interrupt is pending at this point, then another IRQ exception is taken.

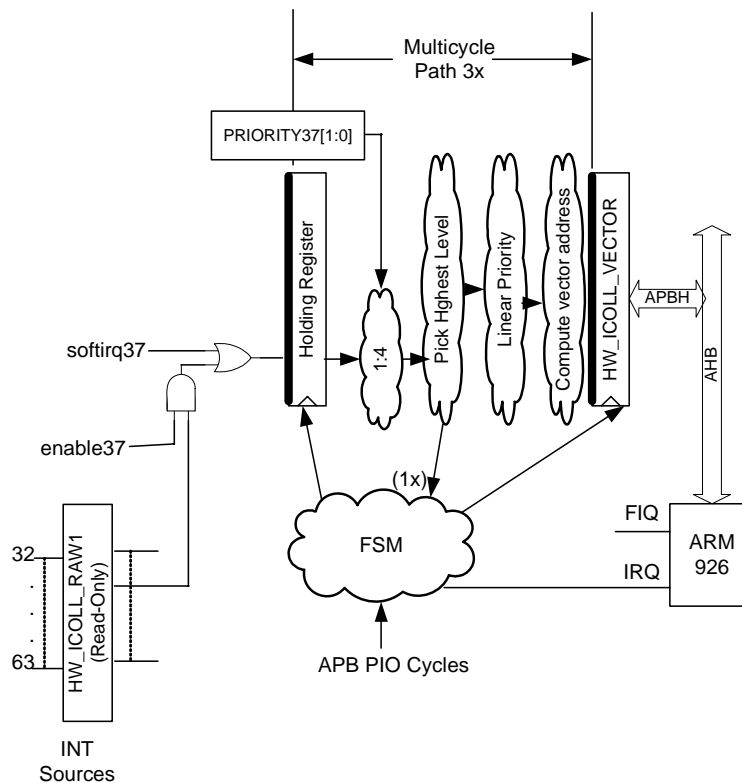


Figure 13. IRQ Control Flow

The example in [Figure 14](#) 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 14](#) 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.

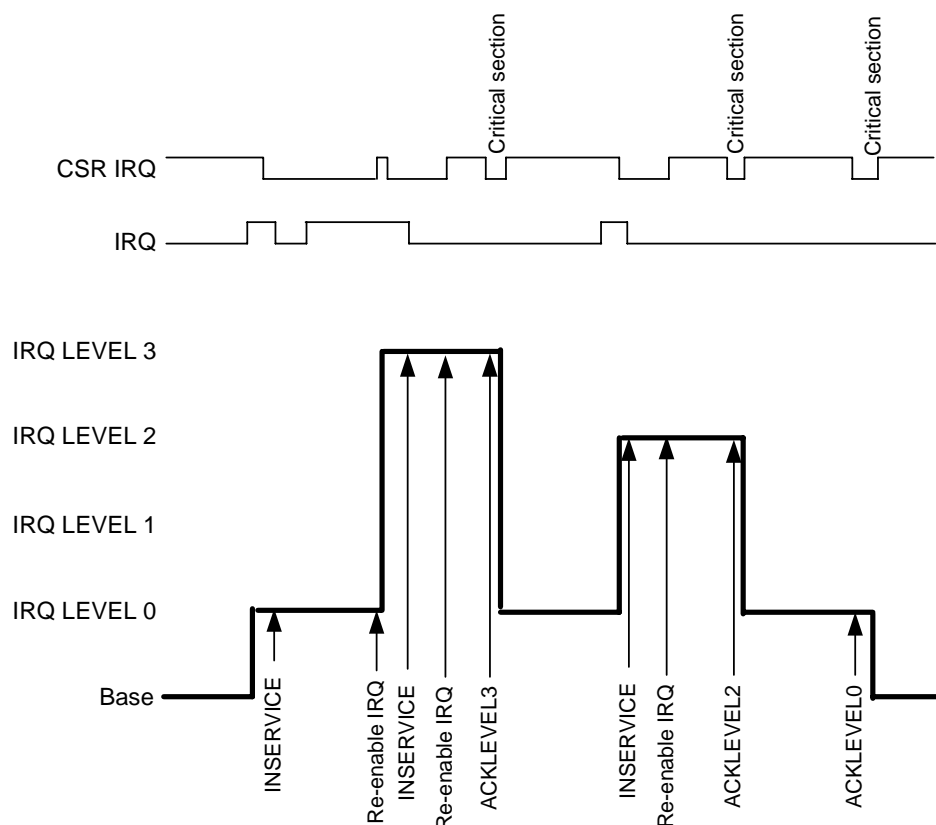


Figure 14. Nesting of Multi-Level IRQ Interrupts

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 14](#).

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.

5.2.2. FIQ Generation

Eight of the interrupt source bits can be used to generate an FIQ instead of an IRQ exception. These are source bits 28 through 35, inclusive. An FIQ may be generated by one of eight source bits. Figure 15 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.

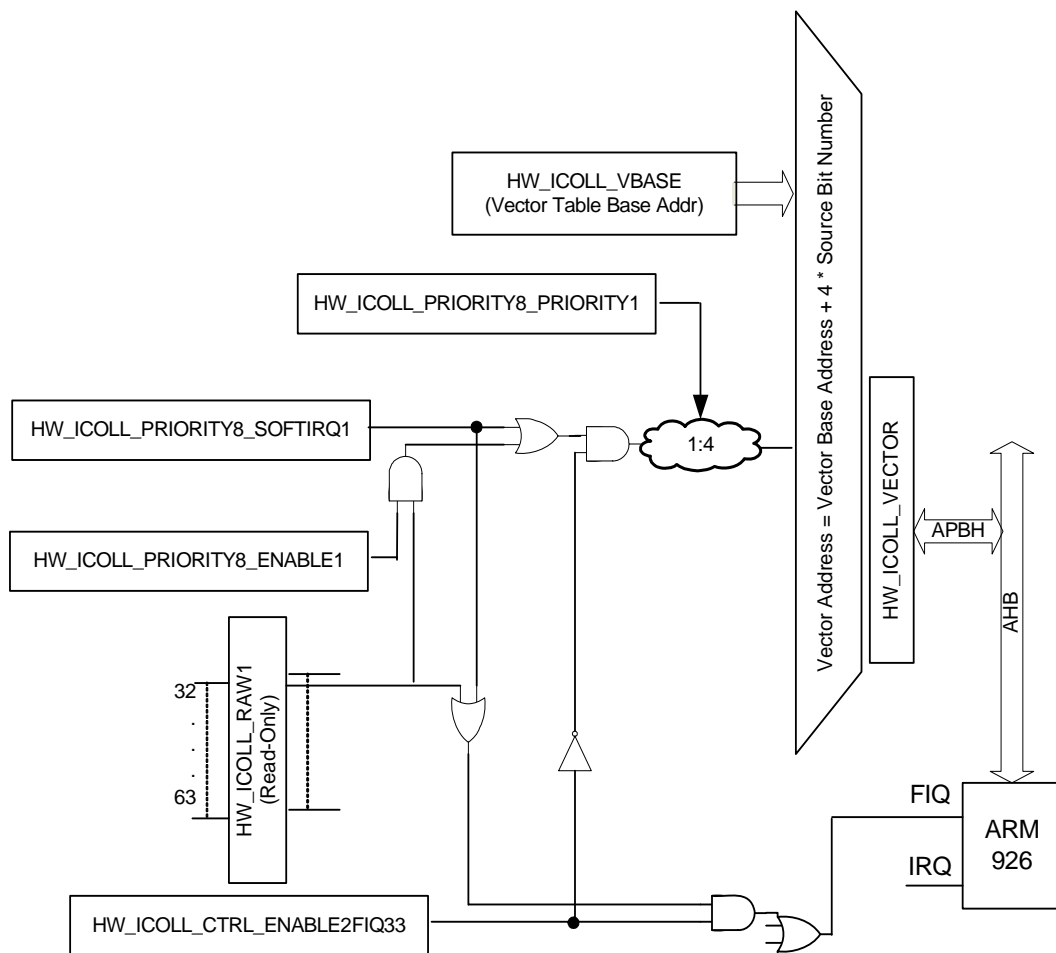


Figure 15. FIQ Generation Logic

5.2.3. Interrupt Sources

Table 38 lists all of the interrupt sources on the STMP3770. Use hw_irq.h to access these bits.

Table 38. STMP3770 Interrupt Sources

SRC	Interrupt Description	Vector	FIQ	Interrupt Source Signal
0	Debug UART error and status.	0x0000	No	dbg_uart
1	JTAG debug communications port.	0x0004	No	comms_tx, comms_rx
2	SSP2 device-level error and status.	0x0008	No	ssp2_error
3	5-V connect or disconnect. Also OTG 4.2 V.	0x000C	No	vdd5v
4	Headphone short: left/right short and common mode short.	0x0010	No	headphone_short_irq
5	DAC DMA channel.	0x0014	No	dac_dma
6	DAC FIFO buffer underflow.	0x0018	No	dac_error
7	ADC DMA channel.	0x001C	No	adc_dma
8	ADC FIFO buffer overflow.	0x0020	No	adc_error
9	SPDIF DMA channel, SAIF2 DMA channel.	0x0024	No	spdif_dma, saif2_dma
10	SPDIF FIFO underflow/overflow, SAIF1 and SAIF2 FIFO underflow/overflow /service request.	0x0028	No	spdif_error_irq, saif1_irq, saif2_irq
11	USB controller core.	0x002C	No	usb
12	USB wakeup. Also ARC core to remain suspended.	0x0030	No	usb_wakeup
13	From DMA channel for GPMI.	0x0034	No	gpmi_dma
14	From DMA channel for SSP1.	0x0038	No	espi_dma
15	SSP1 device-level error and status.	0x003C	No	espi_error
16	GPIO bank 0 interrupt.	0x0040	No	pinctrl0_interrupt
17	GPIO bank 1 interrupt.	0x0044	No	pinctrl1_interrupt
18	GPIO bank 2 interrupt.	0x0048	No	pinctrl2_interrupt
19	SAIF1 DMA channel.	0x004C	No	saif1_dma
20	From DMA channel for SSP2.	0x0050	No	ssp2_dma
21	ECC8 completion interrupt.	0x0054	No	ecc8_icoll
22	RTC alarm event.	0x0058	No	rtc_alarm
23	Application UART transmitter DMA.	0x005C	No	uart_tx_dma
24	Application UART internal error.	0x0060	No	uart
25	Application UART receiver DMA.	0x0064	No	uart_rx_dma
26	From DMA channel for I ² C.	0x0068	No	i2c_dma
27	From I ² C device detected errors and line conditions.	0x006C	No	i2c_error
28	Timer0, also routable to FIQ.	0x0070	Yes	timer0
29	Timer1, also routable to FIQ.	0x0074	Yes	timer1
30	Timer2, also routable to FIQ.	0x0078	Yes	timer2
31	Timer3, also routable to FIQ.	0x007C	Yes	timer3
32	Power module battery brownout detect, also routable to FIQ.	0x0080	Yes	batt_brownout

STMP3770**Table 38. STMP3770 Interrupt Sources (Continued)**

SRC	Interrupt Description	Vector	FIQ	Interrupt Source Signal
33	Power module VDDD brownout detect, also routable to FIQ.	0x0084	Yes	vddd_brownout
34	Power module VDDIO brownout detect, also routable to FIQ.	0x0088	Yes	vddio_brownout
35	Power module VDD18 brownout detect, also routable to FIQ.	0x008C	Yes	vdd18_brownout
36	Touch detection.	0x0090	No	touch
37	LRADC Channel 0 complete.	0x0094	No	lradc_ch0
38	LRADC Channel 1 complete.	0x0098	No	lradc_ch1
39	LRADC Channel 2 complete.	0x009C	No	lradc_ch2
40	LRADC Channel 3 complete.	0x00A0	No	lradc_ch3
41	LRADC Channel 4 complete.	0x00A4	No	lradc_ch4
42	LRADC Channel 5 complete.	0x00A8	No	lradc_ch5
43	LRADC Channel 6 complete.	0x00AC	No	lradc_ch6
44	LRADC Channel 7 complete.	0x00B0	No	lradc_ch7
45	From DMA channel for LCDIF.	0x00B4	No	lcdif_dma
46	LCDIF error.	0x00B8	No	lcdif_error
47	AHB arbiter debug trap.	0x00BC	No	digctl_debug_trap
48	RTC 1 ms tick interrupt.	0x00C0	No	rtc_1msec
49	From DMA channel for DRI.	0x00C4	No	dri_dma
50	From DRI internal error and attention IRQ.	0x00C8	No	dri_error
51	From GPPI internal error and status IRQ.	0x00CC	No	gpml_out
52	From IrDA internals. Note that the IrDA port shares DMA channels with the application UART.	0x00D0	No	ir
53	DCP Channel 0 virtual memory page copy.	0x00D4	No	dcp_vmi
54	DCP (per channel and CSC).	0x00D8	No	dcp
63:55	Reserved.	0x00DC– 0x00FC	No	interrupt_bits (force_irq_source_bits)

5.2.4. 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 0)
    "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.

5.3. 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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

5.4.1. Interrupt Collector Interrupt Vector Address Register Description

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

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
IRVECTOR																													RSVD		

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	RSVD	RO	0x0	Reserved.

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

LDPC_HW_ICOLL_VECTOR_ADDR; IRQ exception at 0xffff0018

5.4.2. Interrupt Collector Level Acknowledge Register Description

```
HW_ICOLL_LEVELACK      0x80000010
```


Table 41. HW_ICOLL_LEVELACK

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	
RSVD																										IRQLEVELACK			

Table 42. HW_ICOLL_LEVELACK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	RSVD	RO	0x0	Reserved.
3:0	IRQLEVELACK	RW	0x0	<p>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.</p> <p>LEVEL0 = 0x1 Level 0. LEVEL1 = 0x2 Level 1. LEVEL2 = 0x4 Level 2. LEVEL3 = 0x8 Level 3.</p>

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 level ACK register is decoded as not binary, i.e., 8, 4, 2, 1.

EXAMPLE:

```
HW ICOLL LEVELACK WR(HW ICOLL LEVELACK  LEVEL3);
```

5.4.3. 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 0 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 43. HW_ICOLL_CTRL

[illegible]

Table 44. HW_ICOLL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	When set to 1, 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 1, 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 soft reset 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.
29:24	RSVD	RO	0x0	Reserved.
23:21	VECTOR_PITCH	RW	0x0	When an interrupt occurs, one of the 64 input requests becomes the winning bit number, i.e., 0 to 63. This bit field selects one of eight constant multiplier values to multiply the winning bit number. The multiplied bit number is added to the vector table base to become the vector address. 0x0 and 0x1 yield a multiplier of 4 bytes. 0x2 yields a multiplier of 8 bytes. 0x3 yields a multiplier of 12 bytes, i.e., (8 + 4) bytes per step. DEFAULT_BY4 = 0x0 One word pitch. BY4 = 0x1 One-word pitch. BY8 = 0x2 Two-word pitch. BY12 = 0x3 Three-word pitch. BY16 = 0x4 Four-word pitch. BY20 = 0x5 Five-word pitch. BY24 = 0x6 Six-word pitch. BY28 = 0x7 Seven-word pitch.

Table 44. HW_ICOLL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	BYPASS_FSM	RW	0x0	Set this bit to 1 to bypass the FSM control of the request holding register and the vector address. With this bit set to 1, the vector address register is continuously updated as interrupt requests come in. Turn off all enable bits and walk a 1 through the software interrupts, observing the vector address changes. Clear to 0 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 1 disable interrupt level nesting, i.e., higher priority interrupt will interrupt lower priority. For normal operation, clear this bit to 0. NORMAL = 0x0 Normal. NO_NEST = 0x1 No support for interrupt nesting.
18	ARM_RSE_MODE	RW	0x0	Set this bit to 1 enable the ARM-style read side effect associated with the vector address register. In this mode, interrupt-in-service is signaled by the read of the HW_ICOLL_VECTOR register to acquire the interrupt vector address. Clear this bit to 0 for normal operation, in which the ISR signals an in-service interrupt explicitly by means of a write to HW_ICOLL_VECTOR. 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	Clear this bit to 0 for testing the interrupt collector without causing actual CPU interrupts. Set this bit to 1 to enable the final FIQ output to the CPU. DISABLE = 0x0 Disable. ENABLE = 0x1 Enable.
16	IRQ_FINAL_ENABLE	RW	0x1	Set this bit to 1 to enable the final IRQ output to the CPU. Clear this bit to 0 for testing the interrupt collector without causing actual CPU interrupts. DISABLE = 0x0 Disable. ENABLE = 0x1 Enable.
15:8	RSVD	RO	0x0	Reserved.
7	ENABLE2FIQ35	RW	0x0	Set this bit to 1 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.
6	ENABLE2FIQ34	RW	0x0	Set this bit to 1 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.
5	ENABLE2FIQ33	RW	0x0	Set this bit to 1 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.

Table 44. HW_ICOLL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	ENABLE2FIQ32	RW	0x0	Set this bit to 1 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.
3	ENABLE2FIQ_T3	RW	0x0	Set this bit to 1 enable Timer 3, interrupt bit 31, as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable. ENABLE = 0x1 Enable.
2	ENABLE2FIQ_T2	RW	0x0	Set this bit to 1 enable Timer 2, interrupt bit 30, as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable. ENABLE = 0x1 Enable.
1	ENABLE2FIQ_T1	RW	0x0	Set this bit to 1 enable Timer 1, interrupt bit 29, as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable. ENABLE = 0x1 Enable.
0	ENABLE2FIQ_T0	RW	0x0	Set this bit to 1 enable Timer 0, interrupt bit 28, as a source for the FIQ. WARNING: Disable IRQ for this bit prior to enabling FIQ. DISABLE = 0x0 Disable. ENABLE = 0x1 Enable.

DESCRIPTION:

This register handles the overall control of the interrupt collector, including soft reset and clock gate. In addition, it handles state machine variations like 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);
```

5.4.4. Interrupt Collector Status Register Description

This register provides a read-only view into various internal states, including the vector number of the current interrupt.

```
HW_ICOLL_STAT                                0x80000030
```

Table 45. HW ICOLL STAT

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0					
RSVD																									VECTOR_NUMBER											

Table 52. HW_ICOLL_PRIORITY0 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 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 way 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);
```

5.4.8. Interrupt Collector Priority Register 1 Description

This register specifies 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 53. HW_ICOLL_PRIORITY1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD				SOFTIRQ3	ENABLE3	PRIORITY3		RSVD				SOFTIRQ2	ENABLE2	PRIORITY2		RSVD				SOFTIRQ1	ENABLE1	PRIORITY1		RSVD				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 54. HW_ICOLL_PRIORITY1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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.
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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 to force a software interrupt. IRQ Bit 5. NO_INTERRUPT = 0x0 Turn off the software interrupt request. FORCE_INTERRUPT = 0x1 Force a software interrupt.

STMP3770

Table 56. HW_ICOLL_PRIORITY2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 8. FORCE_INTERRUPT = 0x1 Force a software interrupt.

Table 56. HW_ICOLL_PRIORITY2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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 specifies 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);
```

5.4.10. Interrupt Collector Priority Register 3 Description

This register specifies 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 57. HW ICOLL PRIORITY3

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD				SOFTIRQ3	ENABLE3	PRIORITY3		RSVD				SOFTIRQ2	ENABLE2	PRIORITY2		RSVD				SOFTIRQ1	ENABLE1	PRIORITY1		RSVD				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 58. HW_ICOLL_PRIORITY3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request IRQ Bit 15. FORCE_INTERRUPT = 0x1 Force a software interrupt.
26	ENABLE3	RW	0x0	Enable the interrupt bit through the collector. IRQ Bit 15. DISABLE = 0x0 Disable. ENABLE = 0x1 Enable.

STMP3770

Table 58. HW_ICOLL_PRIORITY3 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 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies the priority associated with four interrupt bits. In addition, this register controls the enable and software generated interrupts for the four interrupt

Table 60. HW_ICOLL_PRIORITY4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:12	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies 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);
```

5.4.12. Interrupt Collector Priority Register 5 Description

This register specifies 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 61. HW_ICOLL_PRIORITY5

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RSVD				SOFTIRQ3	ENABLE3	PRIORITY3	RSVD				SOFTIRQ2	ENABLE2	PRIORITY2	RSVD				SOFTIRQ1	ENABLE1	PRIORITY1	RSVD				SOFTIRQ0	ENABLE0	PRIORITY0				

Table 62. HW_ICOLL_PRIORITY5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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.

STMP3770

Table 62. HW_ICOLL_PRIORITY5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:4	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies 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);
```


Table 64. HW_ICOLL_PRIORITY6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies 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);
```

5.4.14. Interrupt Collector Priority Register 7 Description

This register specifies 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 65. HW_ICOLL_PRIORITY7

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4
RSVD				SOFTIRQ3	ENABLE3	PRIORITY3	RSVD				SOFTIRQ2	ENABLE2	PRIORITY2	RSVD				SOFTIRQ1	ENABLE1	PRIORITY1	RSVD				SOFTIRQ0	ENABLE0	PRIORITY0

Table 66. HW_ICOLL_PRIORITY7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 to force a software interrupt. NO_INTERRUPT = 0x0 Turn off the software interrupt request. IRQ Bit 28. FORCE_INTERRUPT = 0x1 Force a software interrupt.

Table 68. HW_ICOLL_PRIORITY8 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 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies the priority associated with four interrupt bits. In addition, this register controls the enable and software generated interrupts for the four interrupt

Table 70. HW_ICOLL_PRIORITY9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:12	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies 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);
```

5.4.17. Interrupt Collector Priority Register 10 Description

This register specifies 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

STMP3770

Table 71. HW_ICOLL_PRIORITY10

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD				SOFTIRQ3	ENABLE3	PRIORITY3		RSVD				SOFTIRQ2	ENABLE2	PRIORITY2		RSVD				SOFTIRQ1	ENABLE1	PRIORITY1		RSVD				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 72. HW_ICOLL_PRIORITY10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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.

STMP3770

Table 74. HW_ICOLL_PRIORITY11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies 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);
```

5.4.19. Interrupt Collector Priority Register 12 Description

This register specifies 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 75. HW_ICOLL_PRIORITY12

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD				SOFTIRQ3	ENABLE3	PRIORITY3		RSVD				SOFTIRQ2	ENABLE2	PRIORITY2		RSVD				SOFTIRQ1	ENABLE1	PRIORITY1		RSVD				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 76. HW_ICOLL_PRIORITY12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 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.
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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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.

Table 76. HW_ICOLL_PRIORITY12 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 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies 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);
```

5.4.20. Interrupt Collector Priority Register 13 Description

This register specifies 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 77. HW_ICOLL_PRIORITY13

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD				SOFTIRQ3	ENABLE3	PRIORITY3		RSVD				SOFTIRQ2	ENABLE2	PRIORITY2		RSVD				SOFTIRQ1	ENABLE1	PRIORITY1		RSVD				SOFTIRQ0	ENABLE0	PRIORITY0	

Table 78. HW_ICOLL_PRIORITY13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27	SOFTIRQ3	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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.

Table 80. HW_ICOLL_PRIORITY14 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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	RSVD	RO	0x0	Reserved.
19	SOFTIRQ2	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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.

Table 82. HW_ICOLL_PRIORITY15 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 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	RSVD	RO	0x0	Reserved.
11	SOFTIRQ1	RW	0x0	Set this bit to 1 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	RSVD	RO	0x0	Reserved.
3	SOFTIRQ0	RW	0x0	Set this bit to 1 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 specifies 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);
```

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

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0		
TABLE_ADDRESS																																RSVD	

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	RSVD	RO	0x0	Reserved.

This register specifies the base address of the interrupt vector table. It is used in the computation of the value supplied in HW_ICOLL_VECTOR register.

```
HW_ICOLL_VBASE_WR(pInterruptVectorTable);
```

5.4.24. Interrupt Collector Debug Register 0 Description

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

Table 98. HW_ICOLL_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x02	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x00	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

EXAMPLE:

```
if (HW_ICOLL_VERSION.B.MAJOR != 2)
    Error();
```

ICOLL Block v2.0

STMP3770



6. DEFAULT FIRST-LEVEL PAGE TABLE (DFLPT)

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

6.1. Overview

The DFLPT provides a unique method of implementing the ARM MMU first-level page table (L1PT) using a hardware-based approach. The ARM MMU L1PT must consist of 4096 page-table entries (PTE), each of which maps to 1-Mbyte sections of the 4-Gbyte system memory. Each PTE consists of a 32-bit descriptor, such that 16 Kbytes of memory is required to implement the L1PT. Using 16 Kbytes of system memory for the L1PT can be an issue for memory-constrained embedded systems.

The STMP3770 DFLPT implements a very sparse L1PT in hardware, as shown in [Figure 16](#). This is achieved by having eight *movable* page table entries (MPTE) that are fully programmable and one semi-programmable *fixed* PTE. Any of the eight MPTEs can be bound to 4095 of the 4096 sections using eight locator registers in the DIGCTL block (see [Section 7.4. “Programmable Registers” on page 131](#)).

This implementation, although sparse, is very useful in low-memory applications. For small SRAM systems (where the L1PT would typically be placed in SRAM), the DFLPT provides significant speed and power advantages (as well as saving 16 Kbytes). Using the DFLPT, a level-one descriptor fetch takes two HCLK cycles to complete.

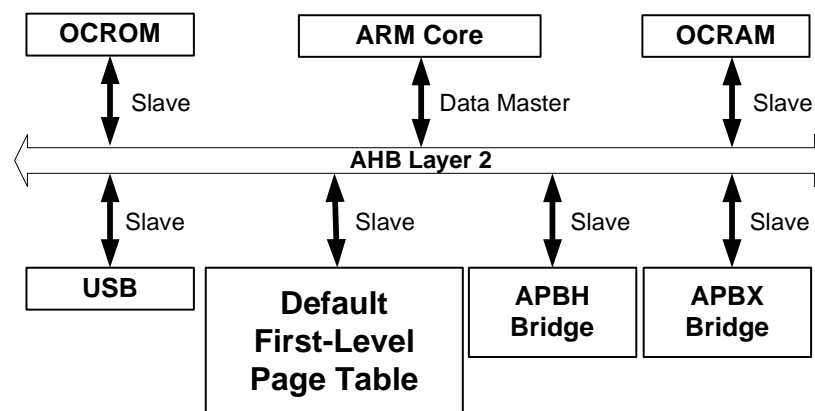


Figure 16. Default First-Level Page Table (DFLPT) Block Diagram

6.2. Operation

The DFLPT provides the following features as part of memory management within the STMP3770 embedded software system:

- 16-Kbyte AHB slave located at addresses 0x800C_0000–0x800C_3FFF supports the required 4K L1 PTEs. To use the DFLPT, point the ARM TTB to 0x800C_0000.
 - Only 32-bit word accesses are supported.
 - All AHB burst types are supported.
 - Each access has a fixed 1-cycle AHB wait state.
 - Bus errors are supported.

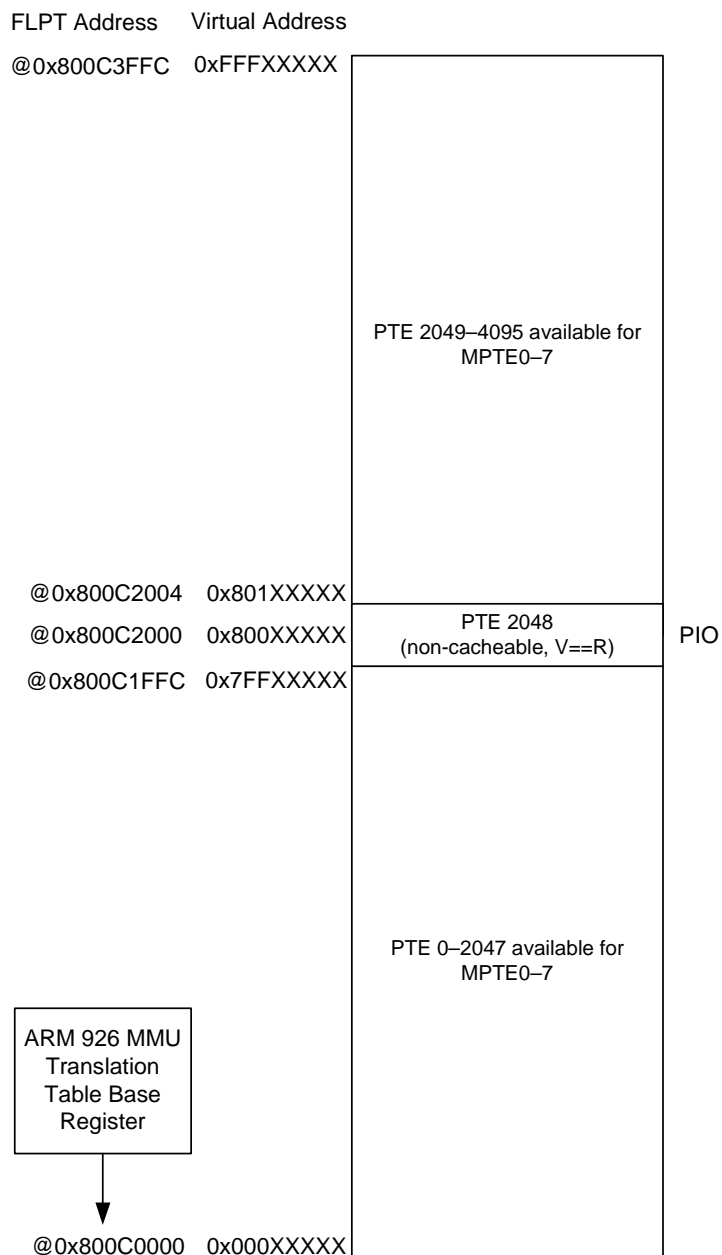
- Eight fully programmable (value and location) page table entries. All 32-bits are programmable. The location/binding of each MPTE is determined via the HW_DIGCTL_MPTEn_LOC registers in the DIGCTL module.
 - A read from an unbound PTE returns 0x0000_0000. When the MMU is enabled, this results in a section translation fault if the read is the result of a page-walk.
 - A write to an unbound PTE results in a bus error, which in turn results in an ARM data abort.
 - Each MPTE has a reset value of 0x0000_0000.
- One fixed PTE at location 2048 covers the STMP3770 PIO and register space. The location is fixed as “virtual = real”, is non-cacheable, and includes programmable bufferable Domain and AP fields.
- Each MPTE requires a location register, such that it can be mapped to any of the 4K L1 section descriptors. These location registers (HW_DIGCTL_MPTEn_LOC where n = 0...7) are located in the DIGCTL module.
 - Each HW_DIGCTL_MPTEn_LOC must be programmed with a 12-bit value that corresponds to one of 4K section entries. Once programmed, the AHB physical address of the MPTE is determined as:

$$0x800C_0000 + (HW_DIGCTL_MPTEn_LOC \ll 2)$$

$$HW_DIGCTL_MPTEn_LOC = 0x000 - 0xFFFF$$
 - The reset state of each HW_DIGCTL_MPTEn_LOC register is n (n = 0...7), e.g., HW_DIGCTL_MPTEn3_LOC has reset value of 0x0000_0003.
 - None of the MPTEn_LOC registers should be programmed to 0x800 (2048). This corresponds to the fixed entry that covers the STMP3770 register space.
 - No checking/status is given for incorrect programming (e.g., overlap). If multiple MPTEs are programmed to the same address in the DFLPT, the behavior is non-deterministic.

6.2.1. Memory Map

The virtual memory view of the DFLPT can be seen in [Figure 17](#). [Table 99](#) lists the page-table entries available in the DFLPT. See [Section 7.4](#), “Programmable Registers” on [page 131](#) for the location register descriptions.

**Figure 17. DFLPT Virtual Memory Map**

7. DIGITAL CONTROL AND ON-CHIP RAM

This chapter describes the digital control block and the on-chip RAM features of the STMP3770. It includes sections on controlling the SRAM, performance monitors, high-entropy pseudo-random number seed, and free-running microseconds counter. Programmable registers for the block are described in [Section 7.4](#).

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 first-level page table (DFLPT) controls, and HCLK performance counter, as shown in [Figure 18](#).

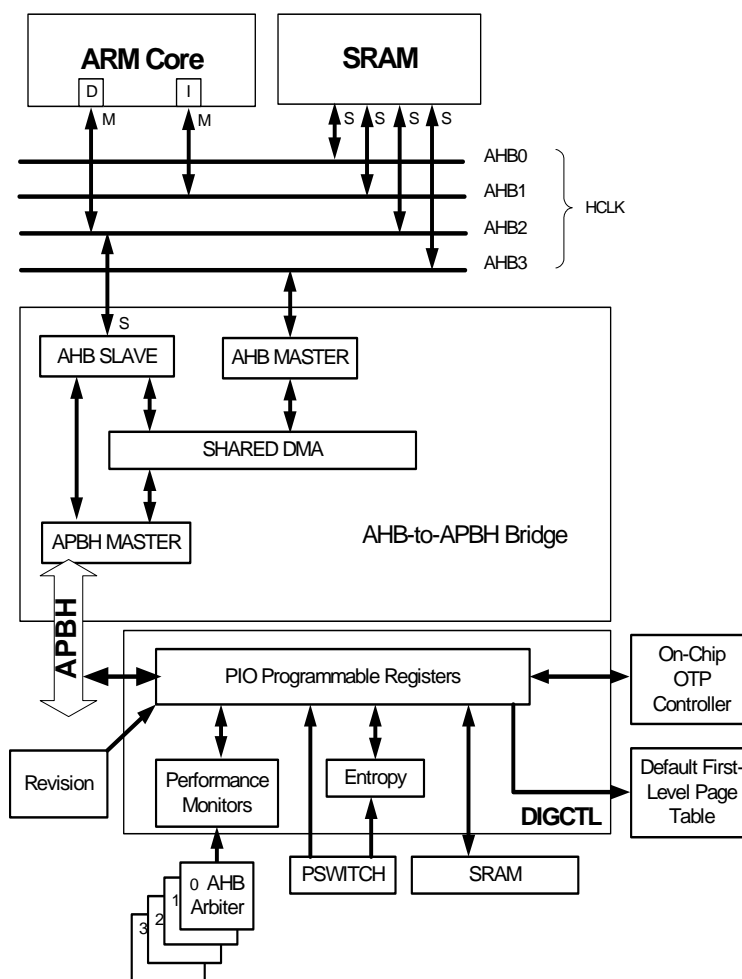


Figure 18. Digital Control (DIGCTL) Block Diagram

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 128 Kbytes. The memory is word-address interleaved, which means that consecutive word addresses skip to consecutive banks, as shown in Figure 19.

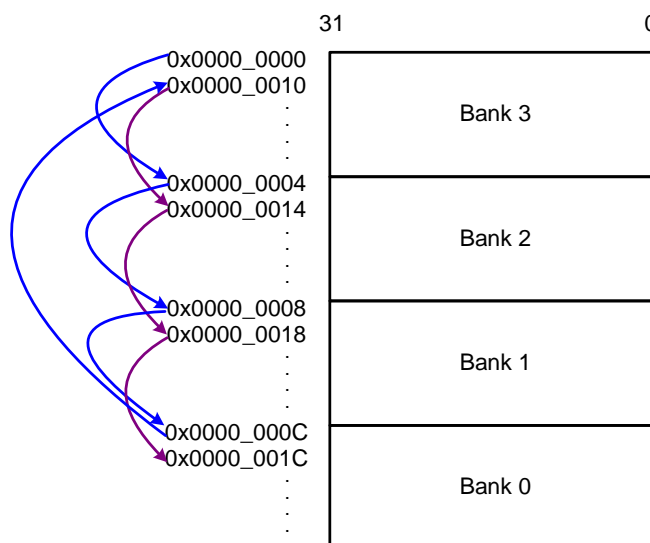


Figure 19. On-Chip RAM Partitioning

A 32-bit AHB address is arranged as shown in Table 102.

Table 102. On-Chip RAM Address Bits

AHB ADDR BITS	USAGE	DESCRIPTION
18:4	Address	Selects one of 32K words in a bank.
3:2	Bank Address	Selects between the four banks in memory.
1:0	Byte Address	Selects/masks out specific bytes within a word.

Accessing on-chip RAM requires only one initial wait state for arbitration. Other wait states happen whenever there is a read immediately following a write and also when a master is waiting to access a bank because some other master is accessing the same bank on a cycle.

The STMP3770 contains a simple 32-bit word RAM repair scheme. The purpose of this scheme is to address single-bit errors in the on-chip RAM. To enable the repair, HW_DIGCTL_RAMCTRL_RAM_REPAIR_EN must be set. Once this bit is set, the RAM controller replaces any access to the word address specified in HW_DIGCTL_RAMREPAIR_ADDR with a 32-bit redundant hardware memory. The actual repair enable and address must be read from OTP. Because these registers must be loaded by software, ROM boot code reads the OTP to see if the repair enable bit is set. If the repair enable bit is set, the ROM boot code sets

HW_DIGCTL_RAMCTRL_RAM_REPAIR_EN and copies the 16-bit word address from OTP to HW_DIGCTL_RAMREPAIR_ADDR.

7.3. Miscellaneous Controls

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

7.3.1. Performance Monitoring

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

In addition, there exists a performance monitoring register for each AHB layer (L0–L3). The HW_DIGCTL_L(n)_AHB_DATA_STALLED and HW_DIGCTL_L(n)_AHB_ACTIVE_CYCLES 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 Active 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 forced to focus on specific masters (which connect to that layer). See the HW_DIGCTL_AHB_STATS_SELECT bit description for details.

7.3.2. 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.

7.3.3. 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.

7.3.4. 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.

7.4. Programmable Registers

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

7.4.1. 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 103. HW_DIGCTL_CTRL

[illegible]

Table 104. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29	TRAP_IRQ	RW	0x0	This bit is set when an AHB access occurs to the range defined by the TRAP_ADDR registers below and the trap function is enabled with the TRAP_ENABLE bit.
28:24	RSVD	RO	0x0	Reserved.
23	DCP_BIST_CLKEN	RW	0x0	Set this bit to enable the DCP memory BIST cloc.k
22	DCP_BIST_START	RW	0x0	Set this bit to start the DCP memory BIST.
21	ARM_BIST_CLKEN	RW	0x0	Set this bit to enable the ARM BIST clock.
20	USB_TESTMODE	RW	0x0	Set this bit to get into USB test mode.
19	ANALOG_TESTMODE	RW	0x0	Set this bit to get into analog test mode.
18	DIGITAL_TESTMODE	RW	0x0	Set this bit to get into digital test mode.
17	ARM_BIST_START	RW	0x0	Set this bit to start the ARM cache BIST controller.
16	UART_LOOPBACK	RW	0x0	Set this bit 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	SAIF_LOOPBACK	RW	0x0	Set this bit to loop SAIF1 to SAIF2 and SAIF2 to SAIF1. To use SAIF loopback, configure one SAIF for transmit and the other for receive. Because this bit connects SAIF1 output to SAIF2 input and SAIF2 output to SAIF1 input, it does not matter which of the two ports is configured for TX and the other for RX. Either configuration will produce an internal TX-to-RX loopback. Note that SAIF_CLKMST_SEL is ignored when loopback is enabled. NORMAL = 0x0 No loopback. LOOPIT = 0x1 Loop SAIF1 and SAIF2 back to each other.

Table 104. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
14:13	SAIF_CLKMUX_SEL	RW	0x0	<p>Selects the muxed pins and directions for the SAIF1 and SAIF2 master clock (MCLK), bit clock (BITCLK), and left/right sample clock (LRCLK). MCLK is an optional output and, when used, is connected to the SAIF_MCLK_BITCLK muxed pin output. BITCLK can be either an input or output and can be connected to either the SAIF_MCLK_BITCLK muxed pin or the SAIF_ALT_BITCLK muxed pin. LRCLK can also be either an input or output and is connected to the SAIF_LRCLK muxed pin. When either MCLK, MCLK/BITCLK, or MCLK/BITCLK/LRCLK are configured to be outputs, the SAIF_CLKMST_SEL bit is used to determine which of the two SAIFs drives these clocks. Note that only one of the two SAIFs can be clock master at a time (READ_MODE=0 and/or SLAVE_MODE=0). When MCLK is not used and BITCLK/LRCLK are both inputs, one or both SAIFs must be configured as RX clock slaves (READ_MODE=1 and SLAVE_MODE=1). Valid configurations for SAIF_CLKMUX_SEL, as well as SAIF1 and SAIF2 SLAVE_MODE and READ_MODE bits, are:</p> <ol style="list-style-type: none"> 1) One SAIF port is TX or RX master and controls BITCLK/LRCLK (both to the pins and to the other SAIF), and MCLK can optionally be an output, while the other SAIF is an RX slave; 2) Both ports are in RX slave mode and are driven by the BITCLK/LRCLK pin inputs, and MCLK can optionally be an output; 3) Only one SAIF port is used as a TX or RX master during a given time, and SAIF_CLKMST_SEL is configured to give control of MCLK/BITCLK/LRCLK to the active port; or 4) Only one of the two ports is used as an RX slave. <p>See the table earlier in this chapter for a complete list of SAIF_CLKMUX_SEL/SAIF_CLKMST_SEL options, as well as SAIF1 and SAIF2 SLAVE_MODE/READ_MODE configurations. Note that SAIF_CLKMUX_SEL is ignored when SAIF_LOOPBACK=1. Also note that when the SAIF_ALT_BITCLK pinmux is selected to input/output BITCLK, 6-channel mode cannot be used since the SAIF2_SDATA2 pin is used for the alternate BITCLK.</p> <p>MBL_CLK_OUT = 0x0 MCLK output to SAIF_MCLK_BITCLK, BITCLK output to SAIF_ALT_BITCLK, LRCLK output to SAIF_LRCLK. BL_CLK_OUT = 0x1 BITCLK output to SAIF_MCLK_BITCLK, LRCLK output to SAIF_LRCLK. M_CLK_OUT_BL_CLK_IN = 0x2 MCLK output to SAIF_MCLK_BITCLK, BITCLK input to SAIF_ALT_BITCLK, LRCLK input to SAIF_LRCLK. BL_CLK_IN = 0x3 BITCLK input to SAIF_MCLK_BITCLK, LRCLK input to SAIF_LRCLK.</p>

Table 104. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12	SAIF_CLKMST_SEL	RW	0x0	Selects whether SAIF1 or SAIF2 drives MCLK/BITCLK/LRCLK when they are configured as outputs via SAIF_CLKMUX_SEL. Note that the selected SAIF must be configured in clock master mode (READ_MODE=0 and/or SLAVE_MODE=0). This bit must also be configured when SAIF_LOOPBACK = 1 to determine the clock master. SAIF1_MST = 0x0 SAIF1 clocks are used to output MCLK/BITCLK/LRCLK. SAIF2_MST = 0x1 SAIF2 clocks are used to output MCLK/BITCLK/LRCLK.
11	SAIF_ALT_BITCLK_SEL	RW	0x0	When the master SAIF (as selected by SAIF_CLKMST_SEL) requires all three clocks (MCLK/BITCLK/LRCLK), which is selected by programming SAIF_CLKMUX_SEL=00 or 10, this bit selects whether SAIF2_SDATA2 or PWM2 is used to input/output BITCLK. 0 = SAIF2_SDATA2 pinmux pin selected for BITCLK. 1 = PWM2 pin selected for BITCLK. Note that this bit is ignored when SAIF_CLKMUX_SEL = 01 or 11. Also note that the corresponding pin selected for BITCLK must have its MUXSEL bit field correctly programmed in the pin control block.
10:7	RSVD	RO	0x0	Reserved.
6	USE_SERIAL_JTAG	RW	0x0	Selects whether the one-wire serial JTAG interface or the alternative six-wire parallel JTAG interface is used. 0 = Parallel six-wire JTAG is enabled and is mapped to a collection of module pins that must be enabled by programming their pin MUXSEL bits in the pin control block. 1 = Serial JTAG is enabled and uses the dedicated DEBUG pin. The ROM bootcode writes this field prior to enabling JTAG, selecting which type of JTAG pin signaling to use. OLD_JTAG = 0x0 Use six-wire parallel JTAG mode. SERIAL_JTAG = 0x1 Use one-wire serial JTAG mode.
5	TRAP_IN_RANGE	RW	0x0	Determines whether the debug trap function causes a match when the master address is inside (low-address <= current-address <= high-address) the specified range. 0 = The trap occurs when the master address falls outside of the range. 1 = The check is inside the range.
4	TRAP_ENABLE	RW	0x0	Enables the AHB arbiter debug trap functions. When a trap occurs and this bit is set, an interrupt is sent to the ARM core.

Table 104. HW_DIGCTL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	DEBUG_DISABLE	RW	0x0	Set this bit to disable the ARM core's debug logic (for power savings). This bit must remain 0 following power-on reset for normal JTAG debugger operation of the ARM core. When set to 1, 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 cleared to 0 for normal operation of the USB controller. When set to 1, it gates off the clocks to the USB controller. USB_CLKGATE can be set during suspend to gate the USB clock during suspend. If this is gated, then the USB controller Reset Received bit (HW_USBCTRL_USBSTS_URI) should be not be polled for reset during suspend; use the HW_USBPHY_CTRL_RESUME_IRQ bit instead. 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	0x0	0 = The JTAG debugger is enabled. 1 = The JTAG debugger is disabled. NORMAL = 0x0 JTAG debugger enabled. SHIELDS_UP = 0x1 JTAG debugger disabled.
0	LATCH_ENTROPY	RW	0x0	Setting this bit latches the current value of the entropy register into HW_DIGCTL_ENTROPY_VALUE. This can be used get a stable value on players that do not deassert the PSWITCH while powered up.

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
```

7.4.2. 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

STMP3770

Table 105. HW_DIGCTL_STATUS

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
USB_HS_PRESENT	USB_OTG_PRESENT	USB_HOST_PRESENT	USB_DEVICE_PRESENT	RSVD																	DCP_BIST_FAIL	DCP_BIST_PASS	DCP_BIST_DONE	RSVD			JTAG_IN_USE	PACKAGE_TYPE			WRITTEN

Table 106. HW_DIGCTL_STATUS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	USB_HS_PRESENT	RO	0x1	This read-only bit returns a 1 when USB high-speed mode is present.
30	USB_OTG_PRESENT	RO	0x1	This read-only bit returns a 1 when USB on-the-go (OTG) functionality is present.
29	USB_HOST_PRESENT	RO	0x1	This read-only bit returns a 1 when USB host functionality is present.
28	USB_DEVICE_PRESENT	RO	0x1	This read-only bit returns a 1 when USB device functionality is present.
27:11	RSVD	RO	0x0	Reserved.
10	DCP_BIST_FAIL	RO	0x0	This read-only bit is a 1 if the DCP memory BIST returns a failure.
9	DCP_BIST_PASS	RO	0x0	This read-only bit is a 1 if the DCP memory BIST returns a pass.
8	DCP_BIST_DONE	RO	0x0	This read-only bit is a 1 if the DCP memory BIST has completed.
7:5	RSVD	RO	0x0	Reserved.
4	JTAG_IN_USE	RO	0x0	This read-only bit is a 1 if JTAG debugger usage has been detected.
3:1	PACKAGE_TYPE	RO	0x0	This read-only bit field returns the pin count and package type. 000=169BGA, 001=100BGA, 010=100TQFP, 011=128TQFP, 100-111=Reserved.
0	WRITTEN	RO	0x0	Set to 1 by any successful write to the HW_DIGCTL_WRITEONCE register.

DESCRIPTION:

The DIGCTL Status Register provides a read-only view to various input conditions and internal states.

EXAMPLE:

```
if(HW_DIGCTL_STATUS.PACKAGE_TYPE)
{
  // do 100-pin package things
}
```


Table 110. HW_DIGCTL_RAMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:8	SPEED_SELECT	RW	0x0	Speed select for 16Kx32 OCRM instances. Recommended value is 0x0.
7:1	RSVD	RO	0x0	Reserved.
0	RAM_REPAIR_EN	RW	0x0	Enable word repair for OCRM, using address specified in HW_DIGCTL_RAMREPAIR.

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
```

7.4.5. On-Chip RAM Repair Address Register Description

The On-Chip RAM Repair Address Register holds repair address for the on-chip SRAM. The value must be read from the OTP and copied here.

HW_DIGCTL_RAMREPAIR	0x8001C040
HW_DIGCTL_RAMREPAIR_SET	0x8001C044
HW_DIGCTL_RAMREPAIR_CLR	0x8001C048
HW_DIGCTL_RAMREPAIR_TOG	0x8001C04C

Table 111. HW DIGCTL RAMREPAIR

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RSVD															ADDR																

Table 112. HW_DIGCTL_RAMREPAIR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	RSVD	RO	0x0	Reserved.
15:0	ADDR	RW	0x0	Word repair address for OCRAM. Must be read from OTP and copied to this register. The repair is enabled when HW_DIGCTL_RAMCTRL_RAM_REPAIR_EN is set.

EXAMPLE:

```
HW_DIGCTL_RAMREPAIR.ADDR= 0xBADA; // read modify write is ok
```

7.4.6. On-Chip ROM Control Register Description

The On-Chip ROM Control Register provides settings for the OCROM.

HW_DIGCTL_ROMCTRL	0x8001C050
HW_DIGCTL_ROMCTRL_SET	0x8001C054
HW_DIGCTL_ROMCTRL_CLR	0x8001C058

0x8001C05C

Table 113. HW_DIGCTL_ROMCTRL

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0				
RSVD																												RD_MARGIN							

Table 114. HW_DIGCTL_ROMCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	RSVD	RO	0x0	Reserved.
3:0	RD_MARGIN	RW	0x2	This field is used for setting the read margin for the on-chip ROM. It programs the sense amp differential setting, and allows the trade-off between speed and robustness. This field should not be changed unless instructed by SigmaTel.

7.4.7. Software Write-Once Register Description

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

HW_DIGCTL_WRITEONCE

0x8001C060

Table 115. HW_DIGCTL_WRITEONCE

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
BITS																															

Table 116. 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;
```

7.4.8. Entropy Register Description

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

HW_DIGCTL_ENTROPY

0x8001C090

Table 117. HW DIGCTL ENTROPY

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
VALUE																															

Table 118. 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.

EXAMPLE:

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

7.4.9. Entropy Latched Register Description

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

```
HW DIGCTL ENTROPY LATCHED    0x8001C0A0
```

Table 119. HW DIGCTL ENTROPY LATCHED

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
VALUE																															

Table 120. HW_DIGCTL_ENTROPY_LATCHED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	VALUE	RO	0x0	When the LATCH_ENTROPY bit in the HW_DIGCTL_CTRL register is set to 1, the value of the HW_DIGCTL_ENTROPY register is latched into this register. This can be used to latch a stable random value on players where the PSWITCH is not deasserted after power-up.

7.4.10. SJTAG Debug Register Description

The SJTAG Debug Register controls various debug points within the SJTAG block and provides read-only views into the SJTAG state machines.

HW_DIGCTL_SJTAGDBG	0x8001C0B0
HW_DIGCTL_SJTAGDBG_SET	0x8001C0B4
HW_DIGCTL_SJTAGDBG_CLR	0x8001C0B8
HW_DIGCTL_SJTAGDBG_TOG	0x8001C0BC

Table 121. HW_DIGCTL_SJTAGDBG

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD				SJTAG_STATE												RSVD			SJTAG_TDO	SJTAG_TDI	SJTAG_MODE	DELAYED_ACTIVE				ACTIVE	SJTAG_PIN_STATE	SJTAG_DEBUG_DATA	SJTAG_DEBUG_OE		

Table 122. HW_DIGCTL_SJTAGDBG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:27	RSVD	RO	0x0	Reserved.
26:16	SJTAG_STATE	RO	0x2	Shows the state of the sjtag_state flip-flops inside the SJTAG controller. These bits implement the second state machine in the SJTAG block.
15:11	RSVD	RO	0x0	Reserved.
10	SJTAG_TDO	RO	0x0	Shows the state of the ARM JTAG TDO signal as seen inside the SJTAG controller.
9	SJTAG_TDI	RO	0x0	Shows the state of the JTAG TDI capture FF inside the SJTAG controller.
8	SJTAG_MODE	RO	0x0	Shows the state of the JTAG mode capture FF inside the SJTAG controller.
7:4	DELAYED_ACTIVE	RO	0x0	Shows the state of the delay_onewire_active_reg FF inside the SJTAG controller. These bits implement the first state machine in the SJTAG block.
3	ACTIVE	RO	0x0	Shows the state of the onewire_active_reg FF inside the SJTAG controller.
2	SJTAG_PIN_STATE	RO	0x0	Reflects the state of the input driver sampling the SJTAG pin. When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, external source can pull the SJTAG pin high without starting the SJTAG state machines. In this mode, the SJTAG_PIN_STATE bit is used to confirm continuity from the pad to the SJTAG block.

Table 122. HW_DIGCTL_SJTAGDBG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	SJTAG_DEBUG_DATA	RW	0x0	When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, then the SJTAG pin is placed in a diagnostic mode. In that case, this bit controls the input to the pad data drive signal. If HW_DIGCTL_CTRL_SJTAG_DEBUG_OE is set to 1, then this bit also controls the state of the SJTAG pin itself.
0	SJTAG_DEBUG_OE	RW	0x0	When HW_DIGCTL_CTRL_USE_SERIAL_JTAG is cleared to 0, then the SJTAG pin is placed in a diagnostic mode. In that case, this bit controls the input to the pad data output enable signal. Setting this bit to 1 turns on the SJTAG pad and drives it to the state indicated by HW_DIGCTL_CTRL_SJTAG_DEBUG_DATA.

7.4.11. Digital Control Microseconds Counter Register Description

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

HW_DIGCTL_MICROSECONDS	0x8001C0C0
HW_DIGCTL_MICROSECONDS_SET	0x8001C0C4
HW_DIGCTL_MICROSECONDS_CLR	0x8001C0C8
HW_DIGCTL_MICROSECONDS_TOG	0x8001C0CC

Table 123. HW DIGCTL MICROSECONDS

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
VALUE																															

Table 124. 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 1-microsecond rate. The 1-MHz clock driving this counter is derived from the 24.0-MHz crystal oscillator. The count value is not preserved over power downs. The 32-bit value wraps in less than two hours.

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
```

7.4.12. Digital Control Debug Read Test Register Description

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

```
HW DIGCTL DBGRD 0x8001C0D0
```


Table 147. HW_DIGCTL_OCRAM_STATUS8

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
FAILADDR01																FAILADDR00															

Table 148. HW_DIGCTL_OCRAM_STATUS8 Bit Field Descriptions

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

DESCRIPTION:

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

EXAMPLE:

```
fail_data = HW_DIGCTL_OCRAM_STATUS8_RD();
```

7.4.24. SRAM Status Register 9 Description

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

```
HW_DIGCTL_OCRAM_STATUS9      0x8001C1A0
HW_DIGCTL_OCRAM_STATUS9_SET  0x8001C1A4
HW_DIGCTL_OCRAM_STATUS9_CLR  0x8001C1A8
HW_DIGCTL_OCRAM_STATUS9_TOG  0x8001C1AC
```

Table 149. HW_DIGCTL_OCRAM_STATUS9

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
FAILADDR11																FAILADDR10															

Table 150. HW_DIGCTL_OCRAM_STATUS9 Bit Field Descriptions

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

Table 163. HW_DIGCTL_ARMCACHE

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RSVD																				CACHE_SS	RSVD	DTAG_SS	RSVD	ITAG_SS							

Table 164. HW_DIGCTL_ARMCACHE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:10	RSVD	RO	0x0	Reserved.
9:8	CACHE_SS	RW	0x0	Timing Control for 1024x32x4 RAMs (both instruction and data cache arrays).
7:6	RSVD	RO	0x0	Reserved.
5:4	DTAG_SS	RW	0x0	Timing Control for 256x22x4 RAM (DTAG).
3:2	RSVD	RO	0x0	Reserved.
1:0	ITAG_SS	RW	0x0	Timing Control for 128x22x4 RAM (ITAG).

EXAMPLE:

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

7.4.32. Debug Trap Range Low Address Description

The Debug Trap Range Low Address Register defines the lower bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AHB cycle occurs within this range.

HW_DIGCTL_DEBUG_TRAP_ADDR_LOW 0x8001C2C0

Table 165. HW_DIGCTL_DEBUG_TRAP_ADDR_LOW

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDR																															

Table 166. HW_DIGCTL_DEBUG_TRAP_ADDR_LOW Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x0	This field contains the 32-bit lower address for the debug trap range.

DESCRIPTION:

This register sets the lower address which defines the debug trap function. When this function is enabled, any active AHB cycle on either Layer 0 or Layer 3 which accesses this range will trigger an interrupt to the ARM core.

HW_DIGCTL_L0_AHB_ACTIVE_CYCLES 0x8001C340

Table 173. HW_DIGCTL_L0_AHB_ACTIVE_CYCLES

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
COUNT																															

Table 174. HW_DIGCTL_L0_AHB_ACTIVE_CYCLES Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	COUNT	RW	0x0	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 0.

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_AHB_STATS_SELECT_L0_MASTER_SELECT` are used in the arbiter to mask which master's cycles are actually recorded here.

EXAMPLE:

```
NumberCycles = HW_DIGCTL_L0_AHB_ACTIVE_CYCLES_COUNT_RD();
```

7.4.37. AHB Layer 0 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 L0 AHB DATA STALLED 0x8001C350
```

Table 175. HW_DIGCTL_L0_AHB_DATA_STALLED

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6
COUNT																									

Table 176. HW_DIGCTL_L0_AHB_DATA_STALLED 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 counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE:

```
NumberStalledCycles = HW_DIGCTL_L0_AHB_DATA_STALLED_COUNT_RD();
```


EXAMPLE:

```
NumberStalledCycles = HW DIGCTL L3 AHB DATA STALLED COUNT RD();
```

7.4.47. AHB Layer 3 Performance Metric for Valid Bus Cycles Register Description

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

```
HW DIGCTL L3 AHB DATA CYCLES 0x8001C3F0
```

Table 195. HW DIGCTL L3 AHB DATA CYCLES

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
COUNT																															

Table 196. HW DIGCTL L3 AHB DATA CYCLES 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 field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE:

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_L3_AHB_DATA_CYCLES.COUNT less than 1000000)
{
    // wait for a specific number of xfers
}
ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

7.4.48. Default First-Level Page Table Movable PTE Locator 0 Description

This register is used by the DFLPT to set the location for MPTE0.

```
HW_DIGCTL_MPTC0_LOC 0x8001C400
```

Table 197. HW DIGCTL MPTE0 LOC

3	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0		
RSVD																	LOC																

Table 198. HW DIGCTL MPTE0 LOC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:0	LOC	RW	0x0	Value of LOC corresponds to 1MB section number (0x000-0xFFFF) within the DFLPT. Do not program to 0x800 (Fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value. Doing so will result in non-deterministic behavior of the DFLPT.

DESCRIPTION:

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE0) to any of the 4096 sections.

7.4.49. Default First-Level Page Table Movable PTE Locator 1 Description

This register is used by the DFLPT to set the location for MPTE1.

```
HW DIGCTL MPTE1 LOC                                0x8001C410
```

Table 199. HW DIGCTL MPTE1 LOC

3	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RSVD																LOC															

Table 200. HW_DIGCTL_MPTE1_LOC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:0	LOC	RW	0x1	Value of LOC corresponds to 1MB section number (0x000-0xFFFF) within the DFLPT. Do not program to 0x800 (Fixed PIO entry). No two HW_DIGCTL_MPTEN_LOC registers can have the same value. Doing so will result in non-deterministic behavior of the DFLPT.

DESCRIPTION:

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE1) to any of the 4096 sections.

7.4.50. Default First-Level Page Table Movable PTE Locator 2 Description

This register is used by the DFLPT to set the location for MPTE2.

```
HW DIGCTL MPTE2 LOC                                0x8001C420
```


STMP3770

**7.4.52. Default First-Level Page Table Movable PTE Locator 4 Description**

This register is used by the DFLPT to set the location for MPTE4.

HW_DIGCTL_MPTE4_LOC 0x8001C440

Table 205. HW_DIGCTL_MPTE4_LOC

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0							
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0					
RSVD																				LOC																

Table 206. HW_DIGCTL_MPTE4_LOC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:0	LOC	RW	0x4	Value of LOC corresponds to 1MB section number (0x000-0xFFFF) within the DFLPT. Do not program to 0x800 (Fixed PIO entry). No two HW_DIGCTL_MPTE _n _LOC registers can have the same value. Doing so will result in non-deterministic behavior of the DFLPT.

DESCRIPTION:

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE4) to any of the 4096 sections.

7.4.53. Default First-Level Page Table Movable PTE Locator 5 Description

This register is used by the DFLPT to set the location for MPTE5.

HW_DIGCTL_MPTE5_LOC 0x8001C450

Table 207. HW_DIGCTL_MPTE5_LOC

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0							
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0					
RSVD																				LOC																

Table 208. HW_DIGCTL_MPTE5_LOC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:0	LOC	RW	0x5	Value of LOC corresponds to 1MB section number (0x000-0xFFFF) within the DFLPT. Do not program to 0x800 (Fixed PIO entry). No two HW_DIGCTL_MPTE _n _LOC registers can have the same value. Doing so will result in non-deterministic behavior of the DFLPT.

DESCRIPTION:

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE5) to any of the 4096 sections.

7.4.54. Default First-Level Page Table Movable PTE Locator 6 Description

This register is used by the DFLPT to set the location for MPTE6.

HW_DIGCTL_MPTE6_LOC 0x8001C460

Table 209. HW_DIGCTL_MPTE6_LOC

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0						
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0				
RSVD																		LOC																	

Table 210. HW_DIGCTL_MPTE6_LOC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:0	LOC	RW	0x6	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT. Do not program to 0x800 (Fixed PIO entry). No two HW_DIGCTL_MPTE _n _LOC registers can have the same value. Doing so will result in non-deterministic behavior of the DFLPT.

DESCRIPTION:

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE6) to any of the 4096 sections.

7.4.55. Default First-Level Page Table Movable PTE Locator 7 Description

This register is used by the DFLPT to set the location for MPTE7.

HW_DIGCTL_MPTE7_LOC 0x8001C470

Table 211. HW_DIGCTL_MPTE7_LOC

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RSVD																LOC															

STMP3770**Table 212. HW_DIGCTL_MPTE7_LOC Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:0	LOC	RW	0x7	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT. Do not program to 0x800 (Fixed PIO entry). No two HW_DIGCTL_MPTE _n _LOC registers can have the same value. Doing so will result in non-deterministic behavior of the DFLPT.

DESCRIPTION:

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE7) to any of the 4096 sections.

8. ON-CHIP OTP (OCOTP) CONTROLLER

This chapter describes the on-chip OTP (OCOTP) controller included on the STMP3770. Programmable registers are described in [Section 8.4](#).

8.1. Overview

The on-chip OTP controller (OCOTP) provides the following functions:

- Full memory-mapped (restricted) read access of 1 Kbit of on-chip OTP ROM.
- Data-register programming interface for the 1 Kbit of OTP.
- Generation of the chip hardware capability bus.
- Chip-level pin access to nonrestricted portions of OTP.

The OCOTP is connected to the APBH system peripheral bus and is accessible via the ARM core Data-AHB layer (Layer 2). Read accesses can be done at maximum HCLK frequency. Programming/writes can be performed at 24 MHz. The system diagram for the OCOTP is shown in [Figure 20](#).

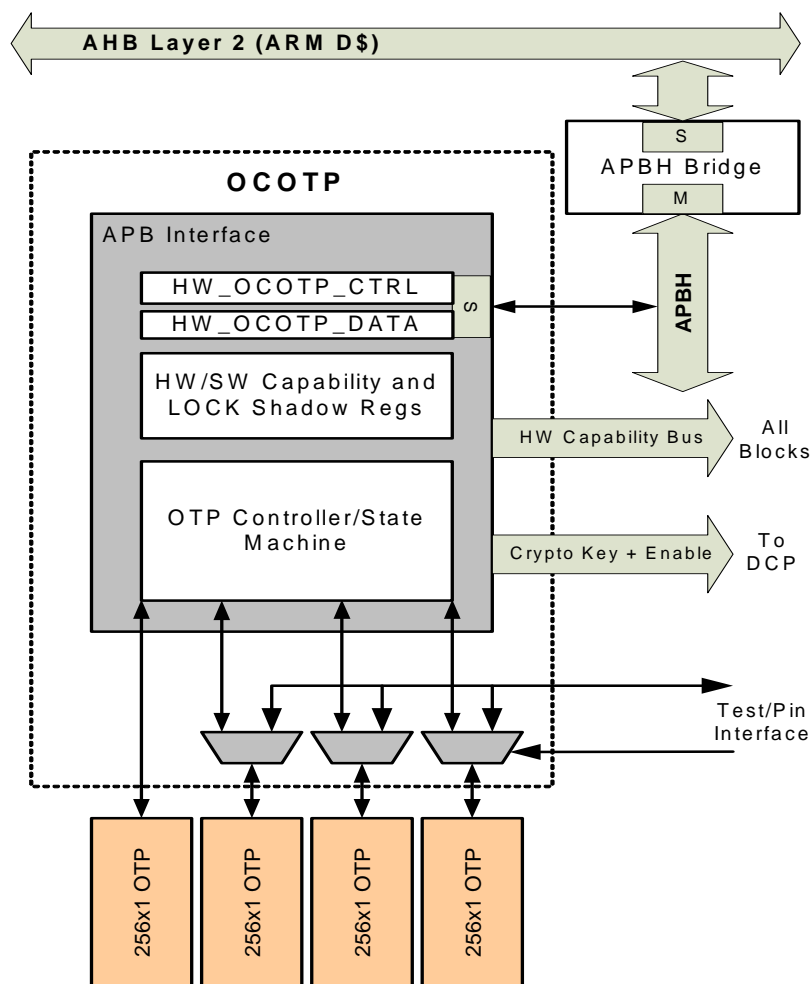


Figure 20. On-Chip OTP (OCOTP) Controller Block Diagram

8.2. Operation

The APB interface of the OCOTP provides two functions:

- Programmer-model access to registers (see [Section 8.4](#) for register details). These operations require a bank opening sequence via HW_OCOTP_CTRL_RD_BANK_OPEN.
- Restricted 32-bit word write/program access to the 1-Kbit OTP

The OTP is divided into 32-bit words (32 in total). All the 32 words are memory-mapped to APBH addresses (for reads only). Writes require the use of HW_OCOTP_CTRL_ADDR. The customer view of the high-level OTP allocation is shown in [Figure 21](#).

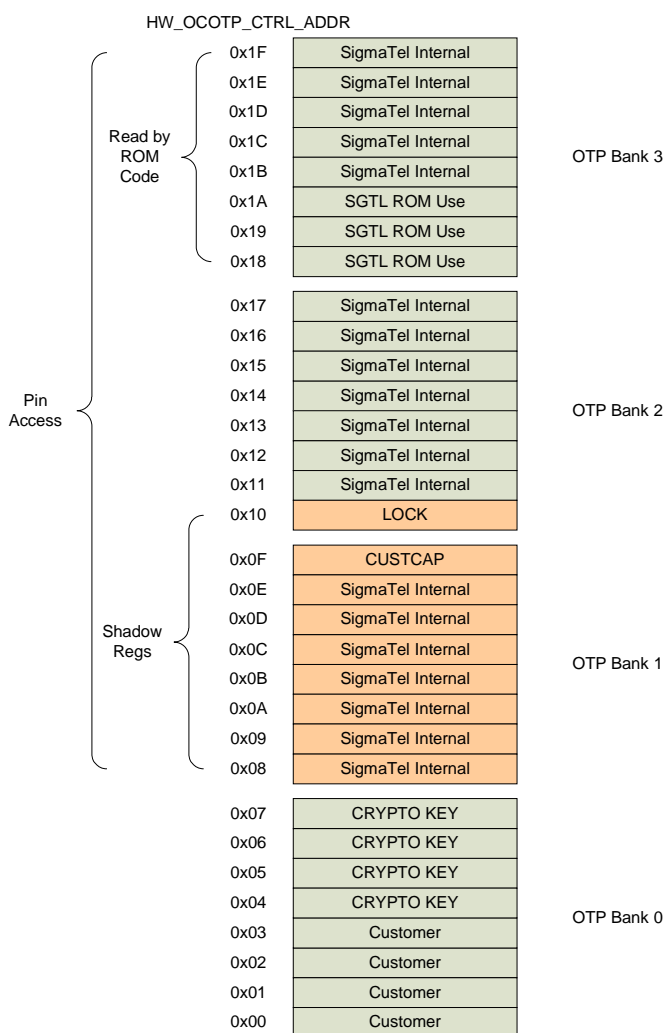


Figure 21. OCOTP Allocation—Customer View

OTP reads and writes can be performed on 32-bit words only. For writes, the 32-bit word reflects the “write-mask”, such that bit fields with 0 will not be programmed and bit fields with 1 will be programmed.

For OTP random access, the programming interface consists of:

- HW_OCOTP_DATA—Data register (32-bit) for OTP programming (writes).
- HW_OCOTP_CTRL_ADDR—Address register (5-bit) for OTP programming (writes).
- HW_OCOTP_CTRL_BUSY—Programming/write request/status handshake bit.
- HW_OCOTP_CTRL_ERROR—Read/write access error status.
- HW_OCOTP_CTRL_RD_BANK_OPEN—Status of OTP read availability (reads).

8.2.1. Software Read Sequence

Reading OTP contents is relatively simple, because all OTP words are memory-mapped on the APB space (see [Section 8.4](#) for details). These registers are read-only, except for the HW/SW capability shadow registers, which are writable until the appropriate LOCK bit in OTP is set.

Due to the fuse-read architecture, the OTP banks must be open before they can be read. This is accomplished as follows (the following does not apply to shadow registers, which can be read at any time).

1. Program the HCLK to a frequency up to the maximum allowable HCLK frequency. Note that this cannot exceed 200 MHz.
2. Check that HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_ERROR are clear.
3. Set HW_OCOTP_CTRL_RD_BANK_OPEN. This will kick the controller to put the fuses into read mode. The controller will set HW_OCOTP_CTRL_BUSY until the OTP contents are readable. Note that if there was a pending write (holding HW_OCOTP_CTRL_BUSY) and HW_OCOTP_CTRL_RD_BANK_OPEN was set, the controller would complete the write and immediately move into read operation (keeping HW_OCOTP_CTRL_BUSY set while the banks are being opened).
4. Poll for HW_OCOTP_CTRL_BUSY clear. When HW_OCOTP_CTRL_BUSY is clear and HW_OCOTP_CTRL_RD_BANK_OPEN is set, read the data from the appropriate memory-mapped address. Note that this is not necessary for registers that are shadowed. Reading before HW_OCOTP_CTRL_BUSY is cleared by the controller, will return 0xBADA_BADA and will result in the setting of HW_OCOTP_CTRL_ERROR. Because opening banks takes approximately 33 HCLK cycles, immediate polling for BUSY is not recommended.
5. Once accesses are complete, clear HW_OCOTP_CTRL_RD_BANK_OPEN. Leaving the banks open will cause current drain.

If data is accessed from a protected region (such as the crypto key, once a read LOCK bit has been set), the controller returns 0xBADA_BADA. In addition HW_OCOTP_CTRL_ERROR is set. It must be cleared by software before any new write access can be issued. Subsequent reads to unrestricted mapped OTP locations will still work successfully assuming that HW_OCOTP_CTRL_RD_BANK_OPEN is set and HW_OCOTP_CTRL_BUSY is clear.

It should be noted that after opening the banks, read latencies to OTP are “instant” (meaning they behave like regular reads from hardware registers), since parallel loading is used.

It should also be noted that setting HW_OCOTP_CTRL_RELOAD_SHADOWS to reload shadow registers does not set HW_OCOTP_CTRL_RD_BANK_OPEN. HW_OCOTP_CTRL_RD_BANK_OPEN can only be set and cleared by software. Forced reloading of shadows is covered in [Section 8.2.4](#).

8.2.2. Software Write Sequence

In order to avoid erroneous code performing erroneous writes to OTP, a special unlocking sequence is required for writes.

1. Program HCLK to 24 MHz. OTP writes do not work at frequencies above 24 MHz.
2. Set the VDDIO voltage to 2.8 V (using HW_POWER_VDDIOCTRL_TRG). The VDDIO voltage is used to program OTP. Incorrect voltage and frequency settings will result in the OTP being programmed with incorrect values.
3. Check that HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_ERROR are clear. Overlapped accesses are not supported by the controller. Any pending write must be completed before a write access can be requested. In addition, the banks cannot be open for reading, so HW_OCOTP_CTRL_RD_BANK_OPEN must also be clear. If the write is done following a previous write, the postamble wait period of 2 μ s must be followed after the clearing of HW_OCOTP_CTRL_BUSY (see [Section 8.2.3](#)).
4. Write the requested address to HW_OCOTP_CTRL_ADDR and program the unlock code into HW_OCOTP_CTRL_WR_UNLOCK. This must be programmed for each write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation.
5. Write the data to HW_OCOTP_DATA. This automatically sets HW_OCOTP_CTRL_BUSY and clears HW_OCOTP_CTRL_WR_UNLOCK. In this case, the data is a programming mask. Bit fields with ones will result in that OTP bit being set. Only the controller can clear HW_OCOTP_CTRL_BUSY. The controller will use the mask to program a 32-bit word in the OTP per the address in ADDR. At the same time that the write is accepted, the controller makes an internal copy of HW_OCOTP_CTRL_ADDR that cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to HW_OCOTP_CTRL_ADDR will not affect an active write operation. It should also be noted that, during the programming, HW_OCOTP_DATA will shift right (with zero fill). This shifting is required to program the OTP serially. During the write operation, HW_OCOTP_DATA cannot be modified.
6. Once complete, the controller clears BUSY. Beyond this, the 2- μ s postamble requirement must be met before submitting any further OTP operations (see [Section 8.2.3](#)). A write request to a protected region will result in no OTP access and no setting of HW_OCOTP_CTRL_BUSY. In addition, HW_OCOTP_CTRL_ERROR will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are in the order of 10s to 100s of microseconds per word. Write latencies will vary based on the location of the word within the OTP bank. Once a write is initiated, HW_OCOTP_DATA is shifted one bit per every 32 HCLK cycles.

Given:

8 words per OTP bank
 32 bits per word
 t_{HCLK} is the HCLK clock period
 n word locations (where $0 \leq n \leq 7$)

Then, the approximate write latency for a given word is:

$$t_{HCLK} * 32 * 32 * n$$

In addition to this latency, software must allow for the 2- μ s postamble (using HW_DIGCTL_MICROSECONDS), as described in [Section 8.2.3](#)

8.2.3. Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 μ s after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes. A recommended software sequence to meet the postamble requirements is as follows:

1. Issue the write and poll for BUSY (as per [Section 8.2.2](#)).
2. Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 μ s.
3. Perform the next OTP operation.

8.2.4. Shadow Registers and Hardware Capability Bus

The on-chip customer hardware capability bus is generated using a direct connection to the HW_OCOTP_CUSTCAP shadow register. The bits are copied from the OTP on reset. They can be modified until HW_OCOTP_LOCK_CUSTCAP_SHADOW is set.

The user can force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device, which is useful for debugging code. To force a reload:

- Set HW_OCOTP_CTRL_RELOAD_SHADOWS.
- Wait for HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_RELOAD_SHADOWS to be cleared by the controller.
- Attempting to write to the shadow registers while the shadows are being reloaded will result in the setting of HW_OCOTP_CTRL_ERROR. In addition, the register will not take the attempted write (yielding to the reload instead).
- Attempting to write to a shadow register that is locked will result in the setting of HW_OCOTP_CTRL_ERROR.

HW_OCOTP_CTRL_RELOAD_SHADOWS can be set at any time. There is no need to wait for HW_OCOTP_CTRL_BUSY or HW_OCOTP_CTRL_ERROR to be clear.

- In the case of HW_OCOTP_CTRL_BUSY being set due to an active write, the controller will perform the bank opening and shadow reloading immediately after the completion of the write.
- In the case where HW_OCOTP_CTRL_RD_BANK_OPEN is set, the shadow reload will be performed immediately after the banks are closed by software (by clearing HW_OCOTP_CTRL_RD_BANK_OPEN). It should be noted that BUSY will take approximately 33 HCLK cycles to clear, so polling for

Table 214. HW_OCOTP_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
13	RELOAD_SHADOWS	RW	0x0	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation automatically opens banks (but does not set RD_BANK_OPEN) and sets BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller. There is no need to set RD_BANK_OPEN to force the reload. If RD_BANK_OPEN is already set, its still possible to set RELOAD_SHADOWS. In this case, the shadow registers are only updated upon the clearing of RD_BANK_OPEN.
12	RD_BANK_OPEN	RW	0x0	Set this bit to open the all the OTP banks for reading. When set, the controller sets BUSY to allow time for the banks to become available (approximately 32 HCLK cycles later, at which time the controller will clear BUSY). Once BUSY is clear, the various OTP words are accessible via their memory-mapped address. Note that shadowed OTP words can be read at anytime and will not be affected by RD_BANK_OPEN. This bit must be cleared after reading is complete. Keeping the OTP banks open causes additional current draw. BUSY must be clear before this setting takes effect. If there is a write transaction pending (holding BUSY), then the bank opening sequence begins automatically upon the previous transaction clears BUSY. Note that if a read is performed from non-shadowed locations without RD_BANK_OPEN, ERROR will be set.
11:10	RSVD	RO	0x0	Reserved.
9	ERROR	RW	0x0	This bit is set by the controller when either an access to a locked region is requested or a read is requested from non-shadowed efuse locations without the banks being open. This bit must be cleared before any further write access can be performed. This bit can only be set by the controller. This bit is also set if the pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the pin interface access has completed. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
8	BUSY	RO	0x0	OTP Controller Status. When active, no new write access or bank open operations (including RELOAD_SHADOWS) can be performed. Cleared by the controller when access complete (for writes), or the banks are open (for reads). After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.

Table 236. HW_OCOTP_LOCK Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	RESERVED	RO	0x0	Reserved.
7	CUSTCAP_SHADOW	RO	0x0	Status of Customer Capability shadow register lock. When set, override of customer capability shadow bits is blocked.
6	RESERVED	RO	0x0	Reserved.
5	CRYPTODCP	RO	0x0	Status of read lock bit for DCP APB crypto access. When set, the DCP disallows reads of its crypto keys via its APB interface.
4	CRYPTOKEY	RO	0x0	Status of crypto key region (ADDR = 0x04-0x07) read/write lock bit. When set, region is locked.
3	CUST3	RO	0x0	Status of customer region word (ADDR = 0x03) write lock bit. When set, the region is locked.
2	CUST2	RO	0x0	Status of customer region word (ADDR = 0x02) write lock bit. When set, the region is locked.
1	CUST1	RO	0x0	Status of customer region word (ADDR = 0x01) write lock bit. When set, the region is locked.
0	CUST0	RO	0x0	Status of customer region word (ADDR = 0x00) write lock bit. When set, the region is locked.

8.4.13. Value of OTP Bank 3 Word 0 (ROM Use 0) Description

OTP banks must be open via HW_OCOTP_CTRL_RD_BANK_OPEN before reading this register. Reading this register without having HW_OCOTP_CTRL_RD_BANK_OPEN set and HW_OCOTP_CTRL_BUSY clear will result in HW_OCOTP_CTRL_ERROR being set and 0xBADA_BADA being returned. The bits in this register are read/write until set, at which time they become read-only.

HW_OCOTP_ROM0

0x8002C1A0

Table 237. HW_OCOTP_ROM0

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0							
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3							
BOOT_MODE								RSVD		SD_POWER_GATE_GPIO		SD_POWER_UP_DELAY				SD_BUS_WIDTH		SSP_SCK_INDEX		RSVD		DISABLE_SPI_NOR_FAST_READ		ENABLE_USB_BOOT_SERIAL_NUM		ENABLE_UNENCRYPTED_BOOT		SD_MBR_BOOT		RSVD		USE_ALT_DEBUG_UART_PINS		RSVD	

Table 238. HW_OCOTP_ROM0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	BOOT_MODE	RW	0x0	Encoded boot mode.
23:22	RSVD	RW	0x0	Reserved.
21:20	SD_POWER_GATE_GPIO	RW	0x0	SD Card Power Gate GPIO Pin Select. 00 = PWM3 01 = PWM4 10 = ROTARYA 11 = NO_GATE
19:14	SD_POWER_UP_DELAY	RW	0x0	SD Card Power-Up Delay Required after Enabling GPIO Power Gate: 000000 = 0 ms 000001 = 10 ms 000010 = 20 ms ... 111111 = 630 ms
13:12	SD_BUS_WIDTH	RW	0x0	SD Card Bus Width. 00 = 4-bit 01 = 1-bit 10 = 8-bit 11 = Reserved
11:8	SSP_SCK_INDEX	RW	0x0	Index to SSP clock speed.
7	RSVD	RW	0x0	Reserved.
6	DISABLE_SPI_NOR_FAST_READ	RW	0x0	Set to disable SPI NOR fast reads, which are used by default.
5	ENABLE_USB_BOOT_SERIAL_NUM	RW	0x0	Set to enable USB boot serial number.
4	ENABLE_UNENCRYPTED_BOOT	RW	0x0	Set to enable unencrypted boot modes.
3	SD_MBR_BOOT	RW	0x0	Set to enable SD card master boot record boot mode.
2	RSVD	RW	0x0	Reserved.
1	USE_ALT_DEBUG_UART_PINS	RW	0x0	Use alternate ROTARYA/B Debug UART RX/TX pins.
0	RSVD	RW	0x0	Reserved.

8.4.14. Value of OTP Bank 3 Word 1 (ROM Use 1) Description

OTP banks must be open via HW_OCOTP_CTRL_RD_BANK_OPEN before reading this register. Reading this register without having HW_OCOTP_CTRL_RD_BANK_OPEN set and HW_OCOTP_CTRL_BUSY clear will result in HW_OCOTP_CTRL_ERROR being set and 0xBADA_BADA being returned. The bits in this register are read/write until set, at which time they become read-only.

HW_OCOTP_ROM1

0x8002C1B0

STMP3770**Table 240. HW_OCOTP_ROM1 Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
12	USE_ALTERNATE_CE	RW	0x0	Set to direct boot loader to use the alternate chip enables.
11:8	BOOT_SEARCH_COUNT	RW	0x0	Number of 64 page blocks that should be read by the boot loader.
7:3	RSVD	RW	0x0	Reserved.
2:0	NUMBER_OF_NANDS	RW	0x0	Encoded value indicates number of external NAND devices (0 to 7). 0 = Indicates ROM will probe for the number of NAND devices connected in the system.

8.4.15. Value of OTP Bank 3 Word 2 (ROM Use 2) Description

OTP banks must be open via HW_OCOTP_CTRL_RD_BANK_OPEN before reading this register. Reading this register without having HW_OCOTP_CTRL_RD_BANK_OPEN set and HW_OCOTP_CTRL_BUSY clear will result in HW_OCOTP_CTRL_ERROR being set and 0xBADA_BADA being returned. The bits in this register are read/write until set, at which time they become read-only.

HW OCOTP ROM2

0x8002C1C0

Table 241. HW_OCOTP_ROM2

3	3	2	2	2	2	2	2	2	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
USB_VID														USB_PID																	

Table 242. HW_OCOTP_ROM2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	USB_VID	RW	0x0	USB Vendor ID.
15:0	USB_PID	RW	0x0	USB Product ID.

8.4.16. OTP Controller Version Register Description

This register indicates the version of the block for debug purposes.

HW OCOTP VERSION

0x8002C220

Table 243. HW OCOTP VERSION

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
MAJOR								MINOR								STEP															

Table 244. HW_OCOTP_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x01	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

OCOTP Block v1.2

STMP3770



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

This chapter describes the USB high-speed On-the-Go controller included on the STMP3770. It includes sections on the PIO, DMA, and UTMI interfaces, along with USB controller flowcharts. Programmable USB controller registers are described in [Section 9.6](#). Descriptions for additional programmable registers mentioned in this chapter can be found in [Section 4.8 on page 59](#), [Section 7.4. on page 131](#), [Section 10.4 on page 256](#), and [Section 29.11 on page 859](#).

9.1. Overview

The STMP3770 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 22](#). 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. The USB controller included on the STMP3770 supports five endpoints: one control, one bulk-out, one bulk-in, and two flexible endpoints. Using the OTG features, the USB controller 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 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 10](#), beginning on page [249](#). The PHY is controlled over the APBX peripheral bus.

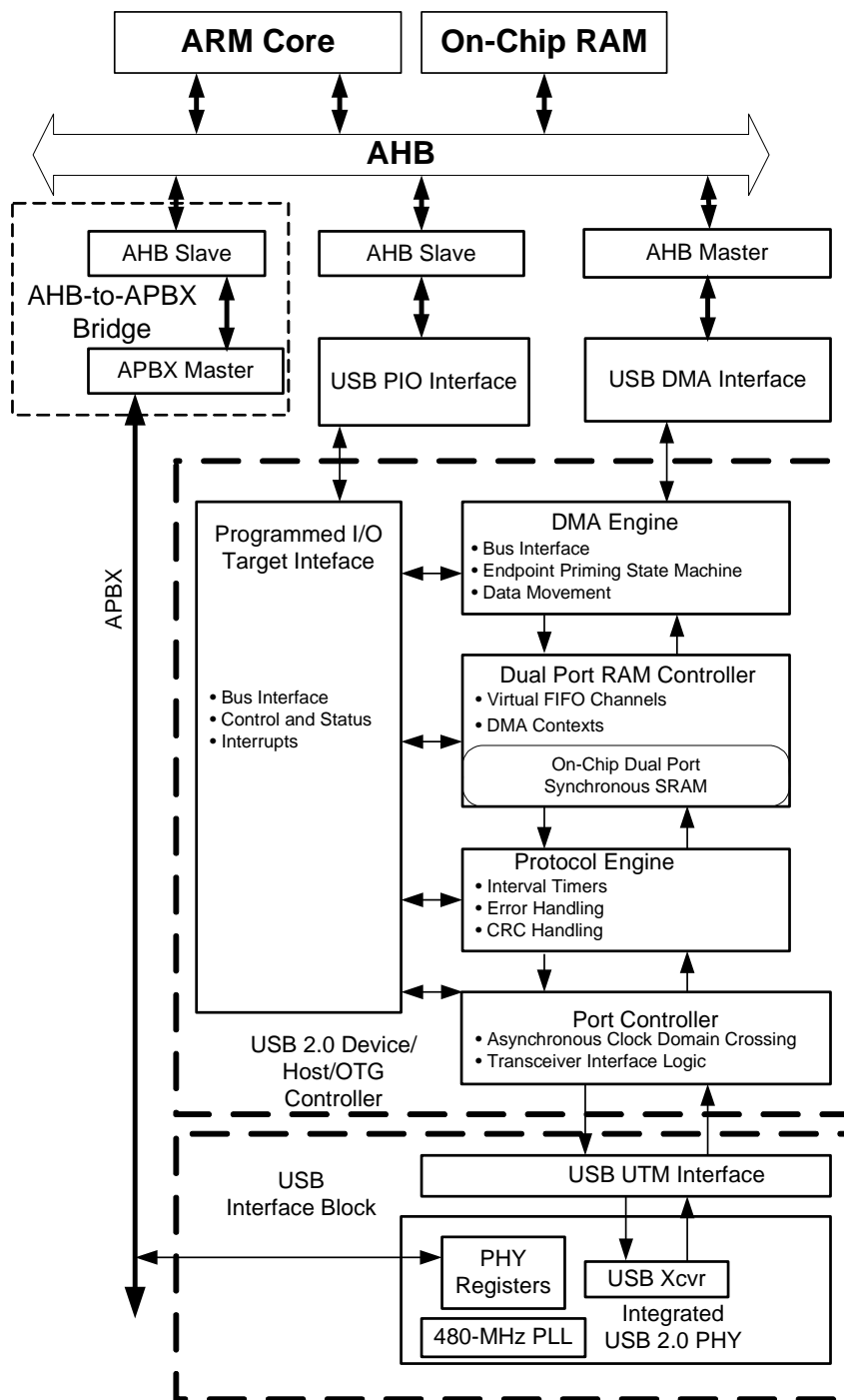


Figure 22. USB 2.0 Device Controller Block Diagram

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

The PIO interface is on an AHB slave of the USB 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.

9.3. 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 linked-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 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 linked-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.

9.4. STMP3770 USB UTM Interface

The USB UTM interface on the STMP3770 implements the specification that allows USB controllers to interface with the USB PHY. Please refer to the *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification, Version 1.05*, for additional details: <http://www.intel.com/technology/usb/spec.htm>

9.4.1. Digital/Analog Loopback Test Mode

Since the UTM 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 UTM. 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.

9.5. USB Controller Flowcharts

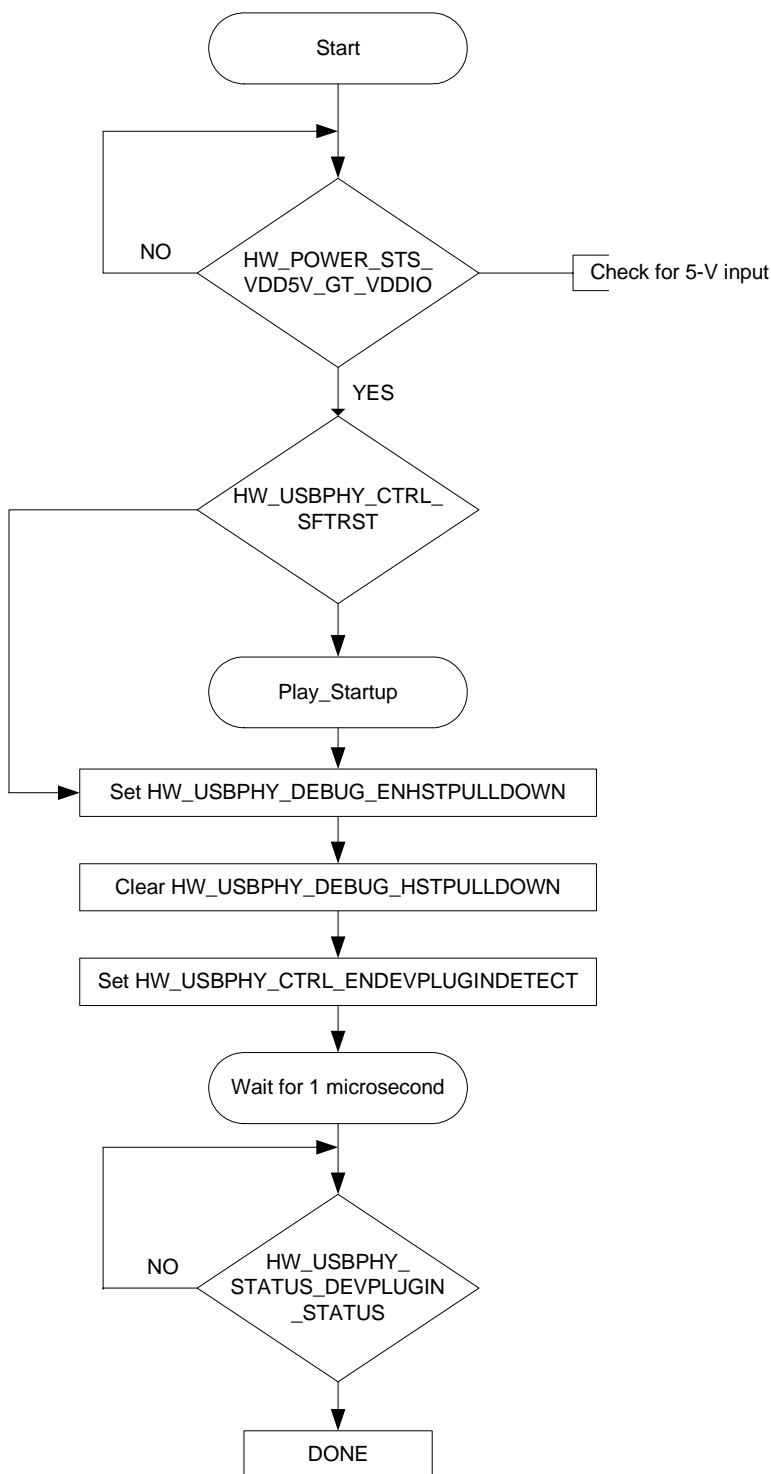


Figure 23. USB 2.0 Check_USB_Plugged_In Flowchart

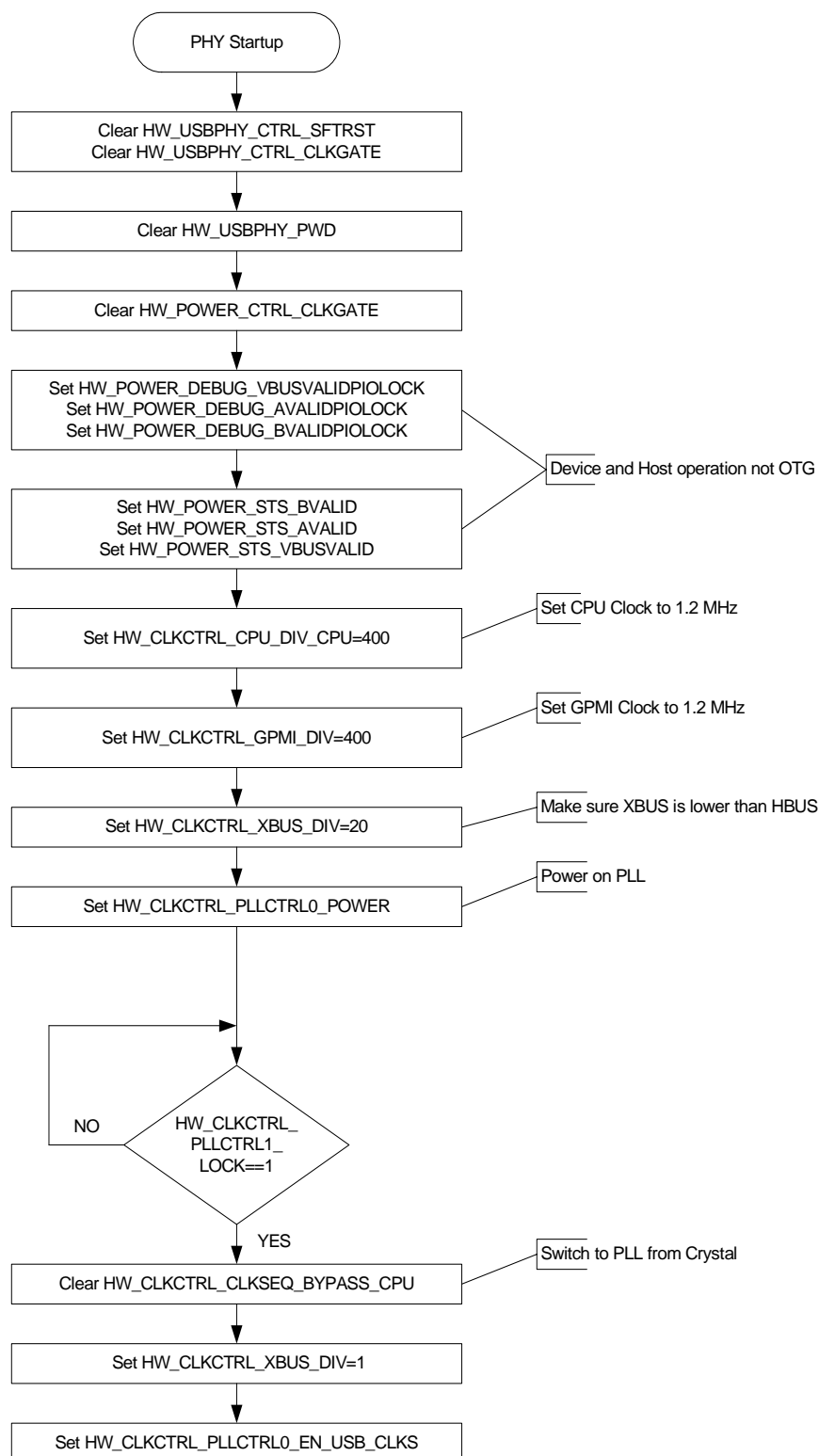


Figure 24. USB 2.0 USB PHY Startup Flowchart

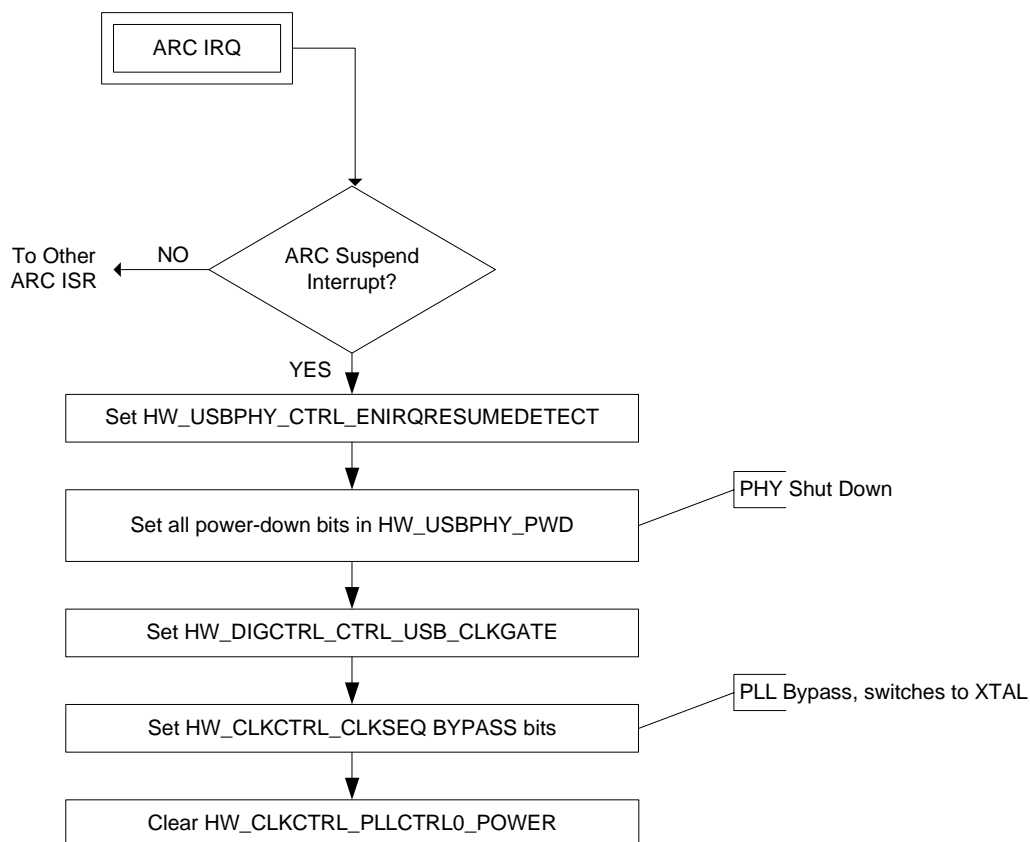


Figure 25. USB 2.0 PHY PLL Suspend Flowchart

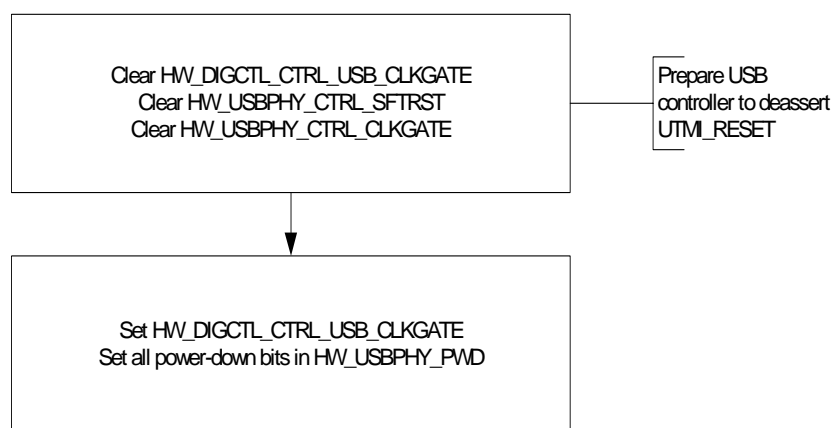


Figure 26. UTMI Powerdown

9.5.1. References

- *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification*, Version 1.05, March 2001, Jon Lueker, Steve McGowan (Editor) Ken Oliver, Dean Warren. <http://www.intel.com>
- *VSI Alliance Virtual Component Interface Standard*, Version 2 (OCB 2 2.0), April 2001, On-Chip Bus Development Working Group. <http://www.vsi.org>
- *Universal Serial Bus Specification*, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. <http://www.usb.org>
- *On-The-Go Supplement to the USB 2.0 Specification*, Revision 1.0, Dec 2001, On-The-Go Working Group of the USB-IF. <http://www.usb.org>
- *Enhanced Host Controller Interface Specification for Universal Serial Bus*, Revision 0.95, November 2000, Intel Corporation. <http://www.intel.com>
- *Universal Serial Bus Specification*, Revision 1.1, September 1998, Compaq, Intel, Microsoft, NEC. <http://www.usb.org>
- *AMBA Specification*, Revision 2.0, May 1999, ARM Limited. <http://www.arm.com>
- *UTMI+ Low Pin Interface (ULPI) Specification*, Revision 1.0 February 2004, ULPI Specification Organization. <http://www.ulpi.org>

9.6. Programmable Registers

This section includes the programmable registers supported in the USB high-speed OTG controller core.

9.6.1. Identification Register Description

The Identification Register provides a simple way to determine if the USB-HS USB 2.0 core is provided in the system. The HW_USBCTRL_ID register identifies the USB-HS USB 2.0 core and its revision.

The default value of this register is 0x0042FA05.

HW_USBCTRL_ID 0x80080000

Table 245. HW_USBCTRL_ID

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Reserved								REVISION								1	1	NID				0	0	ID							

Table 246. HW_USBCTRL_ID Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	Reserved	RO	0x0	Reserved.
23:16	REVISION[7:0]	RO	0x42	Revision number of the core.
15:14	Reserved	RO	0x3	Reserved.
13:8	NID[5:0]	RO	0x3A	One's complement version of ID[5:0].
7:6	Reserved	RO	0x0	Reserved.
5:0	ID[5:0]	RO	0x5	Configuration number. This number is set to 0x05 and indicates that the peripheral is the USB-HS USB 2.0 core.

Table 258. HW_USBCTRL_GPTIMER0LD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	Reserved	RO	0x0	Reserved.
23:0	GPTLD	RW	0x0	General-Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one-millisecond timer, load 1000 – 1=999 or 0x0003E7. Note: Max value is 0xFFFFF or 16.777215 seconds.

9.6.8. General-Purpose Timer 0 Control (Non-EHCI-Compliant) Register Description

The host/device controller driver can measure time-related activities using this timer register. This register is not part of the standard EHCI controller.

This register contains the control for the timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two modes supported by this timer, the first is a one-shot and the second is a looped count that is described in the register table below. When the timer counter value transitions to 0, an interrupt can be generated through the use of the timer interrupts in the USBTS and USBINTR registers.

HW_USBCTRL_GPTIMER0CTRL 0x80080084

Table 259. HW_USBCTRL_GPTIMER0CTRL

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
GPTRUN	GPTRST	Reserved						GPTMODE	GPTCNT																						

Table 260. HW_USBCTRL_GPTIMER0CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	GTPRUN	RW	0x0	General-Purpose Timer Run. 0 = Timer Stop 1 = Timer Run. This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT except if it stops or starts counting.
30	GPTRST	WO	0x0	General-Purpose Timer Reset. 0 = No action. 1 = Load Counter Value. Writing a 1 to this bit will reload the GPTCNT with the value in GPTLD.
29:25	Reserved	RO	0x0	Reserved.
24	GPTMODE	RW	0x0	General-Purpose Timer Mode. 0 = One Shot 1 = Repeat. This bit selects between a single timer countdown and a looped count down. In one-shot mode, the timer will count down to 0, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to 0, generate an interrupt, and automatically reload the counter to begin again.
23:0	GPTCNT	RO	0x0	General-Purpose Timer Counter. This field is the value of the running timer.

9.6.9. General-Purpose Timer 1 Load (Non-EHCI-Compliant) Register Description

Same as GPTIMER0LD description.

HW_USBCTRL_GPTIMER1LD 0x80080088

9.6.10. General-Purpose Timer 1 Control (Non-EHCI-Compliant) Register Description

Same as GPTIMER0CTRL description.

HW_USBCTRL_GPTIMER1CTRL 0x8008008C

9.6.11. Capability Register Length (EHCI-Compliant) Register Description

This 8-bit register is used to indicate which offset to add to the register base address at the beginning of the Operational Register.

HW_USBCTRL_CAPLENGTH 0x80080100

Table 261. HW_USBCTRL_CAPLENGTH

7	6	5	4	3	2	1	0
CAPLENGTH[7:0]							

Table 262. HW_USBCTRL_CAPLENGTH Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:0	CAPLENGTH	RO	0x40	Offset to add to register base address at beginning of the Operational Register.

9.6.12. Host Interface Version Number (EHCI-Compliant) Register Description

This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller.

HW_USBCTRL_HCVERSION 0x80080102

Table 263. HW_USBCTRL_HCVERSION

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
HCVERSION[15:0]															

Table 264. HW_USBCTRL_HCVERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:0	HCVERSION	RO	0x0100	BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision, and the least significant byte is the minor revision.

9.6.13. Host Control Structural Parameters (EHCI-Compliant with Extensions) Register Description

Port-steering logic capabilities are described in this register. The default value of this register is 0x00010011.

HW_USBCTRL_HCSPARAMS 0x80080104

Table 265. HW_USBCTRL_HCSPARAMS

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
Reserved				N_TT				N_PTT				Reserved		PI	N_CC				N_PCC				Reserved				PPC	N_PORTS			

Table 266. HW_USBCTRL_HCSPARAMS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	Reserved	RO	0x0	Reserved.
27:24	N_TT[3:0]	RO	0x0	Number of Transaction Translators (N_TT). Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This in a non-EHCI field to support embedded TT.
23:20	N_PTT[3:0]	RO	0x0	Number of Ports per Transaction Translator (N_PTT). Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. This in a non-EHCI field to support embedded TT.
19:17	Reserved	RO	0x0	Reserved.
16	PI	RO	0x1	Port Indicators (P INDICATOR). Indicates whether the ports support port indicator control. When set to 1, the port status and control registers include a read/writable field for controlling the state of the port indicator.
15:12	N_CC[3:0]	RO	0x0	Number of Companion Controller (N_CC). Indicates the number of companion controllers associated with this USB2.0 host controller. A 0 in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported. A value larger than 0 in this field indicates there are companion USB host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports.
11:8	N_PCC[3:0]	RO	0x0	Number of Ports per Companion Controller. Indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC.
7:5	Reserved	RO	0x0	Reserved.

Table 268. HW_USBCTRL_HCCPARAMS Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
7:4	IST[7:4]	RO	0x0	<p>Isochronous Scheduling Threshold. Indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule.</p> <p>When bit 7 is 0, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state.</p> <p>When bit 7 is a 1, then host software assumes the host controller may cache an isochronous data structure for an entire frame.</p>
3	Reserved	RO	0x0	Reserved.
2	ASP	RO	0x1	<p>Asynchronous Schedule Park Capability. Default = 1.</p> <p>If this bit is set to a 1, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register.</p>
1	PFL	RO	0x1	<p>Programmable Frame List Flag.</p> <p>If this bit is set to 0, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to 0.</p> <p>If set to a 1, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous.</p>
0	ADC	RO	0x0	64-bit Addressing Capability. No 64-bit addressing capability is supported.

9.6.15. Device Interface Version Number (Non-EHCI-Compliant) Register Description

The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

HW_USBCTRL_DCIVERSION 0x80080120

Table 269. HW_USBCTRL_DCIVERSION

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DCIVERSION[15:0]															

Table 270. HW_USBCTRL_DCIVERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:0	DCIVERSION	RO	0x1	Two-byte BCD encoding of the interface version number.

9.6.16. Device Control Capability Parameters (Non-EHCI-Compliant) Register Description

These fields describe the overall host/device capability of the controller. The default value of this register is 0x00000185.

HW_USBCTRL_DCCPARAMS 0x80080124

Table 271. HW_USBCTRL_DCCPARAMS

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3
Reserved																				HC	DC	Reserved			DEN[4:0]			

Table 272. HW_USBCTRL_DCCPARAMS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:9	Reserved	RO	0x0	Reserved.
8	HC	RO	0x1	Host Capable. When this bit is 1, this controller is capable of operating as an EHCI-compatible USB 2.0 host controller.
7	DC	RO	0x1	Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6:5	Reserved	RO	0x0	Reserved.
4:0	DEN[4:0]	RO	0x5	Device Endpoint Number. This field indicates the number of endpoints built into the device controller, which is 5.

9.6.17. USB Command Register Description

The serial bus host/device controller executes the command indicated in this register.

* Default Value: 0x00080B00 (Host mode), 0x00080000 (Device mode)

HW_USBCTRL_USBCMD 0x80080140

STMP3770

Table 273. HW_USBCTRL_USBCMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
Reserved								ITC[7:0]								FS2	ATDTW	SUTW	Reserved	ASPE	Reserved	ASP1	ASP0	LR	IAA	ASE	PSE	FS1	FS0	RST	RS

Table 274. HW_USBCTRL_USBCMD Bit Field Descriptions

BITS	LABEL	RW	RESET*	DEFINITION
31:24	Reserved	RO		Reserved.
23:16	ITC[7:0]	RW		Interrupt Threshold Control. Default 0x08. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are: VALUE MAXIMUM INTERRUPT INTERVAL 0x00 = IMM (Immediate (no threshold)) 0x01 = 1_MICROFRAME 0x02 = 2_MICROFRAME 0x04 = 4_MICROFRAME 0x08 = 8_MICROFRAME 0x10 = 16_MICROFRAME 0x20 = 32_MICROFRAME 0x40 = 64_MICROFRAME
15, 3:2	FS[2:0]	RW or RO		Frame List Size. Default 000b. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. 000b = 1024_ELEMENTS (4096 bytes) Default value 001b = 512_ELEMENTS (2048 bytes) 010b = 256_ELEMENTS (1024 bytes) 011b = 128_ELEMENTS (512 bytes) 100b = 64_ELEMENTS (256 bytes) 101b = 32_ELEMENTS (128 bytes) 110b = 16_ELEMENTS (64 bytes) 111b = 8_ELEMENTS (32 bytes) Only the host controller uses this field.
14	ATDTW	RW		Add dTD TripWire (device mode only). This bit is used as a semaphore to ensure the proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit shall also be cleared by hardware when is state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.

Table 274. HW_USBCTRL_USBCMD Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
13	SUTW	RW		Setup TripWire (device mode only). This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (See USBMODE) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists.
12	Reserved	RO		Reserved.
11	ASPE	RW		Asynchronous Schedule Park Mode Enable (OPTIONAL). This bit defaults to 0x1. Software uses this bit to enable or disable Park mode. When this bit is 1, Park mode is enabled. When this bit is a 0, Park mode is disabled. This field is set to 1 in host mode; 0 in device mode.
10	Reserved	RO		Reserved.
9:8	ASP[1:0]	RW		Asynchronous Schedule Park Mode Count (OPTIONAL). This field defaults to 0x3 and is R/W. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. See Section 4.10.3.2 of the EHCI specification for full operational details. Valid values are 0x1–0x3. Software must not write a 0 to this bit as this will result in undefined behavior. This field is set to 0x3 in host mode; 0x0 in device mode.
7	LR	RO		Light Host/Device Controller Reset (OPTIONAL). Not Implemented. This field will always be 0.
6	IAA	RW		Interrupt on Async Advance Doorbell. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is 1, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to 0 after it has set the Interrupt on Sync Advance status bit in the USBSTS register to 1. Software should not write a 1 to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a 1 to this bit when device mode is selected will have undefined results.

STMP3770**Table 274. HW_USBCTRL_USBCMD Bit Field Descriptions (Continued)**

BITS	LABEL	RW	RESET*	DEFINITION
5	ASE	RW		Asynchronous Schedule Enable. Default 0. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit.
4	PSE	RW		Periodic Schedule Enable. Default 0b. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit.

Table 274. HW_USBCTRL_USBCMD Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
1	RST	RW		<p>Controller Reset (RESET). Software uses this bit to reset the controller. This bit is set to 0 by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a 0 to this register.</p> <p><u>Host Controller:</u> When software writes a 1 to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a 1 when the HCHalted bit in the USBSTS register is a 0. Attempting to reset an actively running host controller will result in undefined behavior.</p> <p><u>Device Controller:</u> When software writes a 1 to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a 1 to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.</p>
0	RS	RW		<p>Run/Stop (RS). Default 0. 1=Run. 0=Stop.</p> <p><u>Host Controller:</u> When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a 1 to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a 1).</p> <p><u>Device Controller:</u> Writing a 1 to this bit will cause the device controller to enable a pullup on D+ and initiate an attach event. This control bit is not directly connected to the pullup enable, as the pullup will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event.</p>

STMP3770

9.6.18. USB Status Register Description

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them.

* Default Value: 0x00001000 (Host mode), 0x00000000 (Device mode)

HW_USBCTRL_USBSTS 0x80080144

Table 275. HW_USBCTRL_USBSTS

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3
Reserved						T11	T10	Reserved				UPI	UAI	Reserved	NAKI	AS	PS	RCL	HCH	Reserved	ULPII	Reserved	SLI	SRI	URI	AAI	SEI	FRI
																												UI

Table 276. HW_USBCTRL_USBSTS Bit Field Descriptions

BITS	LABEL	RW	RESET*	DEFINITION
31:26	Reserved	RO		Reserved.
25	T11	RW		General-Purpose Timer Interrupt 1 (GPTINT1). This bit is set when the counter in the GPTIMER1CTRL (Non-EHCI) register transitions to 0. Writing a 1 to this bit will clear it.
24	T10	RW		General-Purpose Timer Interrupt 0 (GPTINT0). This bit is set when the counter in the GPTIMER0CTRL (Non-EHCI) register transitions to 0. Writing a 1 to this bit will clear it.
23:20	Reserved	RO		Reserved.
19	UPI	RW		USB Host Periodic Interrupt (USBHSTPERINT). This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be 0.

Table 276. HW_USBCTRL_USBSTS Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
18	UAI	RW		<p>USB Host Asynchronous Interrupt (USBHSTASYNCINT). This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule.</p> <p>This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes.</p> <p>This bit is not used by the device controller and will always be 0.</p>
17	Reserved	RO		Reserved.
16	NAKI	RO		<p>NAK Interrupt Bit. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared.</p>
15	AS	RO		<p>Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to 0 the asynchronous schedule status is disabled and if set to 1 the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0).</p> <p>Only used by the host controller.</p>
14	PS	RO		<p>Periodic Schedule Status. 0=Default. This bit reports the current real status of the Periodic Schedule. When set to 0 the periodic schedule is disabled, and if set to 1 the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0).</p> <p>Only used by the host controller.</p>
13	RCL	RO		<p>Reclamation. 0=Default. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller; 0 in device mode.</p>

STMP3770

Table 276. HW_USBCTRL_USBSTS Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
12	HCH	RO		<p>HCHalted. 1 = Default.</p> <p>This bit is a 0 whenever the Run/Stop bit is a 1. The Host Controller sets this bit to 1 after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error).</p> <p>Only used by the host controller; 0 in device mode.</p>
11	Reserved	RO		Reserved.
10	ULPII	RW		Not present in this implementation.
9	Reserved	RO		Reserved.
8	SLI	RW		<p>DCSuspend. 0 = Default.</p> <p>When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state.</p> <p>Only used by the device controller.</p>
7	SRI	RW		<p>SOF Received. 0 = Default.</p> <p>When the device controller detects a Start Of (micro) Frame, this bit will be set to a 1. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp.</p> <p>In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it.</p> <p>This is a non-EHCI status bit.</p>
6	URI	RW		<p>USB Reset Received. 0 = Default. When the device controller detects a USB Reset and enters the default state, this bit will be set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit.</p> <p>Only used by the device controller.</p> <p>NOTE: This bit should not normally be used to detect reset during suspend, as this block will normally be clock-gated during that time. Use HW_USBPHY_CTRL_RESUME_IRQ, instead.</p>

Table 276. HW_USBCTRL_USBSTS Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
5	AAI	RW		<p>Interrupt on Async Advance. 0 = Default.</p> <p>System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a 1 to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source.</p> <p>Only used by the host controller.</p>
4	SEI	RW		<p>System Error. This bit is not used in this implementation and will always be set to 0.</p>
3	FRI	RW		<p>Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles.</p> <p>Only used by the host controller.</p>
2	PCI	RW		<p>Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port.</p> <p>The Device Controller sets this bit to a 1 when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively.</p> <p>This bit is not EHCI compatible.</p>

Table 276. HW_USBCTRL_USBSTS Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
1	UEI	RW		USB Error Interrupt (USBERRINT). When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. See Section 4.15.1 in the EHCI specification for a complete list of host error interrupt conditions. The device controller detects resume signaling only.
0	UI	RW		USB Interrupt (USBINT). This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

9.6.19. USB Interrupt Enable Register Description

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

HW_USBCTRL_USBINTR 0x80080148

Table 277. HW_USBCTRL_USBINTR

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3
Reserved				TIE1	TIE0	Reserved				UPIE	UAIE	Reserved	NAKE	Reserved				ULPIE	Reserved	SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE

Table 278. HW_USBCTRL_USBINTR Bit Field Descriptions

BITS	LABEL	INTERRUPT SOURCE	RW	RESET	DEFINITION
31:26	Reserved	Reserved	RO	0x0	Reserved.
25	TIE1	General-Purpose Timer Interrupt Enable 1	RW	0x0	When this bit is a 1, and the GPTINT1 bit in the USBSTS register is a 1, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the GPTINT1 bit.

Table 278. HW_USBCTRL_USBINTR Bit Field Descriptions (Continued)

BITS	LABEL	INTERRUPT SOURCE	RW	RESET	DEFINITION
24	TIE0	General-Purpose Timer Interrupt Enable 0	RW	0x0	When this bit is a 1, and the GPTINT0 bit in the USBSTS register is a 1, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the GPTINT0 bit.
23:20	Reserved	Reserved.	RO	0x0	Reserved.
19	UPIE	USB Host Periodic Interrupt Enable	RW	0x0	When this bit is a 1, and the USBHSTPERINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTPERINT bit.
18	UAIE	USB Host Async. Interrupt Enable	RW	0x0	When this bit is a 1, and the USBHSTASYNCINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTASYNCINT bit.
17	Reserved	Reserved	RO	0x0	Reserved.
16	NAKE	NAK Interrupt Enable	RW	0x0	This bit is set by software if it wants to enable the hardware interrupt for the NAK Interrupt bit. If both this bit and the corresponding NAK Interrupt bit are set, a hardware interrupt is generated.
15:11	Reserved	Reserved	RO	0x0	Reserved.
10	ULPIE	ULPI Enable	RW	0x0	Not used in this implementation.
9	Reserved	Reserved	RO	0x0	Reserved.
8	SLE	Sleep Enable	RW	0x0	When this bit is a 1, and the DCSuspend bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a 1 to the DCSuspend bit. Only used by the device controller.
7	SRE	SOF Received Enable	RW	0x0	When this bit is a 1, and the SOF Received bit in the USBSTS register is a 1, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the SOF Received bit.
6	URE	USB Reset Enable	RW	0x0	When this bit is a 1, and the USB Reset Received bit in the USBSTS register is a 1, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller.
5	AAE	Interrupt on Async Advance Enable	RW	0x0	When this bit is a 1, and the Interrupt on Async Advance bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller.

Table 278. HW_USBCTRL_USBINTR Bit Field Descriptions (Continued)

BITS	LABEL	INTERRUPT SOURCE	RW	RESET	DEFINITION
4	SEE	System Error Enable	RW	0x0	When this bit is a 1, and the System Error bit in the USBSTS register is a 1, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the System Error bit.
3	FRE	Frame List Rollover Enable	RW	0x0	When this bit is a 1, and the Frame List Rollover bit in the USBSTS register is a 1, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller.
2	PCE	Port Change Detect Enable	RW	0x0	When this bit is a 1, and the Port Change Detect bit in the USBSTS register is a 1, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit.
1	UEE	USB Error Interrupt Enable	RW	0x0	When this bit is a 1, and the USBERRINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register.
0	UE	USB Interrupt Enable	RW	0x0	When this bit is a 1, and the USBINT bit in the USBSTS register is a 1, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit.

9.6.20. USB Frame Index Register Description

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the USBCMD register.

This register must be written as a DWord. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop bit is set to a 1 produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is Read-Only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to 0 (i.e., SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be incremented (i.e., SOF for 125 us micro-frame.)

* The default value of this register is undefined (free-running counter).

HW_USBCTRL_FRINDEX 0x8008014C

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0				
PERBASE[31:12]																				Reserved															

BITS	LABEL	RW	RESET	DEFINITION
31:12	PERBASE	RW	0x0	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
11:0	Reserved	RO	0x0	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

The USBADRA is used to accelerate the SET_ADDRESS sequence by allowing the DCD to preset the USBADR register before the status phase of the SET_ADDRESS descriptor. This is a read/write register. Writes must be DWORD writes.

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0					
USBADR[31:25]								USBADRA	Reserved																											

Table 284. HW_USBCTRL_DEVICEADDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DESCRIPTION
31:25	USBADR	RW	0x0	Device Address. These bits correspond to the USB device address.
24	USBADRA	RW	0x0	Device Address Advance. Default=0. When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1. IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2. OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3. Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
23:0	Reserved	RO	0x0	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

9.6.22. ASYNCLISTADDR and ENDPTLISTADDR Register Descriptions

This register is shared between the host controller and the device controller operation. This is a read/write register. Writes must be DWORD writes.

HW_USBCTRL_ENDPOINTLISTADDR 0x80080158

9.6.22.1. Host Controller Next Asynchronous Address Register

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a 0 when read.

Table 285. HW_USBCTRL_ASYNCLISTADDR

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ASYBASE[31:5]																								Reserved							

Table 289. HW_USBCTRL_TTCTRL

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0					
Reserved	TTHA							Reserved																												

Table 290. HW_USBCTRL_TTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	Reserved	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
30:24	TTHA	RW	0x0	Internal TT Hub Address Representation. Default is 0 (Read/Write). This field is used to match against the Hub Address field in QH & siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in the QH/siTD.
23:0	Reserved	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.

9.6.24. Programmable Burst Size Register Description

This register is used to control dynamically change the burst size used during data movement on the initiator (master) interface. This is a read/write register. Writes must be DWORD writes. The default value is 0x00001010.

HW_USBCTRL_BURSTSIZE 0x80080160

Table 291. HW_USBCTRL_BURSTSIZE

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
Reserved																TXPBURST								RXPBURST							

Table 292. HW_USBCTRL_BURSTSIZE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	Reserved	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
15:8	TXPBURST	RW	0x10	Programmable TX Burst Length. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
7:0	RXPBURST	RW	0x10	Programmable RX Burst Length. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

9.6.25. Host Transmit Pre-Buffer Packet Timing Register Description

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the (micro)frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the (micro)frame is $< T_s$ then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a “back-off” event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the TSCHEALTH (T_{ff}) described below.

This is a read/write register. Writes must be DWORD writes. The default value of this register is 0x00000000.

HW_USBCTRL_TXFILLTUNING 0x80080164

Table 293. HW_USBCTRL_TXFILLTUNING

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
Reserved											TXFIFOTHRES				Reserved				TXSCHEALTH				Reserved	TXSCHOH							

Table 294. HW_USBCTRL_TXFILLTUNING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:22	Reserved	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
21:16	TXFIFOTHRES	RW	0x0	FIFO Burst Threshold. This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be as low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
15:13	Reserved	RO	0x0	Reserved. These bits are reserved and their value has no effect on operation.
12:8	TXSCHHEALTH	RW C	0x0	Scheduler Health Counter. This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.

STMP3770

Table 294. HW_USBCTRL_TXFILLTUNING Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
7	Reserved	RO	0x0	Reserved. This bit is reserved and its value has no effect on operation.
6:0	TXSCHOH	RW	0x0	<p>Scheduler Overhead. This register adds an additional fixed offset to the schedule time estimator described above as T_{ff}. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization.</p> <p>The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode for OTG & SPH.</p> <p>The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode for OTG & SPH.</p> <p>The time unit represented in this register is always 1.267 in the MPH product.</p>

9.6.26. ULPI Viewport Register Description

This register is present but not used in this implementation.

HW_USBCTRL_ULPI

0x80080170

Table 295. HW_USBCTRL_ULPI

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
ULPIWU	ULPIRUN	ULPIRW	Reserved	ULPISS	ULPIPORT			ULPIADDR							ULPIDATRD							ULPIDATWR									

Table 296. HW_USBCTRL_ULPI Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	ULPIWU	RW	0x0	Not used. Read as 0.
30	ULPIRUN	RW	0x0	Not used. Read as 0.
29	ULPIRW	RW	0x0	Not used. Read as 0.
28	Reserved	RO	0x0	Not used. Read as 0.
27	ULPISS	RO	0x0	Not used. Read as 0.
26:24	ULPIPORT	RW	0x0	Not used. Read as 0.
23:16	ULPIADDR	RW	0x0	Not used. Read as 0.

9.6.28. Endpoint NAK Enable Register Description

Attribute: Read/Write Clear

HW_USBCTRL_ENDPTNAKEN 0x8008017C

Table 299. HW_USBCTRL_ENDPTNAKEN

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0		
Reserved											EPTNE[4:0]					Reserved													EPRNE[4:0]				

Table 300. HW_USBCTRL_ENDPTNAKEN Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	Reserved	RO	0x0	Reserved.
20:16	EPTNE[4:0]	RW	0x0	TX Endpoint NAK Enable. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. EPTNE[4] = Endpoint 4 EPTNE[1] = Endpoint 1 EPTNE[0] = Endpoint 0
15:5	Reserved	RO	0x0	Reserved.
4:0	EPRNE[4:0]	RW	0x0	RX Endpoint NAK Enable. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. EPRNE[4] = Endpoint 4 EPRNE[1] = Endpoint 1 EPRNE[0] = Endpoint 0

9.6.29. Port Status and Control 1 Register Description

Host Controller:

A host controller must implement one to eight port registers. The number of port registers implemented by a particular instantiation of a host controller is documented in the HCSPARAMs register and is fixed at 1 in this implementation. Software uses this information as an input parameter to determine how many ports need service.

This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are:

- No device connected
- Port disabled

If the port has port power control, this state remains until software applies power to the port by setting port power to 1.

Device Controller:

A device controller must implement only port register 1 and it does not support power control. Port control in device mode is only used for status port reset, sus-

* Default Value: 00010000000000000000XX0000000000b (Host mode)

```
000100000000000000000001XX0000000100b (Device mode)
```

X = Unknown

HW USBCTRL PORTSC1

0x80080184

Table 301. HW USBCTRL PORTSC1

PTS	STS	PTW	PSPD	Reserved	PFSC	PHCD	WKOC	WKDS	WKN	PTC[3:0]	PIC	PO	PP	LS	HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS							
3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0

Table 302. HW USBCTRL PORTSC1 Bit Field Descriptions

BITS	LABEL	RW	RESET*	DEFINITION
31:30	PTS	RW		Parallel Transceiver Select. For this implementation, always set to 00b for UTMI.
29	STS	RW		Serial Transceiver Select. Always 0.
28	PTW	RO		Parallel Transceiver Width. This bit is always 0, indicating an 8-bit (60-MHz) UTMI interface.
27:26	PSPD	RO		Port Speed. This register field indicates the speed at which the port is operating. For high-speed mode operation in the host controller and high-speed/full-speed operation in the device controller, the port routing steers data to the protocol engine. 0x0 = FULL (Full Speed). 0x1 = HIGH (High Speed). This bit is not defined in the EHCI specification.
25	Reserved	RO		Reserved.
24	PFSC	RW		Port Force Full Speed Connect. Default = 0. Writing this bit to a 1 will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as high-speed. This is useful for testing full-speed configurations with a high-speed host, hub or device. This bit is not defined in the EHCI specification. This bit is for debugging purposes.

Table 302. HW_USBCTRL_PORTSC1 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
23	PHCD	RW		<p>PHY Low Power Suspend - Clock Disable (PLPSCD). Default = 0. Writing this bit to a 1 will disable the PHY clock. Writing a 0 enables it. Reading this bit will indicate the status of the PHY clock.</p> <p><u>In Device Mode:</u> The PHY can be put into Low Power Suspend – Clock Disable when the device is not running (USBCMD Run/Stop=0) or the host has signaled suspend (PORTSC SUSPEND=1). Low-power suspend will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the device controller driver must clear this bit.</p> <p><u>In Host Mode:</u> The PHY can be put into Low Power Suspend – Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. This bit is not defined in the EHCI specification.</p>
22	WKOC	RW		<p>Wake on Over-current Enable (WKOC_E). Default = 0.</p> <p>Writing this bit to a 1 enables the port to be sensitive to over-current conditions as wake-up events. This field is 0 if Port Power (PP) is 0.</p>
21	WKDC	RW		<p>Wake on Disconnect Enable (WKDSCNNT_E). Default=0.</p> <p>Writing this bit to a 1 enables the port to be sensitive to device disconnects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode.</p>
20	WKCEN	RW		<p>Wake on Connect Enable (WKCENNT_E). Default=0.</p> <p>Writing this bit to a 1 enables the port to be sensitive to device connects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode.</p>

Table 302. HW_USBCTRL_PORTSC1 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION																				
19:16	PTC[3:0]	RW		<p>Port Test Control. Default = 0000b. Any other value than 0 indicates that the port is operating in test mode.</p> <table><tr><td>Value</td><td>Specific Test</td></tr><tr><td>0000b</td><td>TEST_DISABLE</td></tr><tr><td>0001b</td><td>TEST_J_STATE</td></tr><tr><td>0010b</td><td>TEST_K_STATE</td></tr><tr><td>0011b</td><td>TEST_J_SE0_NAK for SE0 (host)/NAK (device)</td></tr><tr><td>0100b</td><td>TEST_PACKET</td></tr><tr><td>0101b</td><td>TEST_FORCE_ENABLE_HS</td></tr><tr><td>0110b</td><td>TEST_FORCE_ENABLE_FS</td></tr><tr><td>0111b</td><td>TEST_FORCE_ENABLE_LS</td></tr><tr><td>1000b–1111b</td><td>Reserved</td></tr></table> <p>Refer to Chapter 9 of the USB Specification Revision 2.0 for details on each test mode. The TEST_FORCE_ENABLE_FS and TEST_FORCE_ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the TEST_FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_DISABLE will allow the port state machines to progress normally from that point. Note: Low speed operations are not supported.</p>	Value	Specific Test	0000b	TEST_DISABLE	0001b	TEST_J_STATE	0010b	TEST_K_STATE	0011b	TEST_J_SE0_NAK for SE0 (host)/NAK (device)	0100b	TEST_PACKET	0101b	TEST_FORCE_ENABLE_HS	0110b	TEST_FORCE_ENABLE_FS	0111b	TEST_FORCE_ENABLE_LS	1000b–1111b	Reserved
Value	Specific Test																							
0000b	TEST_DISABLE																							
0001b	TEST_J_STATE																							
0010b	TEST_K_STATE																							
0011b	TEST_J_SE0_NAK for SE0 (host)/NAK (device)																							
0100b	TEST_PACKET																							
0101b	TEST_FORCE_ENABLE_HS																							
0110b	TEST_FORCE_ENABLE_FS																							
0111b	TEST_FORCE_ENABLE_LS																							
1000b–1111b	Reserved																							
15:14	PIC[1:0]	RW		<p>Port Indicator Control. Default = 0.</p> <table><tr><td>00b</td><td>OFF (Port indicators are off)</td></tr><tr><td>01b</td><td>AMBER</td></tr><tr><td>10b</td><td>GREEN</td></tr><tr><td>11b</td><td>UNDEF (Undefined)</td></tr></table> <p>Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used.</p>	00b	OFF (Port indicators are off)	01b	AMBER	10b	GREEN	11b	UNDEF (Undefined)												
00b	OFF (Port indicators are off)																							
01b	AMBER																							
10b	GREEN																							
11b	UNDEF (Undefined)																							

Table 302. HW_USBCTRL_PORTSC1 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
13	PO	RO		Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always read back as 0. The EHCI definition is include here for reference: Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is 0. System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a 1 to this bit when the attached device is not a high-speed device. A 1 in this bit means that an internal companion controller owns and controls the port.
12	PP	RW or RO		Port Power (PP). This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port, the PP bit in each affected port may be transitioned by the host controller driver from a 1 to a 0 (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device implementation, port power control is not necessary.
11:10	LS[1:0]	RO		Line Status. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits is: 00b SE0 10b J-STATE 01b K-STATE 11b UNDEF (Undefined) <u>In Host Mode:</u> The use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. <u>In Device Mode:</u> The use of linestate by the device controller driver is not necessary.
9	HSP	RO		High-Speed Port. Default = 0. When the bit is 1, the host/device connected to the port is in high-speed mode and if set to 0, the host/device connected to the port is not in a high-speed mode. Note: HSP is redundant with PSPD(27:26) but will remain in the design for compatibility. This bit is not defined in the EHCI specification.

Table 302. HW_USBCTRL_PORTSC1 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION								
8	PR	RW or RO		<p>Port Reset</p> <p>This field is 0 if Port Power (PP) is 0.</p> <p><u>In Host Mode:</u> (Read/Write). 1=Port is in Reset. 0=Port is not in Reset. Default 0.</p> <p>When software writes a 1 to this bit, the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to 0 after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a 0 after the reset duration is timed in the driver.</p> <p><u>In Device Mode:</u> This bit is a Read-Only status bit. Device reset from the USB bus is also indicated in the USBSTS register.</p>								
7	SUSP	RW or RO		<p>Suspend</p> <p><u>In Host Mode:</u> (Read/Write)</p> <p>0 = Port not in suspend state.</p> <p>1 = Port in suspend state.</p> <p>Default = 0.</p> <p>Port Enabled Bit and Suspend bit of this register define the port states as follows:</p> <table><tr><td>Bits</td><td>Port State</td></tr><tr><td>0x</td><td>Disable</td></tr><tr><td>10</td><td>Enable</td></tr><tr><td>11</td><td>Suspend</td></tr></table> <p>When in suspend state, the downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.</p> <p>The host controller will unconditionally set this bit to 0 when software sets the Force Port Resume bit to 0. The host controller ignores a write of 0 to this bit. If host software sets this bit to a 1 when the port is not enabled (i.e., Port enabled bit is a 0) the results are undefined.</p> <p>This field is 0 if Port Power (PP) is 0 in host mode.</p> <p><u>In Device Mode:</u> (Read-Only)</p> <p>1=Port in suspend state. 0=Port not in suspend state. Default=0.</p> <p>In device mode, this bit is a Read-Only status bit.</p>	Bits	Port State	0x	Disable	10	Enable	11	Suspend
Bits	Port State											
0x	Disable											
10	Enable											
11	Suspend											

Table 302. HW_USBCTRL_PORTSC1 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
6	FPR	RW		<p>Force Port Resume. 0 = No resume (K-state) detected/driven on port. 1 = Resume detected/driven on port. Default = 0.</p> <p><u>In Host Mode:</u> Software sets this bit to 1 to drive resume signaling. The Host Controller sets this bit to 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a 1 because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to 1. This bit will automatically change to 0 after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a 0 after the resume duration is timed in the driver.</p> <p>Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a 1. This bit will remain a 1 until the port has switched to the high-speed idle. Writing a 0 has no effect because the port controller will time the resume operation and clear the bit when the port control state switches to HS or FS idle. This field is 0 if Port Power (PP) is 0 in host mode. This bit is not-EHCI compatible.</p> <p><u>In Device Mode:</u> After the device has been in Suspend State for 5 ms or more, software must set this bit to 1 to drive resume signaling before clearing. The Device Controller will set this bit to 1 if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a 1 because a J-to-K transition has been detected, the Port Change Detect bit in the USBSTS register is also set to 1.</p>
5	OCC	RW		<p>Over-Current Change. 0 = Default. 1 = This bit gets set to 1 when there is a change to Over-current Active. Software clears this bit by writing a 1 to this bit position.</p> <p>For host/OTG implementations, the user can provide over-current detection to the vbus_pwr_fault input for this condition.</p> <p>For device-only implementations, this bit shall always be 0.</p>

Table 302. HW_USBCTRL_PORTSC1 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
4	OCA	RO		<p>Over-Current Active. 0 = This port does not have an over-current condition. 1 = This port currently has an over-current condition. Default = 0. This bit will automatically transition from 1 to 0 when the over current condition is removed. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.</p>
3	PEC	RW		<p>Port Enable/Disable Change. 0=No change. 1=Port enabled/disabled status has changed. Default = 0. <u>In Host Mode:</u> For the root hub, this bit gets set to a 1 only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a 1 to it. This field is 0 if Port Power (PP) is 0. <u>In Device Mode:</u> The device port is always enabled. (This bit will be 0)</p>
2	PE	RW		<p>Port Enabled/Disabled. 0=Disable. 1=Enable. Default = 0. <u>In Host Mode:</u> Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a 1 to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0) downstream propagation of data is blocked except for reset. This field is 0 if Port Power (PP) is 0 in host mode. <u>In Device Mode:</u> The device port is always enabled. (This bit will be 1)</p>

Table 302. HW_USBCTRL_PORTSC1 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET*	DEFINITION
1	CSC	RW		Connect Status Change. 0 = No change. 1 = Change in Current Connect Status. Default = 0. <u>In Host Mode:</u> Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a 1 to it. This field is 0 if Port Power (PP) is 0 in host mode. <u>In Device Mode:</u> This bit is undefined in device controller mode.
0	CCS	RO		Current Connect Status. <u>In Host Mode:</u> 0 = No device is present. 1 = Device is present on port. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is 0 if Port Power (PP) is 0 in host mode. <u>In Device Mode:</u> 0 = Not Attached. 1 = Attached. Default = 0. A 1 indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A 0 indicates that the device did not attach successfully or was forcibly disconnected by the software writing a 0 to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

9.6.30. OTG Status and Control Register Description

Host Controller:

A host controller implements one On-The-Go (OTG) Status and Control register corresponding to Port 0 of the host controller. The OTGSC register has four sections:

- OTG Interrupt Enables (Read/Write)
- OTG Interrupt Status (Read/Write to Clear)
- OTG Status Inputs (Read-Only)
- OTG Controls (Read/Write)

The status inputs are debounced using a 1-ms time constant. Values on the status inputs that do not persist for more than 1 ms will not cause an update of the status

input register, or cause an OTG interrupt. See also USBMODE register. The default value of this register is 0x00000020.

HW_USBCTRL_OTGSC

0x800801A4

Table 303. HW_USBCTRL_OTGSC

Reserved	3 1	DPIE	3 0	ONEMSE	2 9	BSEIE	2 8	BSVIE	2 7	ASVIE	2 6	AVVIE	2 5	IDIE	2 4	Reserved	2 3	DPIS	2 2	ONEMSS	2 1	BSEIS	2 0	BSVIS	1 9	ASVIS	1 8	AVVIS	1 7	IDIS	1 6	Reserved	1 5	DPS	1 4	ONEMST	1 3	BSE	1 2	BSV	1 1	ASV	1 0	AVV	0 9	ID	0 8	HABA	0 7	HADP	0 6	IDPU	0 5	DP	0 4	OT	0 3	HAAR	0 2	VC	0 1	VD	0 0
----------	--------	------	--------	--------	--------	-------	--------	-------	--------	-------	--------	-------	--------	------	--------	----------	--------	------	--------	--------	--------	-------	--------	-------	--------	-------	--------	-------	--------	------	--------	----------	--------	-----	--------	--------	--------	-----	--------	-----	--------	-----	--------	-----	--------	----	--------	------	--------	------	--------	------	--------	----	--------	----	--------	------	--------	----	--------	----	--------

Table 304. HW_USBCTRL_OTGSC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	Reserved	RO	0x0	Reserved.
30	DPIE	RW	0x0	Data Pulse Interrupt Enable
29	ONEMSE	RW	0x0	1 Millisecond Timer Interrupt Enable
28	BSEIE	RW	0x0	B Session End Interrupt Enable. Setting this bit enables the B session end interrupt.
27	BSVIE	RW	0x0	B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt.
26	ASVIE	RW	0x0	A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt.
25	AVVIE	RW	0x0	A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt.
24	IDIE	RW	0x0	USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt.
23	Reserved	RO	0x0	Reserved.
22	DPIS	RW	0x0	Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software must write a 1 to clear this bit.
21	ONEMSS	RW	0x0	1 Millisecond Timer Interrupt Status. This bit is set once every millisecond. Software must write a 1 to clear this bit.
20	BSEIS	RW	0x0	B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software must write a 1 to clear this bit
19	BSVIS	RW	0x0	B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.

STMP3770

Table 304. HW_USBCTRL_OTGSC Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
18	ASVIS	RW	0x0	A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.
17	AVVIS	RW	0x0	A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a 1 to clear this bit.
16	IDIS	RW	0x0	USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software must write a 1 to clear this bit.
15	Reserved	RO	0x0	Reserved.
14	DPS	RO	0x0	Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port.
13	ONEMST	RO	0x0	1 Millisecond Timer Toggle. This bit toggles once per millisecond.
12	BSE	RO	0x0	B Session End. Indicates VBus is below the B session end threshold.
11	BSV	RO	0x0	B Session Valid. Indicates VBus is above the B session valid threshold.
10	ASV	RO	0x0	A Session Valid. Indicates VBus is above the A session valid threshold.
9	AVV	RO	0x0	A VBus Valid. Indicates VBus is above the A VBus valid threshold.
8	ID	RO	0x0	USB ID. 0 = A device, 1 = B device
7	HABA	RW	0x0	Hardware Assist B-Disconnect to A-connect. 0 = Disabled. 1 = Enable automatic B-disconnect to A-connect sequence.
6	HADP	RW S	0x0	Hardware Assist Data-Pulse – Write to Set. 1 = Start Data Pulse Sequence.
5	IDPU	RW	0x1	ID Pullup. This bit provide control over the ID pullup resistor; 0 = off, 1 = on (default). When this bit is 0, the ID input will not be sampled.
4	DP	RW	0x0	Data Pulsing. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3	OT	RW	0x0	OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2	HAAR	RW	0x0	Hardware Assist Auto-Reset. 0 = Disabled. 1 = Enable automatic reset after connect on host port.

Table 306. HW_USBCTRL_USBMODE Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
4	SDIS	RW	0x0	<p>Stream Disable Mode. 0 = Inactive (default) 1 = Active</p> <p><u>In Device Mode:</u> Setting to a 1 disables double priming on both RX and TX for low bandwidth systems. This mode, when enabled, ensures that the RX and TX buffers are sufficient to contain an entire packet, so that the usual double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active.</p> <p><u>In Host Mode:</u> Setting to a 1 ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB.</p> <p>Note: Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TXFILLTUNING and TXTTFILLTUNING to characterize the adjustments needed for the scheduler when using this feature.</p> <p>Note: The use of this feature substantially limits the overall USB performance that can be achieved.</p>
3	SLOM	RW	0x0	<p>Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. 0 = Setup Lockouts On (default) 1 = Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USBCMD)</p>

Table 306. HW_USBCTRL_USBMODE Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
2	ES	RW	0x0	<p>Endian Select. This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words.</p> <p>0 = Little Endian (default): First byte referenced in least significant byte of 32-bit word.</p> <p>1 = Big Endian: First byte referenced in most significant byte of 32-bit word.</p>
1:0	CM[1:0]	RW	0x0	<p>Controller Mode. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register.</p> <p>00b = IDLE (default for combination host/device)</p> <p>10b = DEVICE (default for device-only controller)</p> <p>11b = HOST (default for host-only controller)</p>

Table 314. HW_USBCTRL_ENDPTSTAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	Reserved	RO	0x0	Reserved.
20:16	ETBR[4:0]	RO	0x0	<p>Endpoint Transmit Buffer Ready. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a 1 by the hardware as a response to receiving a command from a corresponding bit in the ENDTPRIME register. There will always be a delay between setting a bit in the ENDTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.</p> <p>Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.</p> <p>ETBR[4] = Endpoint 4 ETBR[1] = Endpoint 1 ETBR[0] = Endpoint 0</p>
15:5	Reserved	RO	0x0	Reserved.
4:0	ERBR[4:0]	RO	0x0	<p>Endpoint Receive Buffer Ready. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a 1 by the hardware as a response to receiving a command from a corresponding bit in the ENDTPRIME register. There will always be a delay between setting a bit in the ENDTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.</p> <p>Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.</p> <p>ERBR[4] = Endpoint 4 ERBR[1] = Endpoint 1 ERBR[0] = Endpoint 0</p>

9.6.36. Endpoint Compete Register Description

This register is used in device-mode only.

HW_USBCTRL_ENDPTCOMPLETE 0x800801BC

STMP3770

Table 315. HW_USBCTRL_ENDPTCOMPLETE

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
Reserved											ETCE[4:0]					Reserved											ERCE[4:0]				

Table 316. HW_USBCTRL_ENDPTCOMPLETE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	Reserved	RO	0x0	Reserved.
20:16	ETCE[4:0]	RW	0x0	Endpoint Transmit Complete Event. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a 1 will clear the corresponding bit in this register. ETCE[4] = Endpoint 4 ETCE[1] = Endpoint 1 ETCE[0] = Endpoint 0
15:5	Reserved	RO	0x0	Reserved.
4:0	ERCE[4:0]	RW	0x0	Endpoint Receive Complete Event. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a 1 will clear the corresponding bit in this register. ERCE[4] = Endpoint 4 ERCE[1] = Endpoint 1 ERCE[0] = Endpoint 0

9.6.37. Endpoint Control 0 Register Description

Every Device will implement Endpoint0 as a control endpoint. The default value of this register is 0x0080008.

HW_USBCTRL_ENDPTCTRL0 0x800801C0

Table 317. HW_USBCTRL_ENDPTCTRL0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0		
Reserved								TXE	Reserved				TXT		Reserved	TXS	Reserved								RXE	Reserved				RXT		Reserved	RXS

Table 318. HW_USBCTRL_ENDPTCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	Reserved	RO	0x0	Reserved.
23	TXE	RO	0x0	TX Endpoint Enable. 1 = Enabled Endpoint0 is always enabled.
22:20	Reserved	RO	0x0	Reserved. Bit reserved and should be read as zeroes.
19:18	TXT[1:0]	RW	0x2	TX Endpoint Transmit Type. 00 = CONTROL Endpoint0 is fixed as a Control endpoint.
17	Reserved	RO	0x0	Reserved.
16	TXS	RW	0x0	TX Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Note: There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.
15:8	Reserved	RO	0x0	Reserved. Bit reserved and should be read as zeroes.
7	RXE	RO	0x0	RX Endpoint Enable. 1 = Enabled Endpoint0 is always enabled.
6:4	Reserved	RO	0x0	Reserved. Bit reserved and should be read as zeroes.
3:2	RXT[1:0]	RW	0x2	RX Endpoint Receive Type. 00 = CONTROL Endpoint0 is fixed as a Control endpoint.

Table 318. HW_USBCTRL_ENDPTCTRL0 Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
1	Reserved	RO	0x0	Reserved.
0	RXS	RW	0x0	<p>RX Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled.</p> <p>Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.</p> <p>After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.</p> <p>Note: There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.</p>

9.6.38. Endpoint Control 1–4 Registers Description

There is an ENDPTCTRLx register for each endpoint in a device.

CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

HW_USBCTRL_ENDPTCTRL1	0x800801C4
HW_USBCTRL_ENDPTCTRL2	0x800801C8
HW_USBCTRL_ENDPTCTRL3	0x800801CC
HW_USBCTRL_ENDPTCTRL4	0x800801D0

Table 319. HW_USBCTRL_ENDPTCTRLn[1:4]

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
Reserved								TXE	TXR	TXI	Reserved	TXT	TXD	TXS	Reserved								RXE	RXR	RXI	Reserved	RXT	RXD	RXS		

Table 320. HW_USBCTRL_ENDPTCTRLn[1:4] Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	Reserved	RO	0x0	Reserved.
23	TXE	RW	0x0	TX Endpoint Enable. 0 = Disabled (default). 1 = Enabled. An endpoint should be enabled only after it has been configured.
22	TXR	WS	0x0	TX Data Toggle Reset. Write 1 to reset PID sequence. Whenever a configuration event is received for this endpoint, software must write a 1 to this bit in order to synchronize the data PIDs between the host and device.
21	TXI	RW	0x0	TX Data Toggle Inhibit. 0 = PID Sequencing Enabled (default) 1 = PID Sequencing Disabled This bit is only used for test and should always be written as 0. Writing a 1 to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20	Reserved	RO	0x0	Reserved.
19:18	TXT[1:0]	RW	0x0	TX Endpoint Transmit Type. 00b = CONTROL 01b = ISO (Isochronous) 10b = BULK 11b = INT (Interrupt)
17	TXD	RW	0x0	TX Endpoint Data Source. 0 = Dual Port Memory Buffer/DMA Engine (default) Should always be written as 0.

Table 320. HW_USBCTRL_ENDPTCTRLn[1:4] Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
16	TXS	RW	0x0	<p>TX Endpoint Stall. 0 = Endpoint OK 1 = Endpoint Stalled</p> <p>This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.</p> <p>Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints.</p> <p>Note (control endpoint types only): There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, Should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.</p>
15:8	Reserved	RO	0x0	Reserved.
7	RXE	RW	0x0	<p>RX Endpoint Enable. 0 = Disabled (default) 1 = Enabled</p> <p>An Endpoint should be enabled only after it has been configured.</p>
6	RXR	WS	0x0	<p>RX Data Toggle Reset. Write 1 to reset PID Sequence.</p> <p>Whenever a configuration event is received for this endpoint, software must write a 1 to this bit in order to synchronize the data PIDs between the host and device.</p>
5	RXI	RW	0x0	<p>RX Data Toggle Inhibit. 0 = Disabled (default). 1 = Enabled.</p> <p>This bit is only used for test and should always be written as 0. Writing a 1 to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.</p>
4	Reserved	RO	0x0	Reserved.
3:2	RXT[1:0]	RW	0x0	<p>RX Endpoint ReceiveType. 00b = CONTROL 01b = ISO (Isochronous) 10b = BULK 11b = INT (Interrupt)</p>

Table 320. HW_USBCTRL_ENDPTCTRLn[1:4] Bit Field Descriptions (Continued)

BITS	LABEL	RW	RESET	DEFINITION
1	RXD	RW	0x0	RX Endpoint Data Sink. 0 = Dual Port Memory Buffer/DMA Engine (default) Should always be written as 0.
0	RXS	RW	0x0	RX Endpoint Stall. 0 = Endpoint OK (default) 1 = Endpoint Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. Note (control endpoint types only): There is a slight delay (50 clocks maximum) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.

STMP3770



10. INTEGRATED USB 2.0 PHY

This chapter describes the integrated USB 2.0 full-speed and high-speed PHY available on the STMP3770. Programmable registers are described in [Section 10.4](#).

10.1. Overview

The STMP3770 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 27](#) for a block diagram of the PHY. The integrated PHY provides a standard UTM interface. The USB_DP and USB_DN pins connect directly to a USB device connector.

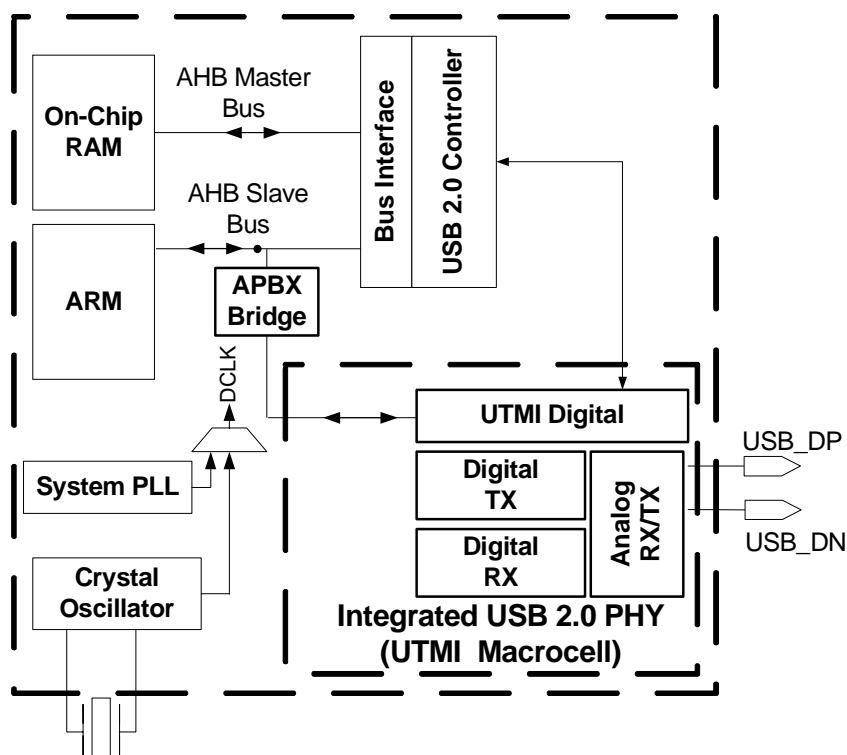


Figure 27. USB 2.0 PHY Block Diagram

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

10.2. Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in [Figure 28](#).

10.2.1. *UTMI*

The UTMI 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.

10.2.2. *Digital Transmitter*

The digital transmitter 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.

10.2.3. *Digital Receiver*

The digital receiver 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 nine 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.).

10.2.4. *Analog Receiver*

The analog receiver comprises five differential receivers and two single-ended receivers, as shown in [Figure 28](#) and described further in this section.

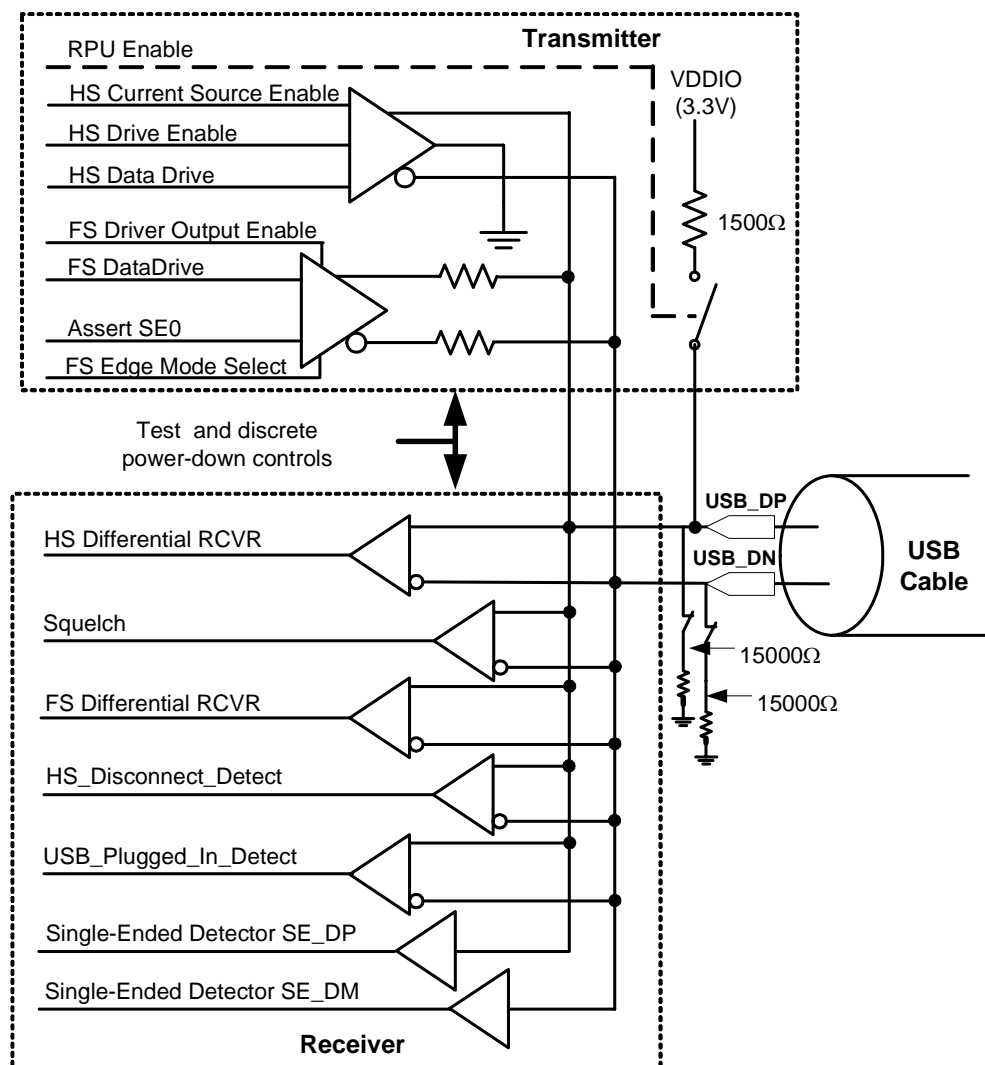


Figure 28. USB 2.0 PHY Analog Transceiver Block Diagram

10.2.4.1. 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 0, 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.

10.2.4.2. Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator. Its output is a 1 if the differential magnitude is less than a nominal 100-mV threshold.

Its output is 0, otherwise. Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

10.2.4.3. 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 1 when the USB_DP line is above the crossover point and the USB_DN line is below the crossover point.

10.2.4.4. HS Disconnect Detector

This host-side function is not used in STMP3770 applications, but is included to make a complete UTMI macrocell. It is a differential analog receiver and threshold comparator. Its output is a 1 if the differential magnitude is greater than a nominal 575-mV threshold. Its output is 0, otherwise.

10.2.4.5. USB Plugged-In Detector

The USB plugged-in detector looks for both USB_DP and USB_DN to be high. There is a pair of large on-chip pullup resistors (200K Ω) that hold both USB_DP and USB_DN high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case.

When in device mode, the host/hub interface that is *upstream* from the STMP3770 contains a 15K Ω pulldown resistor that easily overrides the 200K Ω pullup. When plugged in, at least one signal in the pair will be low, which will force the plugged-in detector's output high.

10.2.4.6. Single-Ended USB_DP Receiver

The single-ended USB_DP receiver output is high whenever the USB_DP input is above its nominal 1.8-V threshold.

10.2.4.7. Single-Ended USB_DN Receiver

The single-ended USB_DN receiver output is high whenever the USB_DN input is above its nominal 1.8-V threshold.

10.2.5. Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5K Ω pullup resistor. See [Figure 28](#).

10.2.5.1. Switchable High-Speed 45 Ω Termination Resistors

High-speed current mode differential signaling requires good 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable. Because each signal is parallel terminated with 45 Ω at each end, each driver sees a 22.5 Ω 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 29](#). The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45 Ω terminator on the USB_DP signal.

10.2.5.2. 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.

10.2.5.3. 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 USB_DP signal or the USB_DN 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.

10.2.5.4. Switchable $1.5K\Omega$ USB_DP Pullup Resistor

The STMP3770 contains a switchable $1.5K\Omega$ pullup resistor on the USB_DP 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.

10.2.5.5. Switchable $15K\Omega$ USB_DP Pulldown Resistor

The STMP3770 contains a switchable $15K\Omega$ pulldown resistor on both USB_DP and USB_DN signals. This is used in host mode to tell the device controller that a host is present.

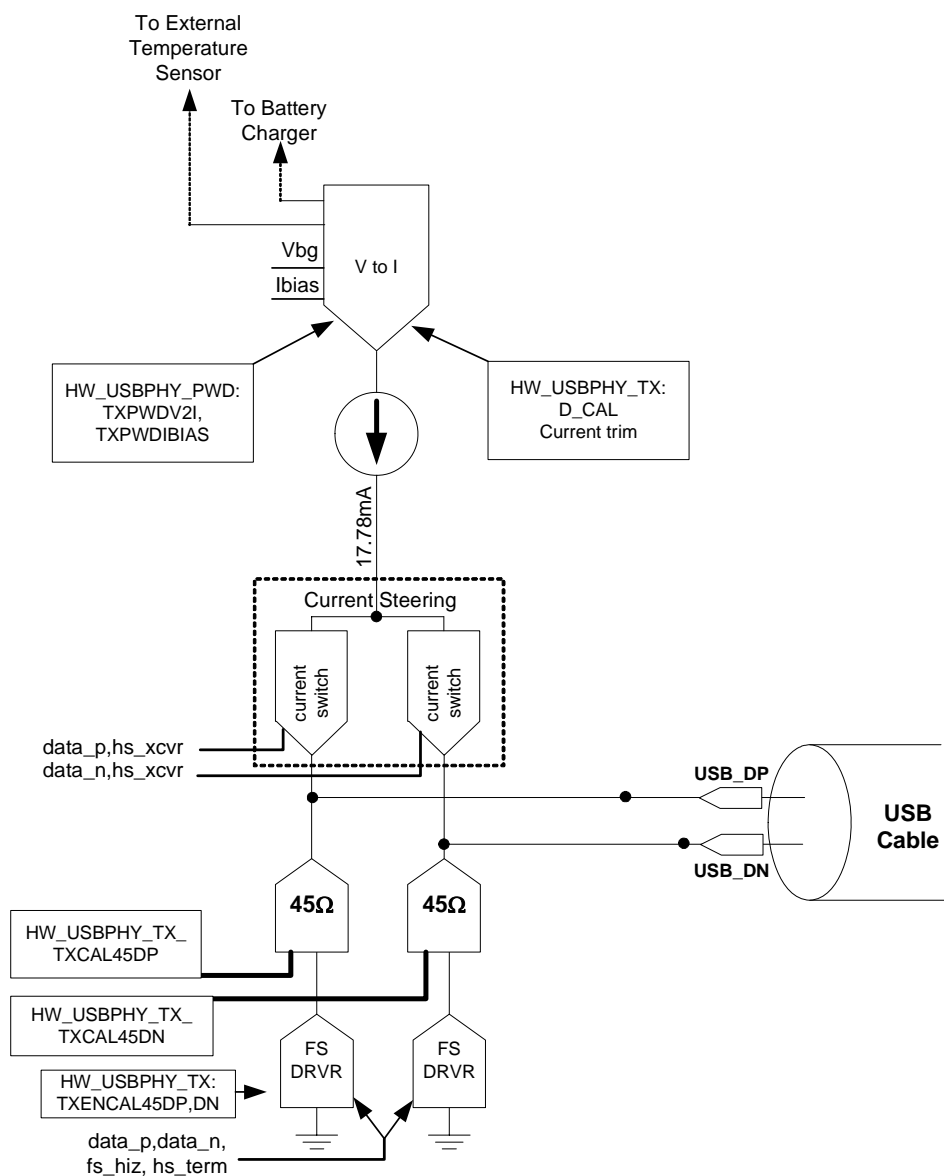


Figure 29. USB 2.0 PHY Transmitter Block Diagram

Table 321 summarizes the response of the PHY analog transmitter to various states of UTMI input and key transmit/receive state machine states.

Table 321. USB PHY Terminator States

UTMI OPMODE	UTM TERM	UTM XCVR	T/R	FUNCTION	45 Ω HIZ	1500 Ω HIZ	
00=Normal	0	0	X	HS	0	1	SUSPEND
	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	
01=NoDrive	0	0	T	HS	1	1	POR
	0	0	R	HS	1	1	
	1	1	X	FS	1	1	
	1	0	X	CHIRP	1	1	
	0	1	X	DISCONNECT	1	1	
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	

10.2.6. Recommended Register Configuration for USB Certification

The register settings in this section are recommended for passing USB certification. The following settings lower the J/K levels to certifiable limits:

HW_USBPHY_TX_TXCAL45DP = 0x0

HW_USBPHY_TX_TXCAL45DN = 0x0

HW_USBPHY_TX_D_CAL = 0x7

Note that HW_AUDIOOUT_REFCTRL_VDDXTAL_TO_VDDD is controlled by the SFTRST and CLKGATE bits in the AUDIOIN block.

10.3. 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 34.4.10, "Correct Way to Soft Reset a Block" on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

Table 323. HW_USBPHY_PWD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11	TXPWDIBIAS	RW	0x1	0 = Normal operation. 1 = Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path. Note that these circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
10	TXPWDFS	RW	0x1	0 = Normal operation. 1 = Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output.
9:5	RSVD	RO	0x0	Reserved.
4:0	RSVD	RO	0x0	Reserved.

10.4.2. USB PHY Transmitter Control Register Description

The USB PHY Transmitter Control Register handles the transmit controls.

HW_USBPHY_TX	0x8007C010
HW_USBPHY_TX_SET	0x8007C014
HW_USBPHY_TX_CLR	0x8007C018
HW_USBPHY_TX_TOG	0x8007C01C

Table 324. HW USBPHY TX

RSVD	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBPHY_TX_EDGECTRL																																
USBPHY_TX_SYNC_INVERT																																
USBPHY_TX_SYNC_MUX																																
TXCMPOUT_STATUS																																
RSVD																																
TXENCAL45DP																																
RSVD																																
TXCAL45DP																																
RSVD																																
TXENCAL45DN																																
RSVD																																
TXCAL45DN																																
TXCALIBRATE																																
RSVD																																
D_CAL																																

Table 325. HW_USBPHY_TX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x0	Reserved.
28:26	USBPHY_TX_EDGECTRL	RW	0x4	Controls the edge-rate of the current sensing transistors used in HS transmit. NOT FOR CUSTOMER USE.

STMP3770**Table 325. HW_USBPHY_TX Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
25	USBPHY_TX_SYNC_INVERT	RW	0x0	Changes clock edge that sync mux will use when USBPHY_TX_SYNC_MUX is high. NOT FOR CUSTOMER USE.
24	USBPHY_TX_SYNC_MUX	RW	0x0	Enables multiplexer to synchronize data from the USB_DP and USB_DM pins 0 = No sync, 1 = Sync. NOT FOR CUSTOMER USE.
23	TXCMPOUT_STATUS	RW	0x0	This bit is not used.
22	RSVD	RO	0x0	Reserved.
21	TXENCAL45DP	RW	0x0	This bit is not used and must remain cleared.
20	RSVD	RO	0x0	Reserved.
19:16	TXCAL45DP	RW	0x6	Decode to select a 45-Ohm resistance to the USB_DP output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
15:14	RSVD	RO	0x0	Reserved.
13	TXENCAL45DN	RW	0x0	This bit is not used and must remain cleared.
12	RSVD	RO	0x0	Reserved.
11:8	TXCAL45DN	RW	0x6	Decode to select a 45-Ohm resistance to the USB_DN output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
7	TXCALIBRATE	RW	0x0	This bit is not used and must remain cleared.
6:4	RSVD	RO	0x0	Reserved.
3:0	D_CAL	RW	0x7	Resistor Trimming Code: 0000 = 0.16% 0111 = Nominal 1111 = +25%

10.4.3. 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 326. HW_USBPHY_RX

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD									RXDBYPASS	RSVD								RSVD				RSVD	DISCONADJ	RSVD	ENVADJ						

Table 327. HW_USBPHY_RX Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:23	RSVD	RO	0x0	Reserved.
22	RXDBYPASS	RW	0x0	0 = Normal operation. 1 = Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver. This test mode is intended for lab use only.
21:16	RSVD	RO	0x0	Reserved.
15:8	RSVD	RO	0x0	Reserved.
7:6	RSVD	RO	0x0	Reserved.
5:4	DISCONADJ	RW	0x0	The DISCONADJ field adjusts the trip point for the disconnect detector: 0000 = Trip-Level Voltage is 0.57500 V 0001 = Trip-Level Voltage is 0.56875 V 0010 = Trip-Level Voltage is 0.58125 V 0011 = Trip-Level Voltage is 0.58750 V 01XX = Reserved 1XXX = Reserved
3:2	RSVD	RO	0x0	Reserved.
1:0	ENVADJ	RW	0x0	The ENVADJ field adjusts the trip point for the envelope detector. 0000 = Trip-Level Voltage is 0.12500 V 0001 = Trip-Level Voltage is 0.10000 V 0010 = Trip-Level Voltage is 0.13750 V 0011 = Trip-Level Voltage is 0.15000 V 01XX = Reserved 1XXX = Reserved

10.4.4. 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 detection enables and results.

HW_USBPHY_CTRL	0x8007C030
HW_USBPHY_CTRL_SET	0x8007C034
HW_USBPHY_CTRL_CLR	0x8007C038
HW_USBPHY_CTRL_TOG	0x8007C03C

STMP3770

Table 328. HW_USBPHY_CTRL

SFTRST	31	
CLKGATE	30	
UTMI_SUSPENDM	29	
HOST_FORCE_LS_SE0	28	
RSVD	27	
	26	
	25	
	24	
	23	
	22	
	21	
	20	
	19	
	18	
	17	
	16	
	15	
	14	
	DATA_ON_LRADC	13
	DEVPLUGIN_IRQ	12
ENIRQDEVPLUGIN	11	
RESUME_IRQ	10	
ENIRQRESUMEDETECT	9	
RSVD	8	
ENOTGIDDETECT	7	
RSVD	6	
DEVPLUGIN_POLARITY	5	
ENDEVPLUGINDETECT	4	
HOSTDISCONDETECT_IRQ	3	
ENIRQHOSTDISCON	2	
ENHOSTDISCONDETECT	1	
ENHSPRECHARGE_XMIT	0	

Table 329. HW_USBPHY_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	Writing a 1 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. Clear to 0 to run clocks. Set to 1 to gate 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	HOST_FORCE_LS_SE0	RW	0x0	Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the HW_USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with HW_USBPHY_DEBUG_HOST_RESUME_DEBUG.
27:14	RSVD	RO	0x0	Reserved.
13	DATA_ON_LRADC	RW	0x0	Enables the LRADC to monitor USB_DP and USB_DM. This is for use in non-USB modes only.
12	DEVPLUGIN_IRQ	RW	0x0	Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
11	ENIRQDEVPLUGIN	RW	0x0	Enables interrupt for the detection of connectivity to the USB line.

Table 329. HW_USBPHY_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
10	RESUME_IRQ	RW	0x0	Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
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	RSVD	RO	0x0	Reserved.
7	ENOTGIDDETECT	RW	0x0	Enables circuit to detect resistance of MiniAB ID pin.
6	RSVD	RO	0x0	Reserved.
5	DEVPLUGIN_POLARITY	RW	0x0	For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.
4	ENDEVPLUGINDETECT	RW	0x0	For device mode, enables 200-KOhm pullups for detecting connectivity to the host.
3	HOSTDISCONDETECT_IRQ	RW	0x0	Indicates that the device has disconnected in high-speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
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 that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet.
0	ENHSPRECHARGEEXMIT	RW	0x1	This field is not currently used in the STMP3770.

10.4.5. USB PHY Status Register Description

The USB PHY Status Register holds results of IRQ and other detects.

HW_USBPHY_STATUS 0x8007C040

HW_USBPHY_DEBUG1_TOG 0x8007C07C

Table 336. HW_USBPHY_DEBUG1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																ENTAILADJVD	ENTX2TX	RSVD				PLL_IS_240	RSVD				DBG_ADDRESS				

Table 337. HW_USBPHY_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:15	RSVD	RO	0x0	Reserved
14:13	ENTAILADJVD	RW	0x0	Delay increment of the rise of squelch: 00 = Delay is nominal 01 = Delay is +20% 10 = Delay is -20% 11 = Delay is -40%
12	ENTX2TX	RW	0x1	This bit has no function in the STMP3770.
11:9	RSVD	RO	0x0	Reserved.
8	PLL_IS_240	RW	0x0	Reserved. Always clear to 0
7:4	RSVD	RO	0x0	Reserved.
3:0	DBG_ADDRESS	RW	0x0	Chooses the multiplexing of the debug register to be shown in HW_USBPHY_DEBUG0_STATUS.

10.4.9. UTMI RTL Version Description

HW_USBPHY_VERSION 0x8007C080

Table 338. HW_USBPHY_VERSION

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
MAJOR								MINOR								STEP															

Table 339. HW_USBPHY_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x0	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x0	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0	Fixed read-only value reflecting the stepping of the RTL version.

USBPHY Block v3.0

STMP3770



11. AHB-TO-APBH BRIDGE WITH DMA

This chapter describes the AHB-to-APBH bridge on the STMP3770, along with its central DMA function and implementation examples. Programmable registers are described in [Section 11.5](#).

11.1. Overview

The AHB-to-APBH bridge provides the STMP3770 with an inexpensive peripheral attachment bus running on the AHB's HCLK. (The “H” in APBH denotes that the APBH is synchronous to HCLK, as compared to APBX, which runs on the crystal-derived XCLK.)

As shown in [Figure 30](#), 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.

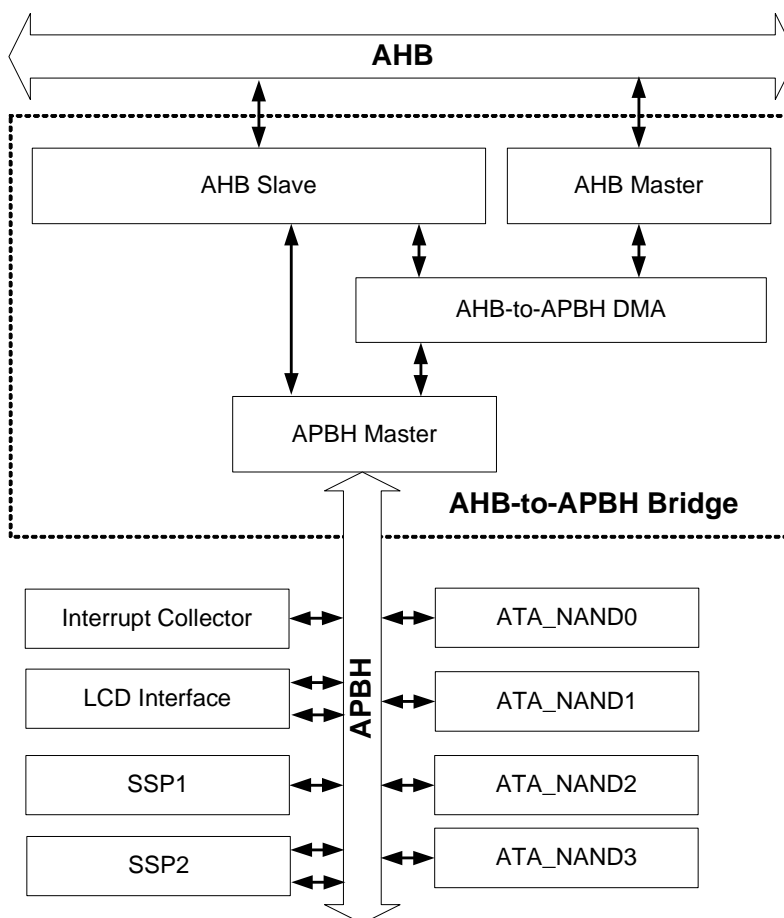


Figure 30. AHB-to-APBH Bridge DMA Block Diagram

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.

11.2. AHBH DMA

The DMA supports eight channels of DMA services, as shown in [Table 340](#). 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 31](#).

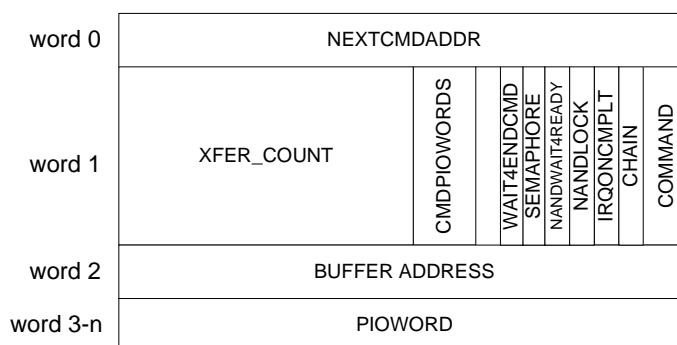
Table 340. APBH DMA Channel Assignments

APBH DMA CHANNEL #	USAGE
0	LCDIF
1	SSP1
2	SSP2
3	Reserved
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 STMP3770, 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 1 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.

The DMA master generates only normal write transfers to the APBH. It does *not* generate SCT set, clear, or toggle transfers.

**Figure 31. 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 341](#) shows the four commands implemented by the DMA.

Table 341. APBH DMA Commands

DMA COMMAND	USAGE
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the 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_ADDDDRESS 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

Figure 32 also shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1 if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) 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 1. 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 commands and set HW_GPMI_CTRL0_COMMAND_MODE_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 0 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_APBH_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_APBH_CHn_NXTCMDAR register, and then writes a 1 to the counting semaphore in HW_APBH_CHn_SEMA. The DMA channel loads HW_APBH_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_APBH_CHn_CURCMDAR at any time to determine the location of the command structure currently being processed.

11.3. Implementation Examples

11.3.1. NAND Read Status Polling Example

[Figure 32](#) 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.

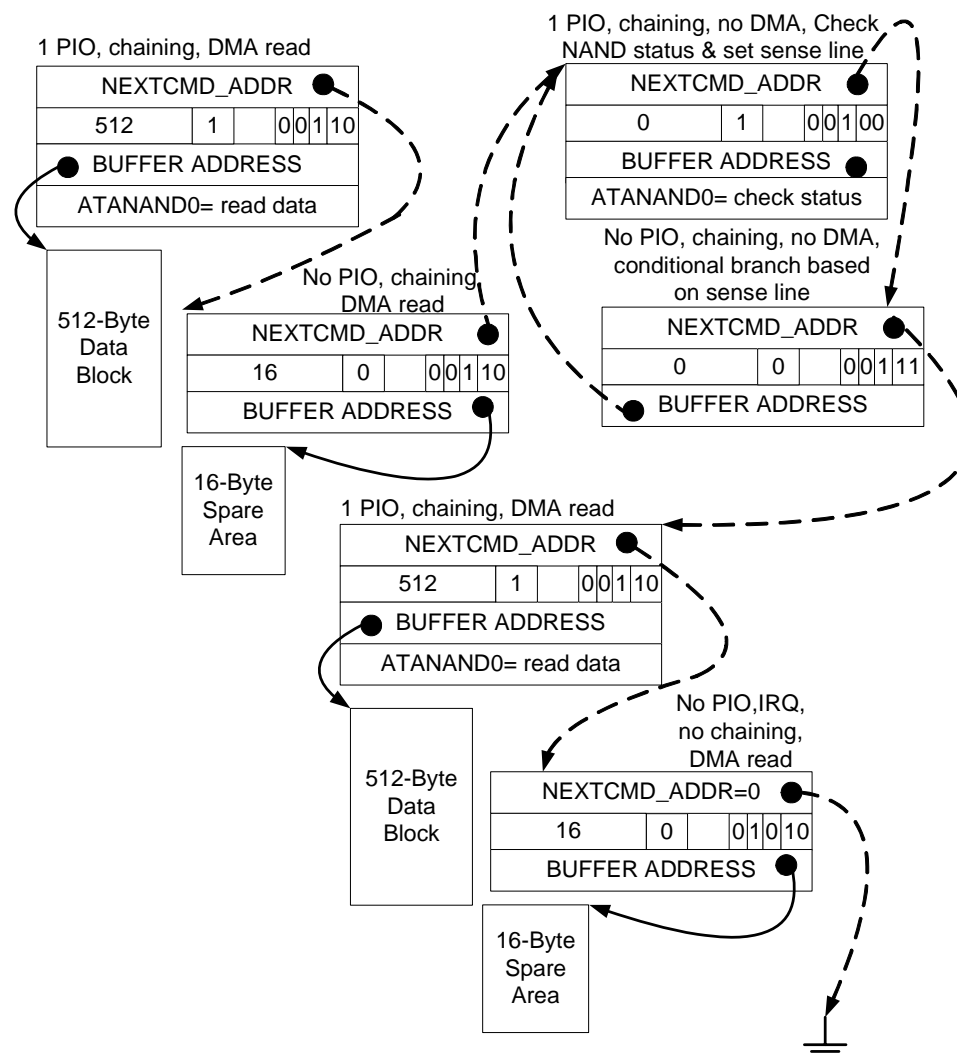


Figure 32. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command

The example in [Figure 32](#) 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.

SIGMATEL®
▶ Listen ● Watch ◀ Share

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 1 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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) 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-APBH bridge block.

11.5.1. 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 343. HW_APBH_CTRL0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
SFTRST	CLKGATE	RSVD							RESET_CHANNEL							CLKGATE_CHANNEL							FREEZE_CHANNEL								

Table 344. HW_APBH_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	Clear this bit to 0 to enable normal APBH DMA operation. Set this bit to 1 (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 cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block.

Table 344. HW_APBH_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29:24	RSVD	RO	0x000000	Reserved.
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. LCDIF = 0x01 SSP1 = 0x02 SSP2 = 0x04 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x40 NAND3 = 0x80
15:8	CLKGATE_CHANNEL	RW	0x00	These bits must be cleared to 0 for normal operation of each channel. When set to 1, they gate off the individual clocks to the channels. LCDIF = 0x01 SSP1 = 0x02 SSP2 = 0x04 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x40 NAND3 = 0x80
7:0	FREEZE_CHANNEL	RW	0x0	Setting a bit in this field freezes 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. LCDIF = 0x01 SSP1 = 0x02 SSP2 = 0x04 ATA = 0x10 NAND0 = 0x10 NAND1 = 0x20 NAND2 = 0x40 NAND3 = 0x80

11.5.2. 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 346. HW_APBH_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17	CH1_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB bus error during a transfer for APBH DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
16	CH0_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB bus error during a transfer for APBH DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
15	CH7_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 7.
14	CH6_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 6.
13	CH5_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 5.
12	CH4_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 4.
11	CH3_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 3.
10	CH2_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 2.
9	CH1_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 1.
8	CH0_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBH DMA Channel 0.
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.

STMP3770

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 bit
field write macro
HW_APBH_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure; // or, assign to bit
field of indexed register's struct
```

11.5.6. 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 0x80004060

Table 353. HW_APBH_CH0_CMD

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3			
XFER_COUNT															CMDWORDS				RSVD		HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND			

Table 354. 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 LCDIF device. 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 control register and incrementing from there. 0 means transfer NO command words
11:9	RSVD	RO	0x0	Reserved.
8	HALTONTERMINATE	RO	0x0	A value of 1 indicates that the channel will immediately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.

Table 354. HW_APBH_CH0_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5	NANDWAIT4READY	RO	0x0	A value of 1 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 1 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 1 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 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	This bit field indicates the type of current command: 00 = No DMA transfer 01 = Write transfers, i.e., data sent from the LCDIF (APB PIO Read) to the system memory (AHB master write). 10 = Read transfer 11 = DMA 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 peripheral 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

Table 359. HW_APBH_CH0_DEBUG1

[illegible]

Table 360. 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	SENSE	RO	0x0	This bit is reserved for this DMA channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI device.
26	READY	RO	0x0	This bit is reserved for this DMA channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI device.
25	LOCK	RO	0x0	This bit is reserved for this channel and always reads 0. For Channels 4-7, 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.
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 360. HW_APBH_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	RSVD	RO	0x0	Reserved
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 0 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p>

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 0.

STMP3770

Table 367. HW_APBH_CH1_CMD

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
XFER_COUNT																CMDWORDS					RSVD		HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND		

Table 368. 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 SSP1 device. 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 SSP1, starting with the base PIO address of the SSP1 control register and incrementing from there. 0 means transfer NO command words
11:9	RSVD	RO	0x0	Reserved.
8	HALTONTERMINATE	RO	0x0	A value of 1 indicates that the channel will immediately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5	NANDWAIT4READY	RO	0x0	A value of 1 indicates that the ATA/NAND DMA channel 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 1 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.

Table 368. HW_APBH_CH1_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	IRQONCMPLT	RO	0x0	A value of 1 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 1 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 bit field indicates the type of current command: 00 = No DMA transfer 01 = Write transfers, i.e., data sent from the SSP1 (APB PIO Read) to the system memory (AHB master write). 10 = Read transfer 11 = DMA 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 peripheral 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.

11.5.14. 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 0x800040E0

Table 369. HW_APBH_CH1_BAR

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDRESS																															

STMP3770

Table 374. HW_APBH_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25	LOCK	RO	0x0	This bit is reserved for this channel and always reads 0. For Channels 4-7, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
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.
19:5	RSVD	RO	0x0	Reserved
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 1 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p>

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 1.

11.5.17. AHB-to-APBH DMA Channel 1 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

```
HW APBH CH1 DEBUG2      0x80004110
```

Table 375. HW_APBH_CH1_DEBUG2

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
APB_BYTES																AHB_BYTES															

Table 376. 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.

11.5.18. 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 0x80004120

Table 377. HW APBH CH2 CURCMDAR

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
CMD_ADDR																															

Table 378. 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.

Table 382. 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 SSP2 device. 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 SSP2, starting with the base PIO address of the SSP2 control register and incrementing from there. 0 means transfer NO command words
11:9	RSVD	RO	0x0	Reserved.
8	HALTONTERMINATE	RO	0x0	A value of 1 indicates that the channel will immediately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5	NANDWAIT4READY	RO	0x0	A value of 1 indicates that the ATA/NAND DMA channel 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 1 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 1 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 394. 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.

11.5.27. APBH DMA Channel 3 Command Register Description

This register is not supported on the STMP3770. Channel 3 is reserved.

HW_APBH_CH3_CMD 0x800041B0

Table 395. HW_APBH_CH3_CMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
XFER_COUNT																CMDWORDS				RSVD		HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND		

Table 396. HW_APBH_CH3_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	XFER_COUNT	RO	0x0	Reserved.
15:12	CMDWORDS	RO	0x00	Reserved.
11:9	RSVD	RO	0x0	Reserved.
8	HALTONTERMINATE	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	Reserved.
6	SEMAPHORE	RO	0x0	Reserved.
5	NANDWAIT4READY	RO	0x0	Reserved.
4	NANDLOCK	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	Reserved.
2	CHAIN	RO	0x0	Reserved.
1:0	COMMAND	RO	0x00	NO_DMA_XFER = 0x0 Reserved. DMA_WRITE = 0x1 Reserved. DMA_READ = 0x2 Reserved. DMA_SENSE = 0x3 Reserved.

Table 402. HW_APBH_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
27	SENSE	RO	0x0	This bit is reserved for this DMA channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI device.
26	READY	RO	0x0	This bit is reserved for this DMA channel and always reads 0. For Channels 4-7, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI device.
25	LOCK	RO	0x0	This bit is reserved for this channel and always reads 0. For Channels 4-7, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
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	0x0	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	0x0	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 408. 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.

11.5.34. 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 0x80004220

Table 409. HW_APBH_CH4_CMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
XFER_COUNT																CMDWORDS				RSVD		HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND		

Table 410. 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 increment from there. 0 means transfer NO command words
11:9	RSVD	RO	0x0	Reserved.
8	HALTONTERMINATE	RO	0x0	A value of 1 indicates that the channel will immediately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.

Table 410. HW_APBH_CH4_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5	NANDWAIT4READY	RO	0x0	A value of 1 indicates that the ATA/NAND DMA channel 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 1 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 1 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 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	This bit field indicates the type of current command: 00 = No DMA transfer 01 = Write transfers, i.e., data sent from the GPPI (APB PIO Read) to the system memory (AHB master write). 10 = Read transfer 11 = DMA 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 peripheral 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.

11.5.35. APBH DMA Channel 4 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_CH4_BAR 0x80004230

Table 411. HW_APBH_CH4_BAR

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	0	0	0
ADDRESS																																		

Table 412. 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.

11.5.36. 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 0x80004240

Table 413. HW_APBH_CH4_SEMA

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD								PHORE								RSVD								INCREMENT_SEMA							

Table 416. HW_APBH_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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 GPPI Sense Signal sent from the APB GPPI device.
26	READY	RO	0x0	This bit reflects the current state of the GPPI Ready Signal sent from the APB GPPI device.
25	LOCK	RO	0x0	This bit reflects the current state of the DMA Channel Lock for a GPPI 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.
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 416. HW_APBH_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	RSVD	RO	0x0	Reserved
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 4 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p> <p>WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPPI device indicates that the external device is ready.</p>

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 4.

11.5.38. AHB-to-APBH DMA Channel 4 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

HW_APBH_CH4_DEBUG2 0x80004260

Table 424. HW_APBH_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	HALTONTERMINATE	RO	0x0	A value of 1 indicates that the channel will immediately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5	NANDWAIT4READY	RO	0x0	A value of 1 indicates that the ATA/NAND DMA channel 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 1 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 1 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 430. HW_APBH_CH5_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:5	RSVD	RO	0x0	Reserved
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 5 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p> <p>WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPPI device indicates that the external device is ready.</p>

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 5.

11.5.45. AHB-to-APBH DMA Channel 5 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

HW_APBH_CH5_DEBUG2 0x800042D0

Table 438. HW_APBH_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	HALTONTERMINATE	RO	0x0	A value of 1 indicates that the channel will immediately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5	NANDWAIT4READY	RO	0x0	A value of 1 indicates that the ATA/NAND DMA channel 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 1 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 1 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 438. HW_APBH_CH6_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	CHAIN	RO	0x0	A value of 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	<p>This bit field indicates the type of current command:</p> <p>00 = No DMA transfer</p> <p>01 = Write transfers, i.e., data sent from the GPMI (APB PIO Read) to the system memory (AHB master write).</p> <p>10 = Read transfer</p> <p>11 = DMA sense</p> <p>NO_DMA_XFER = 0x0 Perform any requested PIO word transfers but terminate command before any DMA transfer.</p> <p>DMA_WRITE = 0x1 Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes.</p> <p>DMA_READ = 0x2 Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.</p> <p>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 peripheral sense is true. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is false.</p>

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.

11.5.49. 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      0x80004310
```

Table 439. HW APBH CH6 BAR

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDRESS																															

Table 440. 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.

Table 444. HW_APBH_CH6_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 6 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p> <p>WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPML device indicates that the external device is ready.</p>

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 6.

11.5.52. AHB-to-APBH DMA Channel 6 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

HW_APBH_CH6_DEBUG2 0x80004340

Table 452. HW_APBH_CH7_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	HALTONTERMINATE	RO	0x0	A value of 1 indicates that the channel will immediately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5	NANDWAIT4READY	RO	0x0	A value of 1 indicates that the ATA/NAND DMA channel 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 1 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 1 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 458. HW_APBH_CH7_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 7 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p> <p>WAIT_READY = 0x1F When the NAND Wait for Ready bit is set, the state machine enters this state until the GPML device indicates that the external device is ready.</p>

DESCRIPTION:

This register allows debug visibility of the APBH DMA Channel 7.

11.5.59. AHB-to-APBH DMA Channel 7 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 7.

HW_APBH_CH7_DEBUG2 0x800043B0

Table 459. HW_APBH_CH7_DEBUG2

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
APB_BYTES																AHB_BYTES															

Table 460. 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.

11.5.60. APBH Bridge Version Register Description

This register indicates the version of the block for debug purposes.

HW_APBH_VERSION 0x800043F0

Table 461. HW_APBH_VERSION

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
MAJOR								MINOR								STEP															

Table 462. HW_APBH_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x01	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

APBH Block v1.1

12. AHB-TO-APBX BRIDGE WITH DMA

This chapter describes the AHB-to-APBX bridge on the STMP3770, along with its central DMA function and implementation examples. Programmable registers are described in [Section 12.5](#).

12.1. Overview

The AHB-to-APBX bridge provides the STMP3770 with an inexpensive peripheral attachment bus running on the AHB's XCLK. (The "X" in APBX denotes that the APBX runs on a crystal-derived clock, as compared to APBH, which is synchronous to HCLK.)

As shown in [Figure 33](#), 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.

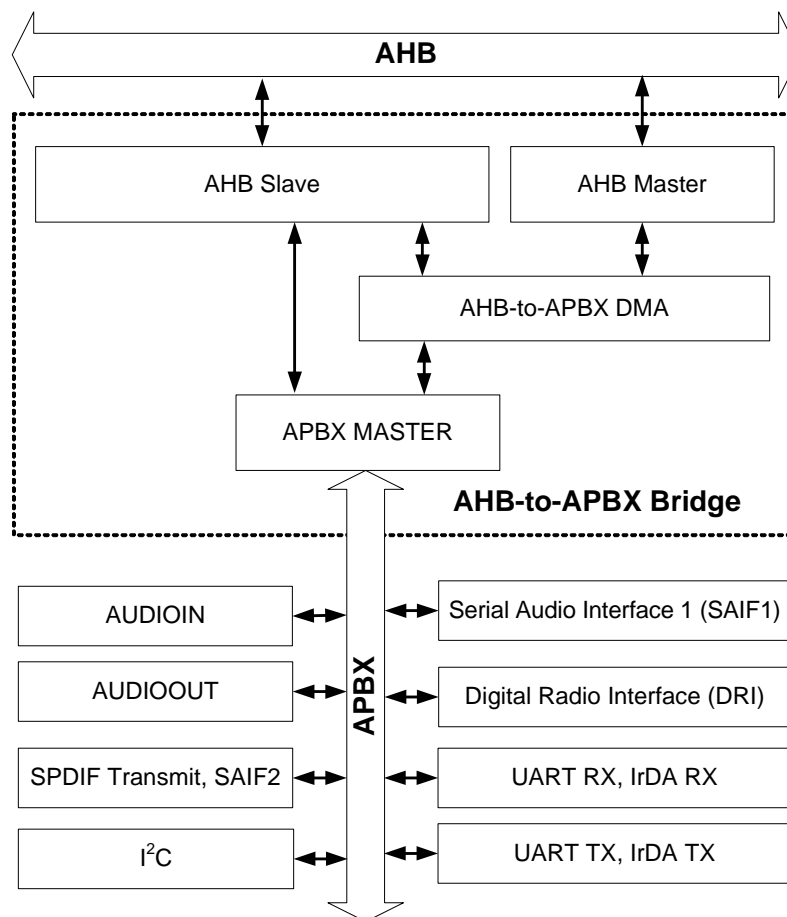


Figure 33. AHB-to-APBX Bridge DMA Block Diagram

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.

12.2. APBX DMA

The DMA supports eight channels of DMA services, as shown in [Table 463](#). 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 34](#).

Table 463. APBX DMA Channel Assignments

APBX DMA CHANNEL #	USAGE
0	Audio ADCs
1	Audio DACs
2	SPDIF TX, SAIF2
3	I ² C
4	SAIF1
5	Digital Radio Interface (DRI)
6	UART RX,
7	UART 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 STMP3770 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 1 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 serial audio interface to send 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 #PIO-WORDS 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 RX channel, the first PIO word in the DMA command is CTRL0. However, for APBX DMA Channel 7, which is the UART TX, the first PIO word in a DMA command is CTRL1.)

The HW_APBX_DEVSEL_CHx bit fields allow reassignment of the APBX device connected to DMA channels 2, 6, and 7. The DMA channel can be programmed to enable an alternate channel owner—for example, SAIF2 instead of SPDIF for Chan-

nel 2. Note that the CHx bit fields can be set only once after the chip is reset. To have the DMA channel provide DMA for another device, the chip must be reset and the HW_APBX_DEVSEL register must be reprogrammed. Whichever device is selected for the DMA channel remains the selected device until the chip is reset and the DMA channel selected for another device.

The DMA master generates only normal write transfers to the APBX. It does *not* generate set, clear, or toggle SCT transfers.

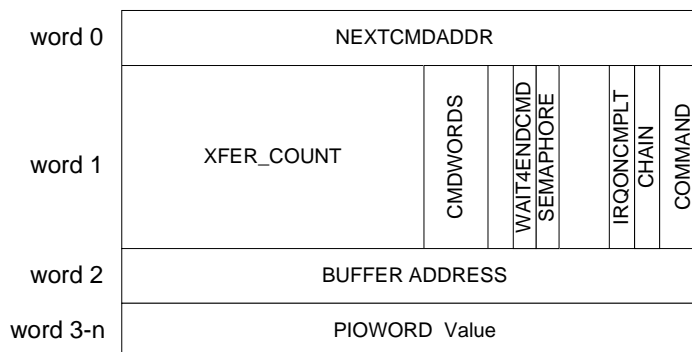


Figure 34. AHB-to-APBX Bridge DMA Channel Command Structure

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 464](#) shows the four commands implemented by the DMA.

Table 464. APBX DMA Commands

DMA COMMAND	USAGE
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the 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_WRITE transfer uses the BUFFER_ADDDDRESS 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

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 1 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_APBX_CHn_CURCMDAR at any time to determine the location of the command structure that is currently being processed.

12.3. DMA Chain Example

The example in [Figure 35](#) 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.

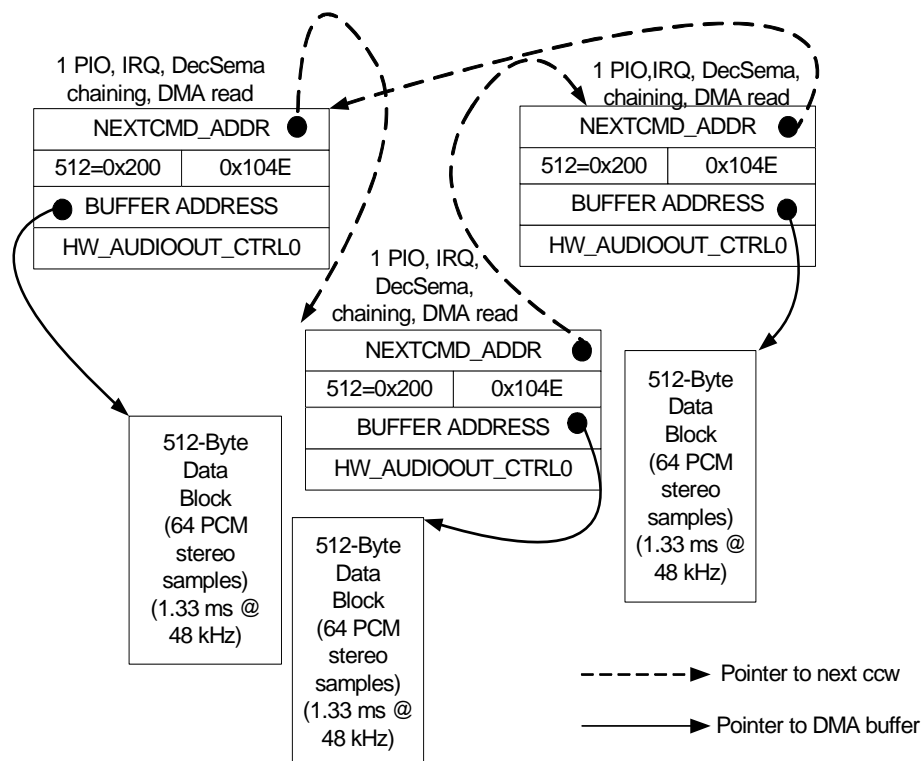


Figure 35. AHB-to-APBX Bridge DMA AUDIOOUT (DAC) Example Command Chain

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 1 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.

12.4. 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 34.4.10, "Correct Way to Soft Reset a Block" on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

Table 469. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	CH6_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB Bus Error during a transfer for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
21	CH5_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB Bus Error during a transfer for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
20	CH4_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB Bus Error during a transfer for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
19	CH3_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB Bus Error during a transfer for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
18	CH2_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB Bus Error during a transfer for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
17	CH1_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB Bus Error during a transfer for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
16	CH0_AHB_ERROR_IRQ	RW	0x0	Interrupt request status bit indicating an AHB Bus Error during a transfer for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. All 8 channel error bits are ORed together to generate the interrupt to the ARM core.
15	CH7_CMDCPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 7.
14	CH6_CMDCPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 6.
13	CH5_CMDCPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 5.
12	CH4_CMDCPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 4.
11	CH3_CMDCPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 3.
10	CH2_CMDCPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 2.
9	CH1_CMDCPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 1.

Table 469. HW_APBX_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	CH0_CMDCMPLT_IRQ_EN	RW	0x0	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 0.
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.
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 bit field write macro
BF_APBX_CTRL1.CH5_CMDCMPLT_IRQ = 0; // or, assign to register struct's bit field
```

12.5.3. AHB-to-APBX DMA Device Assignment Register Description

The AHB-to-APBX DMA Device Assignment Register allows reassignment of the APBX device connected to the DMA channels.

HW_APBX_DEVSEL 0x80024020

Table 476. HW_APBX_CH0_CMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
XFER_COUNT												CMDWORDS				RSVD				WAIT4ENDCMD	SEMAPHORE	RSVD		IRQONCMPLT	CHAIN	COMMAND					

Table 477. 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 increment from there. 0 means transfer NO command words.
11:8	RSVD	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5:4	RSVD	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	A value of 1 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 1 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_NXTCMDAR to find the next command.

Table 483. HW_APBX_CH0_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 0 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p>

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 0.

12.5.10. AHB-to-APBX DMA Channel 0 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 0.

HW_APBX_CH0_DEBUG2 0x800240A0

STMP3770

Table 491. 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 increment from there. 0 means transfer NO command words.
11:8	RSVD	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5:4	RSVD	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	A value of 1 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 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	This bit field 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 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.

Table 497. HW_APBX_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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	RSVD	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.
19:5	RSVD	RO	0x0	Reserved

Table 497. HW_APBX_CH1_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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 APBX DMA Channel 1.

12.5.17. AHB-to-APBX DMA Channel 1 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

HW_APBX_CH1_DEBUG2 0x80024110

0. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

12.5.23. AHB-to-APBX DMA Channel 2 Debug Information Register Description

This register gives debug visibility into the APBX DMA Channel 2 state machine and controls.

```
HW_APBX_CH2_DEBUG1      0x80024170
```

Table 510. HW_APBX_CH2_DEBUG1

REQ	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0		
BURST																																		
KICK																																		
END																																		
RSVD																																		
NEXTCMDADDRVALID																																		
RD_FIFO_EMPTY																																		
RD_FIFO_FULL																																		
WR_FIFO_EMPTY																																		
WR_FIFO_FULL																																		
RSVD																																		
STATEMACHINE																																		

Table 511. 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	RSVD	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.
19:5	RSVD	RO	0x0	Reserved

Table 511. HW_APBX_CH2_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 2 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p>

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 2.

12.5.24. AHB-to-APBX DMA Channel 2 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 2.

HW_APBX_CH2_DEBUG2 0x80024180

Table 512. HW_APBX_CH2_DEBUG2

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
APB_BYTES																AHB_BYTES															

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.

12.5.27. 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

0x800241B0

Table 518. HW_APBX_CH3_CMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
XFER_COUNT												CMDWORDS				RSVD				WAIT4ENDCMD	SEMAPHORE	RSVD		IRQONCMPLT	CHAIN	COMMAND					

Table 519. 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 increment from there. 0 means transfer NO command words.
11:8	RSVD	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5:4	RSVD	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	A value of 1 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.

Table 519. HW APBX CH3 CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	CHAIN	RO	0x0	A value of 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	<p>This bit field 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.</p> <p>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.</p>

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.

12.5.28. 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 0x800241C0

Table 520. HW APBX CH3 BAR

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDRESS																															

Table 521. 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

Table 524. HW_APBX_CH3_DEBUG1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0					
REQ	BURST	KICK	END	RSVD				NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD																STATEMACHINE							

Table 525. 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	RSVD	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.
19:5	RSVD	RO	0x0	Reserved

Table 525. HW_APBX_CH3_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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 APBX DMA Channel 3.

12.5.31. AHB-to-APBX DMA Channel 3 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 3.

HW_APBX_CH3_DEBUG2 0x800241F0

Table 526. HW_APBX_CH3_DEBUG2

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
APB_BYTES																AHB_BYTES															

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.

12.5.34. 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

0x80024220

Table 532. HW_APBX_CH4_CMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
XFER_COUNT												CMDWORDS				RSVD				WAIT4ENDCMD	SEMAPHORE	RSVD				IRQONCMPLT	CHAIN	COMMAND			

Table 533. 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 SAIF1 device. 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 SAIF1. 0 means transfer NO command words.
11:8	RSVD	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5:4	RSVD	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	A value of 1 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.

Table 533. HW APBX CH4 CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	CHAIN	RO	0x0	A value of 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	<p>This bit field 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.</p> <p>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.</p>

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.

12.5.35. 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 0x80024230

Table 534. HW APBX CH4 BAR

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDRESS																															

Table 535. 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

Table 539. HW_APBX_CH4_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4:0	STATEMACHINE	RO	0x0	<p>PIO Display of the DMA Channel 4 state machine state.</p> <p>IDLE = 0x00 This is the idle state of the DMA state machine.</p> <p>REQ_CMD1 = 0x01 State in which the DMA is waiting to receive the first word of a command.</p> <p>REQ_CMD3 = 0x02 State in which the DMA is waiting to receive the third word of a command.</p> <p>REQ_CMD2 = 0x03 State in which the DMA is waiting to receive the second word of a command.</p> <p>XFER_DECODE = 0x04 The state machine processes the descriptor command field in this state and branches accordingly.</p> <p>REQ_WAIT = 0x05 The state machine waits in this state for the PIO APB cycles to complete.</p> <p>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.</p> <p>PIO_REQ = 0x07 This state determines whether another PIO cycle needs to occur before starting DMA transfers.</p> <p>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.</p> <p>READ_WAIT = 0x09 When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete.</p> <p>WRITE = 0x0C During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.</p> <p>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.</p> <p>CHECK_CHAIN = 0x0E Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.</p> <p>XFER_COMPLETE = 0x0F The state machine goes to this state after the DMA transfers are complete, and determines what step to take next.</p> <p>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.</p> <p>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.</p> <p>CHECK_WAIT = 0x1E If the Chain bit is a 0, the state machine enters this state and effectively halts.</p>

DESCRIPTION:

This register allows debug visibility of the APBX DMA Channel 4.

12.5.38. AHB-to-APBX DMA Channel 4 Debug Information Register Description

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

HW_APBX_CH4_DEBUG2 0x80024260

Table 540. HW_APBX_CH4_DEBUG2

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
APB_BYTES																AHB_BYTES															

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.

12.5.41. 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

0x80024290

Table 546. HW_APBX_CH5_CMD

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4				
XFER_COUNT																CMDWORDS				RSVD				WAIT4ENDCMD	SEMAPHORE	RSVD		IRQONCMPLT	CHAIN	COMMAND	

Table 547. 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 increment from there. 0 means transfer NO command words.
11:8	RSVD	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5:4	RSVD	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	A value of 1 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.

Table 547. HW_APBX_CH5_CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	CHAIN	RO	0x0	A value of 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	<p>This bit field 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.</p> <p>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.</p>

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.

12.5.42. 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 0x800242A0

Table 548. HW APBX CH5 BAR

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDRESS																															

Table 549. 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.

12.5.43. 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 0x800242B0

Table 550. HW_APBX_CH5_SEMA

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD								PHORE								RSVD								INCREMENT_SEMA							

Table 551. HW_APBX_CH5_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:16	PHORE	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	RSVD	RO	0x0	Reserved.
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, then the count is incremented by a net 1.

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 which has already reached a value of 0. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

12.5.44. AHB-to-APBX DMA Channel 5 Debug Information Register Description

This register gives debug visibility into the APBX DMA Channel 5 state machine and controls.

HW_APBX_CH5_DEBUG1 0x800242C0

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.

12.5.48. 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

0x80024300

Table 560. HW_APBX_CH6_CMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
XFER_COUNT												CMDWORDS				RSVD				WAIT4ENDCMD	SEMAPHORE	RSVD		IRQONCMPLT	CHAIN	COMMAND					

Table 561. 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 increment from there. 0 means transfer NO command words.
11:8	RSVD	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5:4	RSVD	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	A value of 1 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.

Table 561. HW APBX CH6 CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	CHAIN	RO	0x0	A value of 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	<p>This bit field 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.</p> <p>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.</p>

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.

12.5.49. 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 0x80024310

Table 562. HW APBX CH6 BAR

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDRESS																															

Table 563. 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.

12.5.50. 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 0x80024320

Table 564. HW_APBX_CH6_SEMA

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD								PHORE								RSVD								INCREMENT_SEMA							

Table 565. HW_APBX_CH6_SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:16	PHORE	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	RSVD	RO	0x0	Reserved.
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, then the count is incremented by a net 1.

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 which has already reached a value of 0. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

12.5.51. AHB-to-APBX DMA Channel 6 Debug Information Register Description

This register gives debug visibility into the APBX DMA Channel 6 state machine and controls.

HW_APBX_CH6_DEBUG1 0x80024330

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.

12.5.55. 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

0x80024370

Table 574. HW_APBX_CH7_CMD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
XFER_COUNT												CMDWORDS				RSVD				WAIT4ENDCMD	SEMAPHORE	RSVD		IRQONCMPLT	CHAIN	COMMAND					

Table 575. 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 increment from there. 0 means transfer NO command words.
11:8	RSVD	RO	0x0	Reserved.
7	WAIT4ENDCMD	RO	0x0	A value of 1 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 1 indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to 0, then this channel stalls until software increments it again.
5:4	RSVD	RO	0x0	Reserved.
3	IRQONCMPLT	RO	0x0	A value of 1 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.

Table 575. HW APBX CH7 CMD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	CHAIN	RO	0x0	A value of 1 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_NXTCMDAR to find the next command.
1:0	COMMAND	RO	0x00	<p>This bit field 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.</p> <p>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.</p>

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.

12.5.56. 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 0x80024380

Table 576. HW APBX CH7 BAR

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
ADDRESS																															

Table 577. 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

The APBX DMA Channel 7 Semaphore Register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Table 578. HW APBX CH7 SEMA

Table 579. HW APBX CH7 SEMA Bit Field Descriptions

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 which has already reached a value of 0. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

This register gives debug visibility into the APBX DMA Channel 7 state machine and controls.

HW APBX CH7 DEBUG1 0x800243A0

STMP3770



13. GENERAL-PURPOSE MEDIA INTERFACE (GPMI)

This chapter describes the general-purpose media interface (GPMI) on the STMP3770, including sections on both ATA and NAND modes. Programmable registers are described in [Section 13.4](#).

13.1. Overview

The general-purpose media interface (GPMI) controller is a flexible interface to up to four NAND flash.

- The NAND mode has configurable address and command behavior, providing support for future devices not yet specified.

The GPMI resides on the APBH. The GPMI also provides an interface to the ECC8 module to allow direct parity processing.

Registers are clocked on the HCLK domain. The I/O and pin timing are clocked on a dedicated GPMICK domain. GPMICK can be set to maximize I/O performance.

[Figure 36](#) shows a block diagram of the GPMI controller.

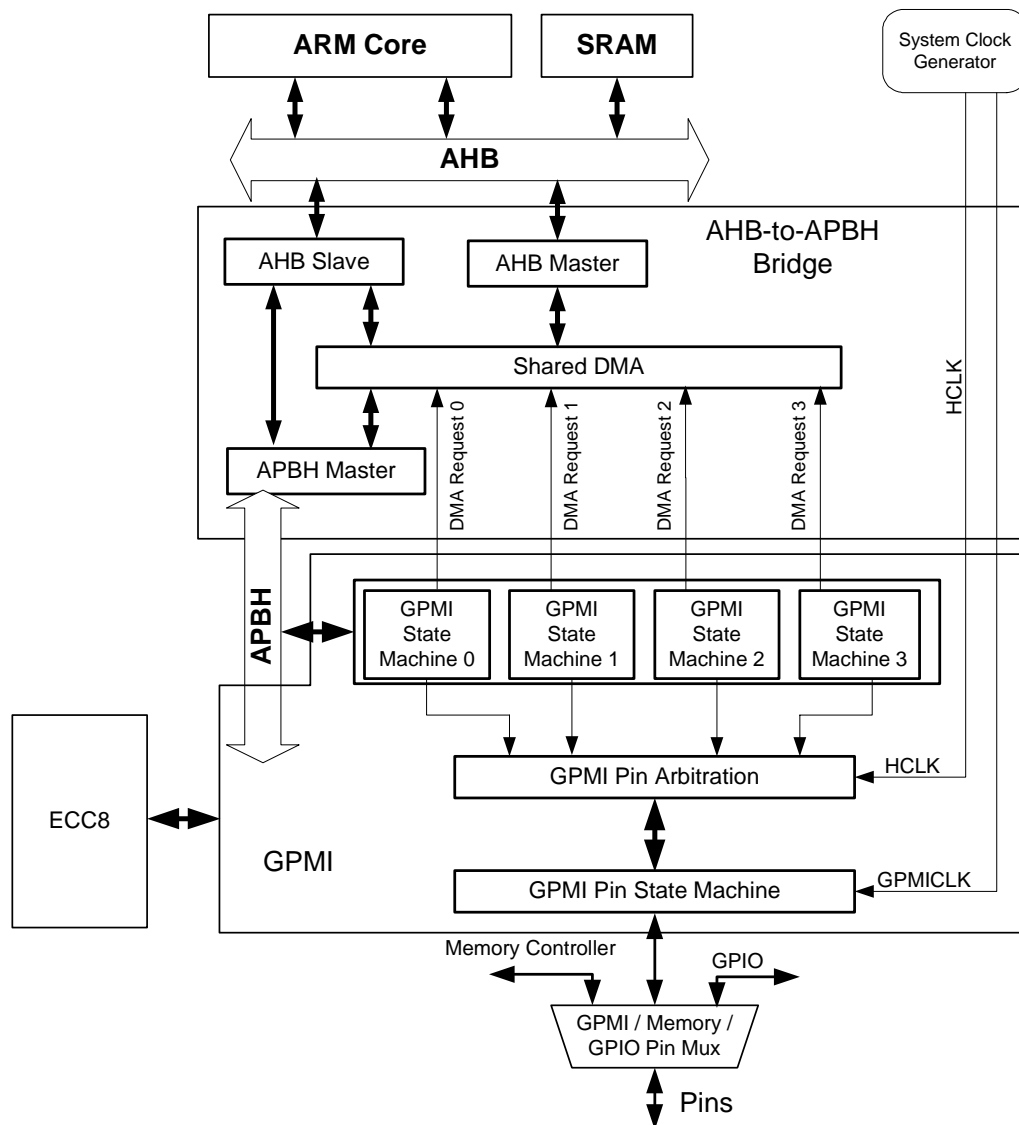


Figure 36. General-Purpose Media Interface Controller Block Diagram

13.2. 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 NANDs in the 100-pin packages).
- 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 time-out counter and can indicate to the DMA that a time-out 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.

13.2.1. 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).

13.2.2. 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 13.4](#). Refer to [Chapter 4](#) for more information about setting GPMICLK.

13.2.3. Basic NAND Timing

[Figure 37](#) illustrates the operation of the timing parameters in NAND mode.

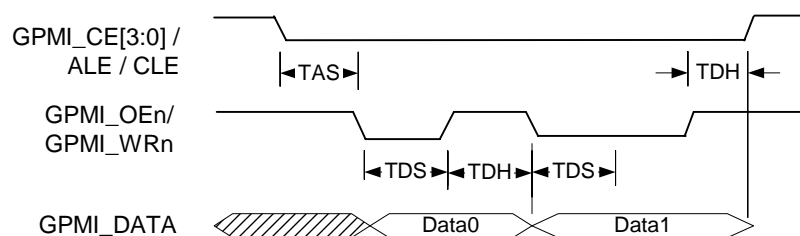
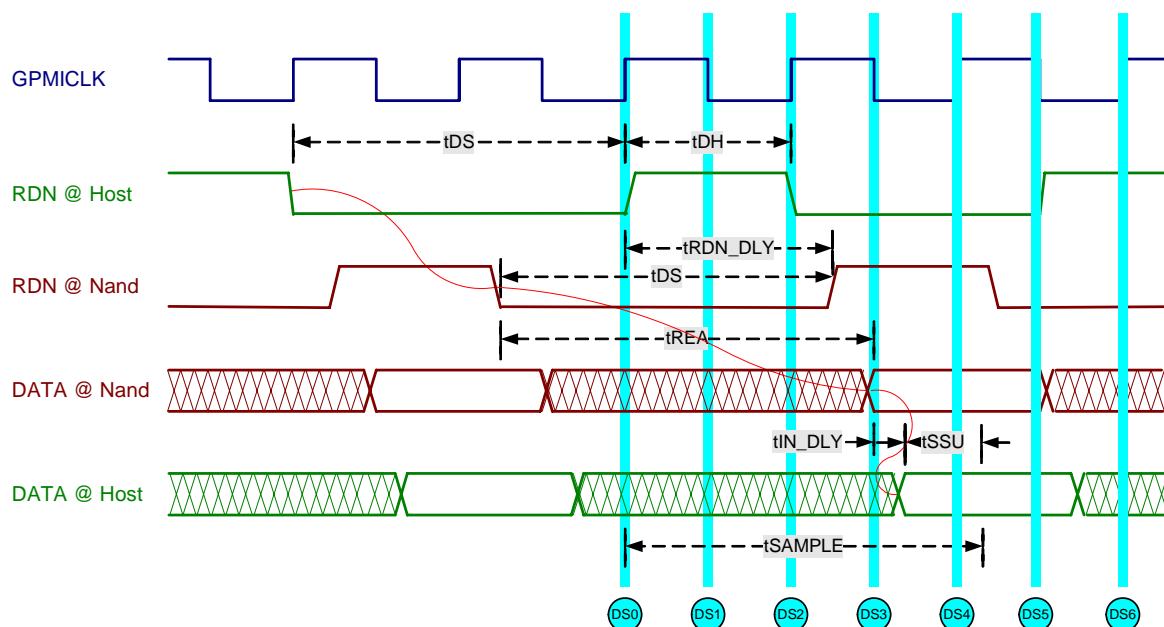


Figure 37. BASIC NAND Timing

13.2.4. High-Speed NAND Timing

In high-speed NANDS, the read data may not be valid until after the read strobe (RDN) deasserts. This is the case when the minimum tDS is programmed to achieve higher bandwidth. The DSAMPLE_TIME is a programmable field in HW_GPML_CTRL1 register that controls when read data from the NAND is sampled in the GPML module. This section describes how to program it. In [Figure 38](#), tSAMPLE represents the time from sample point DS0 (the rising edge of RDN@Host) to the middle of the window of valid data. So, by knowing tSAMPLE and the GPMICLK period, the correct sample point can be selected.

Note: *It is recommended that the drive strength of GPML_RDn and GPML_WRn output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND interface read and write cycle times are below 30 ns. The other GPML pins may remain at 4 mA, since their frequency is only up to half that of GPML_RDn and GPML_WRn.*



NOTES:

$t_{VALDATA}$ is the length of the valid read data window. This can be calculated from the following:

$$t_{VALDATA} = t_{DS} + t_{RHOH} - t_{REA} \quad (\text{Apply when } t_{RC} < 30 \text{ nsec})$$

$$t_{VALDATA} = t_{RC} + t_{RLOH} - t_{REA} \quad (\text{Apply when } t_{RC} > 30 \text{ nsec})$$

The read cycle time is $t_{RC} = t_{DS} + t_{DH}$.

$$t_{SAMPLEmax} = t_{RDN_DLYmin} + (t_{REA} - t_{DS}) + t_{IN_DLYmin} + t_{VALDATA}$$

$$t_{SAMPLEmin} = t_{RDN_DLYmax} + (t_{REA} - t_{DS}) + t_{IN_DLYmax}$$

$$t_{SAMPLE} = (t_{SAMPLEmin} + t_{SAMPLEmax}) / 2$$

$$DSAMPLE_TIME = t_{SAMPLE} / (GPMICKL_PERIOD / 2). \text{ Round to the nearest integer}$$

Note that $(t_{REA} - t_{DS})$ could be negative and therefore t_{SAMPLE} could also be negative. If so, then $DSAMPLE_TIME$ will be 0. Also, parameters must be chosen such that $t_{SAMPLEmax}$ is greater than $t_{SAMPLEmin}$. Also after rounding, ensure that the following condition is met:

$$t_{SAMPLEmin} < (DSAMPLE_TIME * GPMICKL_PERIOD/2) < t_{SAMPLEmax}$$

Figure 38. NAND Read Path Timing

13.2.5. NAND Command and Address Timing Example

Figure 39 illustrates a command and address being sent to a NAND flash.

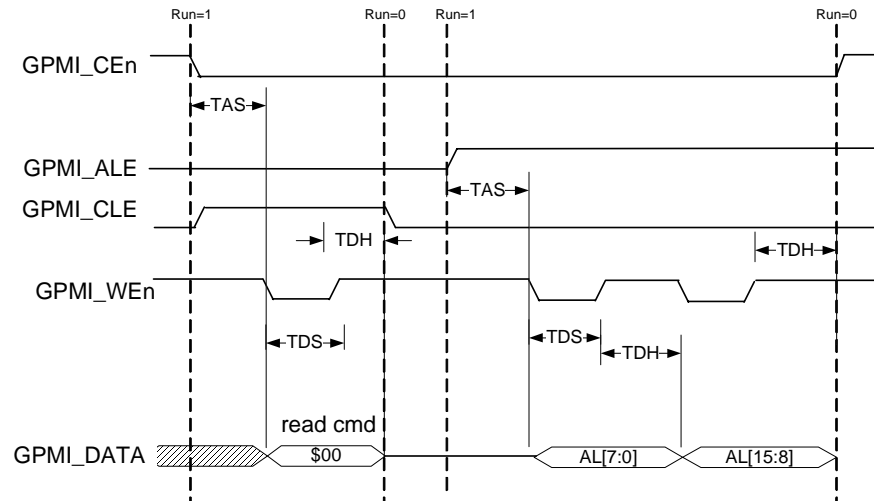


Figure 39. NAND Command and Address Timing Example

13.2.6. Hardware ECC (ECC8) Interface

The GPII provides an interface to the ECC8 module. This reduces the SOC bus traffic and the software involvement. When in ECC8 mode, parity information is inserted on-the-fly during writes to 8-bit NAND devices. During NAND reads, parity is checked and ECC processing is performed after each read block.

In ECC8 mode, during NAND writes, each 512-byte block of payload data is sent to the ECC8 module at the same time it is sent to the NAND. The ECC8 module returns the parity information, which is then appended to the block of data written to the NAND. This is repeated for each block of data written to the NAND. During NAND reads, each block of payload data and parity is redirected to the ECC8 module for ECC processing, instead of DMA to memory.

To program the ECC8 for NAND writes, remove the soft reset and clock gates from HW_ECC8_CTRL_SFTRST and HW_ECC8_CTRL_CLKGATE. The bulk ECC8 programming is actually applied to the GPII via PIO operations embedded in its DMA command structures. This has a subtle implication when writing to the GPII ECC8 registers: access to these registers must be written in progressive register order. Thus, to write to the HW_GPII_ECCCOUNT register, write first (in order) to registers HW_GPII_CTRL0, HW_GPII_COMPARE, and HW_GPII_ECCCTRL before writing to HW_GPII_ECCCOUNT. These additional register writes need to be accounted for in the CMDWORDS field of the respective DMA channel command register. See the descriptive text, flowcharts, and code examples in [Section 14.2.1](#), [Section 14.2.2](#), and [Section 14.2.3](#) for more information about using GPII registers to program the ECC8 function.

Note that the HW_GPII_PAYLOAD and HW_GPII_AUXILIARY pointers need to be word-aligned for proper ECC8 operation. If those pointers are non-word-aligned, then the ECC8 engine will not operate properly and could possibly corrupt system memory in the adjoining memory regions.

13.3. 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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

13.4. Programmable Registers

The following registers provide control for programmable elements of the GPMI module.

13.4.1. 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 586. HW GPMI CTRL0

SFSTRST	31	30	29	28	27	26	25	24	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																																
RUN																																
DEV_IRQ_EN																																
TIMEOUT_IRQ_EN																																
UDMA																																
COMMAND_MODE																																
WORD_LENGTH																																
LOCK_CS																																
CS																																
ADDRESS																																
ADDRESS_INCREMENT																																
XFER_COUNT																																

Table 587. HW_GPMI_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	Clear to 0 for normal operation. When this bit is set to 1 (default), then the entire block is held in its reset state. RUN = 0x0 Allow GPML to operate normally. RESET = 0x1 Hold GPML in reset.
30	CLKGATE	RW	0x1	Clear this bit to 0 for normal operation. Setting this bit to 1 (default), gates all of the block level clocks off for minimizing AC energy consumption. RUN = 0x0 Allow GPML to operate normally. NO_CLKS = 0x1 Do not clock GPML gates in order to minimize power consumption.
29	RUN	RW	0x0	The GPML is busy running a command whenever this bit is set to 1. The GPML is idle whenever this bit is cleared to 0. This can be set to 1 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 GPML is idle. BUSY = 0x1 The GPML is busy running a command.

Table 587. HW_GPML_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
28	DEV_IRQ_EN	RW	0x0	When set to 1 and ATA_IRQ pin is asserted, the GPML_IRQ output will assert.
27	TIMEOUT_IRQ_EN	RW	0x0	Setting this bit to 1 will enable time-out 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 time-out.
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.
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.

Table 599. HW_GPML_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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 time-out 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).
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	GPML_MODE	RW	0x0	0 = NAND mode. 1 = ATA mode. ATA mode is only supported on Channel 0. If ATA mode is selected, then only Channel 3 is available for NAND use. NAND = 0x0 NAND mode. ATA = 0x1 ATA mode.

13.4.8. GPML Timing Register 0 Description

The GPML 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_GPML_TIMING0

0x8000C070

Table 600. HW_GPMI_TIMING0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD								ADDRESS_SETUP								DATA_HOLD								DATA_SETUP							

Table 601. HW_GPMI_TIMING0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
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 0 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 de-asserted in a cycle. A value of 0 is interpreted as 256. 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 256. For ATA PIO modes this is known in the ATA7 specification as "DIOR-/DIOW-".

13.4.9. GPMI Timing Register 1 Description

The GPMI Timing Register 1 specifies the time-outs used when monitoring the NAND READY pin or the ATA IRQ and IOWAIT signals.

HW_GPMI_TIMING1

0x8000C080

Table 608. HW GPMI STAT

PRESENT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RSVD																				RDY_TIMEOUT				ATA_IRQ	INVALID_BUFFER_MASK	FIFO_EMPTY	FIFO_FULL	DEV3_ERROR	DEV2_ERROR	DEV1_ERROR	DEV0_ERROR

Table 609. HW GPMI_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	PRESENT	RO	0x1	0 = GPML is not present in this product. 1 = GPML is present is in this product. UNAVAILABLE = 0x0 GPML is not present in this product. AVAILABLE = 0x1 GPML is present in this product.
30:12	RSVD	RO	0x0	Reserved.
11:8	RDY_TIMEOUT	RO	0x0	Status of the RDY/BUSY time-out flags.
7	ATA_IRQ	RO	0x0	Status of the ATA_IRQ input pin.
6	INVALID_BUFFER_MASK	RO	0x0	0 = ECC Buffer Mask is not invalid. 1 = ECC Buffer Mask is invalid.
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 (Time out 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 (Time out 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 (Time out 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 (Time out or compare failure, depending on COMMAND_MODE).

13.4.13. 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

0x8000C0C0

Table 610. HW_GPMI_DEBUG

READY3	3 1
READY2	3 0
READY1	2 9
READY0	2 8
WAIT_FOR_READY_END3	2 7
WAIT_FOR_READY_END2	2 6
WAIT_FOR_READY_END1	2 5
WAIT_FOR_READY_END0	2 4
SENSE3	2 3
SENSE2	2 2
SENSE1	2 1
SENSE0	2 0
DMAREQ3	1 9
DMAREQ2	1 8
DMAREQ1	1 7
DMAREQ0	1 6
CMD_END	1 5
	1 4
	1 3
	1 2
UDMA_STATE	1 1
	1 0
	0 9
	0 8
BUSY	0 7
PIN_STATE	0 6
	0 5
	0 4
	0 3
MAIN_STATE	0 2
	0 1
	0 0
	0 0

Table 611. 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 time-out occurred.
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 time-out occurred.
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 time-out occurred.
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 time-out occurred.

STMP3770

Table 611. HW_GPMI_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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.
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	UDMA State. USM_IDLE = 0x0 Idle USM_DMARQ = 0x1 DMA REQ USM_ACK = 0x2 DMA ACK USM_FIFO_E = 0x3 FIFO empty USM_WPAUSE = 0x4 WR DMA Paused by device USM_TSTRB = 0x5 Toggle HSTROBE USM_CAPTUR = 0x6 Capture Stage (data sampled with DSTROBE is valid) USM_DATAOUT = 0x7 Change Burst DATAOUT USM_CRC = 0x8 Source CRC to Device USM_WAIT_R = 0x9 Waiting for DDMARDY- USM_END = 0xA; Negate DMAACK (end of DMA) USM_WAIT_S = 0xB Waiting for DSTROBE USM_RPAUSE = 0xC Rd DMA paused by host USM_RSTOP = 0xD Rd DMA stopped by host USM_WTERM = 0xE Wr DMA termination state USM_RTERM = 0xF Rd DMA termination state
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	Pin State Machine. 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	Main State Machine. 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

13.4.14. GPMI Version Register Description

This register reflects the version number for the GPMI.

HW_GPMI_VERSION

0x8000C0D0

STMP3770



14. 8-SYMBOL CORRECTING ECC ACCELERATOR (ECC8)

This chapter describes the DMA-based hardware ECC accelerator (ECC8) available on the STMP3770. 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 may be attached to the STMP3770. For example, modern high-density NAND flash devices presume the existence of forward error-correction algorithms to correct some soft and/or hard bit errors within the device, allowing for higher device yields and, therefore, lower device costs.

The hardware ECC8 accelerator uses the Reed-Solomon block codes, a subset of BCH codes, for multi-symbol error corrections. A *symbol* comprises multiple bits. The ECC8 operates on 9-bit symbols in its computations. A *symbol error* (and correction) means that any one or all bits of the symbol could be in error. Thus, under a best case scenario, a 4-symbol ECC protection can correct up to 36 bits ($= 4 * 9$) in error. Under the worst case scenario, only 4 bits ($= 4 * 1$) can be corrected.

In a Reed-Solomon ECC, the fixed *data payload* to be protected is mapped into data symbols to represent a unique polynomial. The polynomial is divided by a known generator polynomial (that is a function of the number of symbols to be corrected), where the residual remainder polynomial becomes the *parity symbols*. An *ECC codeword* is formed by concatenating the data symbols with the parity symbols. This ECC codeword is then written onto the storage media. All arithmetic operations in the Reed-Solomon ECC algorithm operate under Galois fields. The ECC8 supports $t=4$ symbol correction for 2K page NAND and $t=8$ symbol correction for 4K page NANDs.

Error correction occurs when the ECC codeword is read back from the storage media.

- A syndrome polynomial is generated in parallel as the GPMI reads the ECC codeword from the NAND.
- If the resulting syndrome polynomial is 0, then there are no errors.
- Otherwise, the ECC8 block invokes the correction unit to correct the codeword.
- The correction unit applies the Berlekamp-Massey algorithm on the syndrome polynomial to generate two unique polynomials, λ (λ) and Ω (Ω).
- In the next pipelined state, the correction unit applies the Chien search algorithm on the λ polynomial to identify the location of the corrupted codeword symbol and the Fourny algorithm on the Ω polynomial to compute a correction mask for the bad symbol.

The ECC8 block was designed to operate in a pipelined fashion to maximize throughput. Aside from the initial latency to fill the pipeline stages, the ECC8 throughput is faster than the fastest GPMI read rate of 2 cycles/byte. Thus, the bottleneck in performing NAND reads and error corrections is the GPMI read rate. Current GPMI read rates are approximately 3 cycles/byte for the current generation of NANDs. The ECC8 block has an AHB master that 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 generating parity symbols, checking for errors, or correcting them.

The hardware ECC8 accelerator is illustrated in [Figure 40](#).

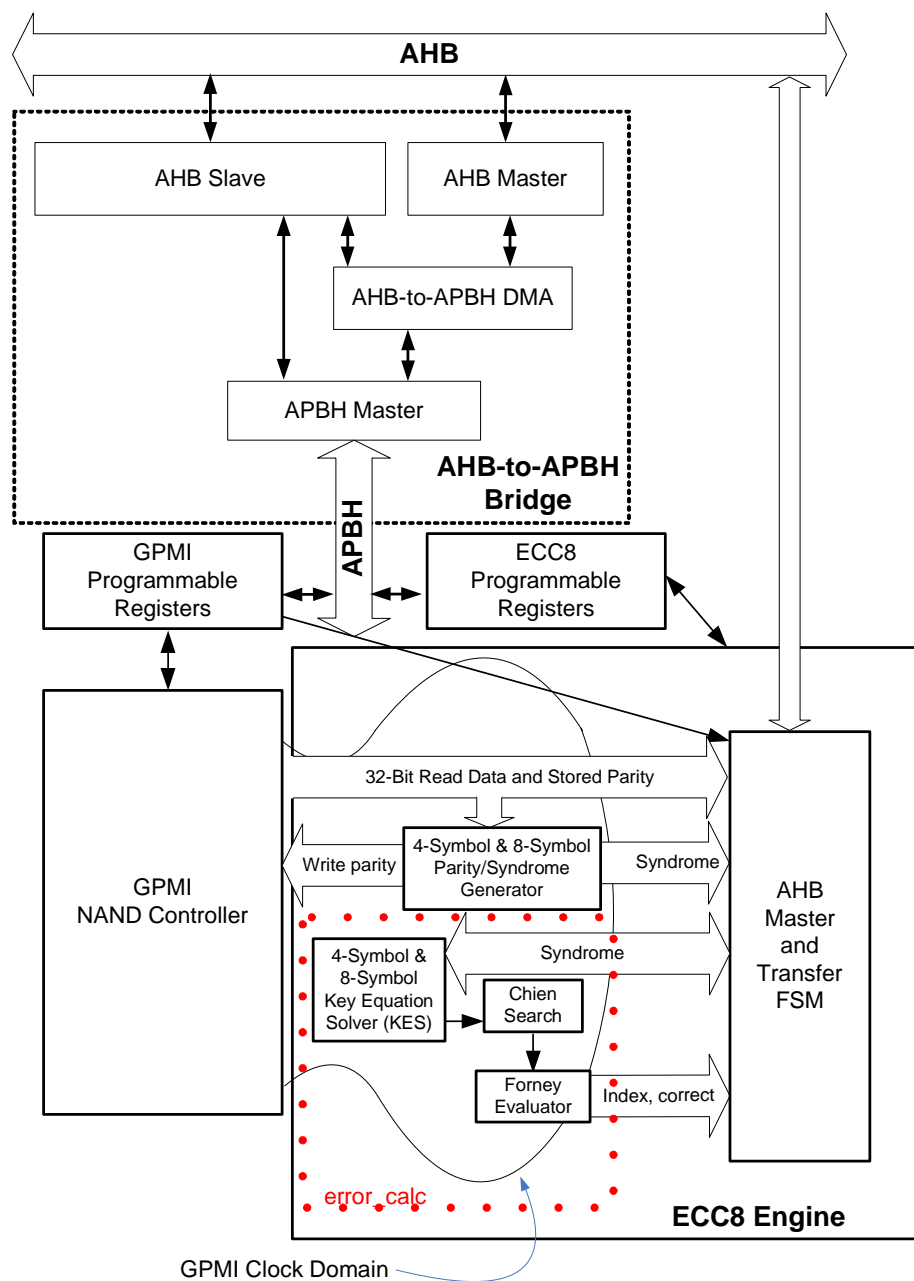


Figure 40. Hardware 8-Symbol Correcting ECC Accelerator (ECC8) Block Diagram

14.2. Operation

The data flow for NAND reads passes data directly from the GPMI controller to the ECC8 accelerator without first passing through system memory. This is a much higher bandwidth flow than the APBH DMA transfers used on the STMP36xx HWECC. In addition, the copying to and from system memory is eliminated.

Because the ECC8 operates on data flowing directly from the GPMI, it handles all error correction operations in an optimized pipelined manner. That is, blocks without errors complete within 20 HCLKs of the GPMI completing the read transfer. If errors are present, then the necessary pipeline stages are activated, including error calculation and error correction stages. Unlike the STMP36xx HWECC, the ECC8 engine directly performs all error corrections in the system memory buffer without CPU intervention, i.e., when the CPU gets an ECC8 interrupt, the error correction process is complete for all blocks of a transaction.

A read transaction for a 4K NAND page can consist of up to 9 block transfers, i.e., eight blocks of 512-byte payload and one block of 65-byte auxiliary data.

For a 2K NAND page, up to 5 block transfers can be specified for a single transaction, i.e., four blocks of 512-byte payload and one block of 19-byte auxiliary data.

For NAND write operations, the GPMI fetches the write data via its DMA interface as usual. However, it forks a copy of the write data to the ECC8 parity/syndrome generator. The ECC8 computes the parity bytes for the transfer on the fly. As soon as the GPMI writes the last data byte to the NAND, it switches its data flow so that the 9 or 18 bytes of Reed-Solomon ECC parity is copied from the ECC8 parity/syndrome generator directly to the NAND. In this case, no extra buffer in system memory is required. The ECC parity generation is fully overlapped with the data write transfer, so that the parity bytes are written immediately after the data is written, with only a few GPMI clocks of latency.

The ECC8 engine supports both an 8-symbol correcting mode and a 4-symbol correcting mode. The number of parity bytes required for each mode is different. For example, the 8-symbol correcting mode requires 16 parity symbols to be stored with the data. This corresponds to 18 bytes of parity information. Since a 2K page NAND device has only 64 bytes of spare, it cannot hold the required $4 \times 18 = 72$ bytes of parity data. Recall that a 2K page holds four 512-byte payload blocks. Thus, the 4-symbol correcting mode must be used for 2K page NAND devices. Fortunately, this is consistent with the bit error densities guaranteed for 2K page NAND devices.

[Figure 41](#) and [Figure 42](#) show the organization of the 4-symbol correcting mode 2K page NAND storage, both on the NAND and in the system memory footprint.

[Figure 43](#) and [Figure 44](#) show the NAND image and the system memory footprint used in the 8-symbol correcting mode available for 4K pages only. Notice that the auxiliary data is protected by the 4-symbol error correcting mode, regardless of whether it is stored on a 2K page NAND device or a 4K page NAND device.

Total NAND Memory Footprint: 2112 Bytes
 2048 Bytes Data + 64 Bytes Redundant Area = $4 * (512\text{B} + 9\text{B}) + (19 + 9\text{B})$

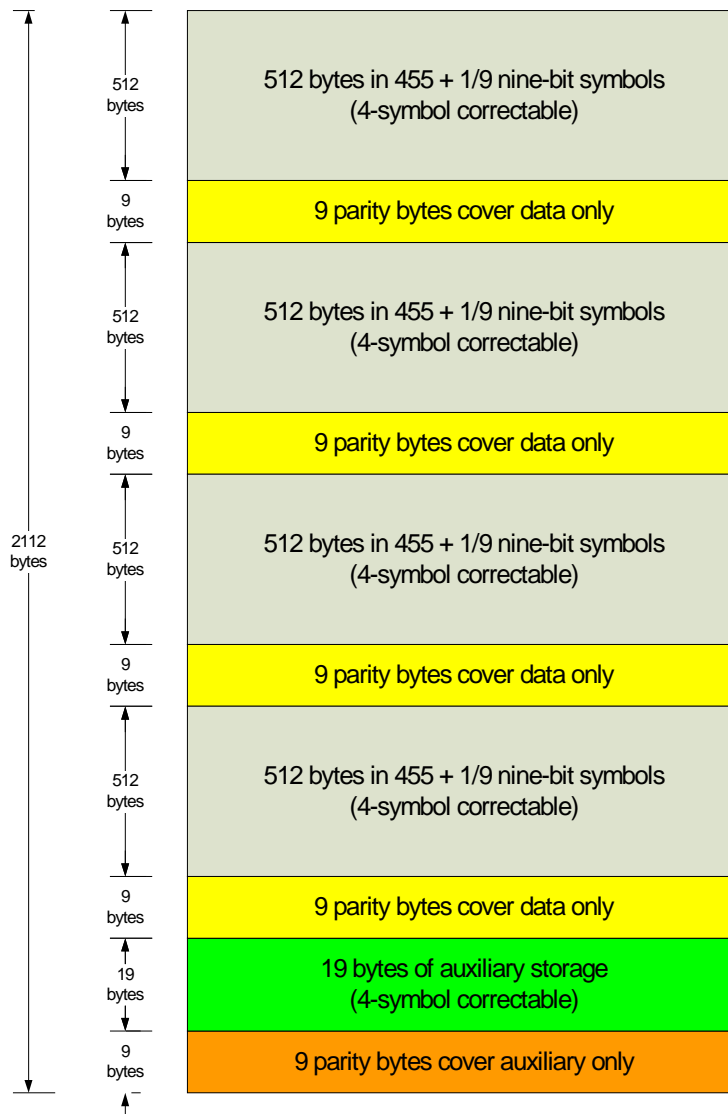


Figure 41. ECC-Protected 2K NAND Page Data—NAND Memory Footprint

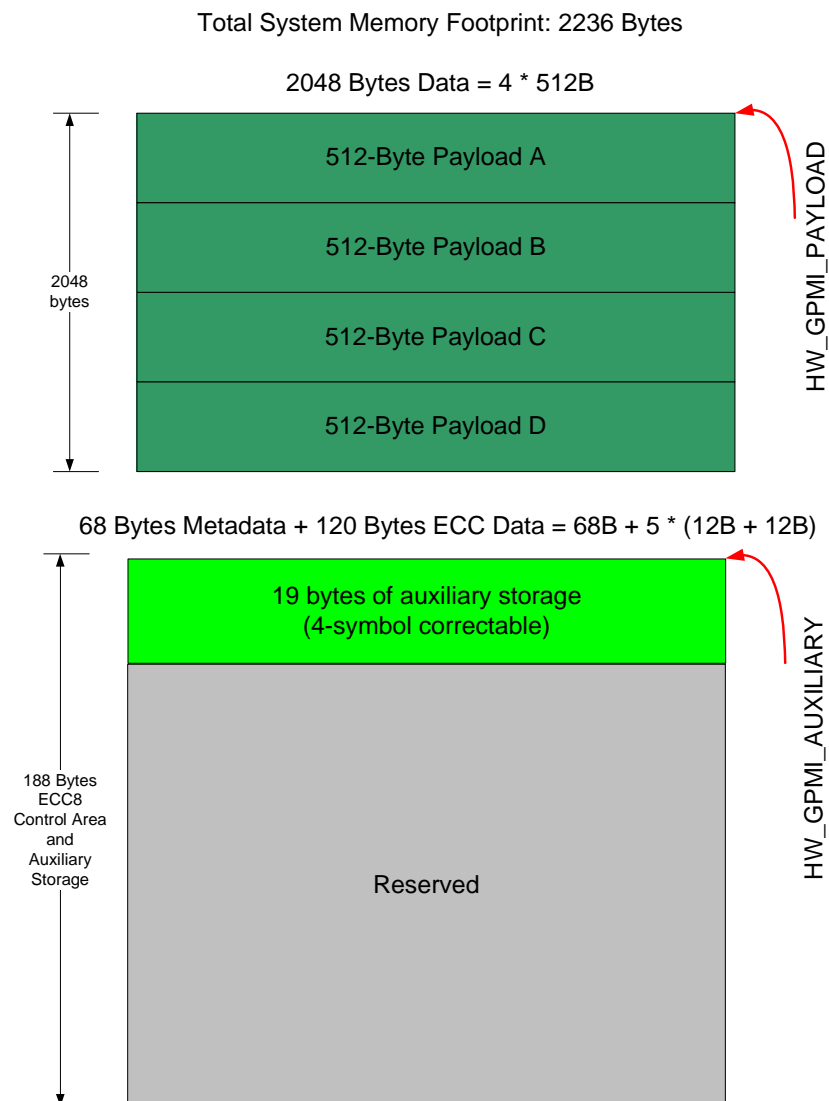


Figure 42. ECC-Protected 2K NAND Page Data—System Memory Footprint

Total NAND Memory Footprint: 4314 Bytes
 4096 Bytes Data + 218 Bytes Redundant Area = $8 * (512B + 18B) + (65B + 9B)$

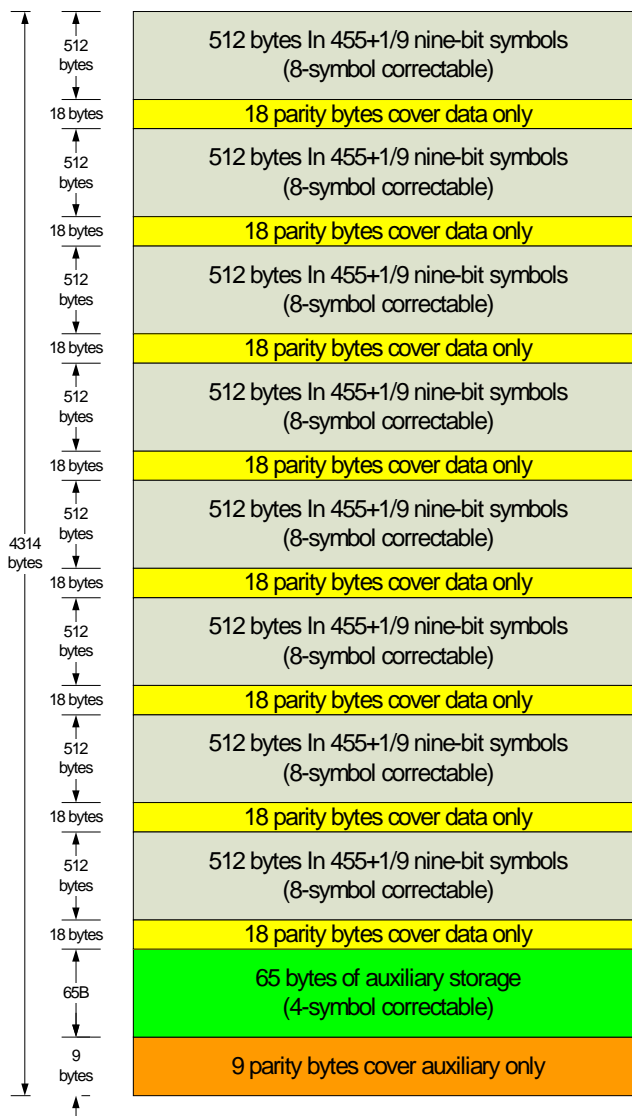


Figure 43. ECC-Protected 4K NAND Page Data—NAND Memory Footprint

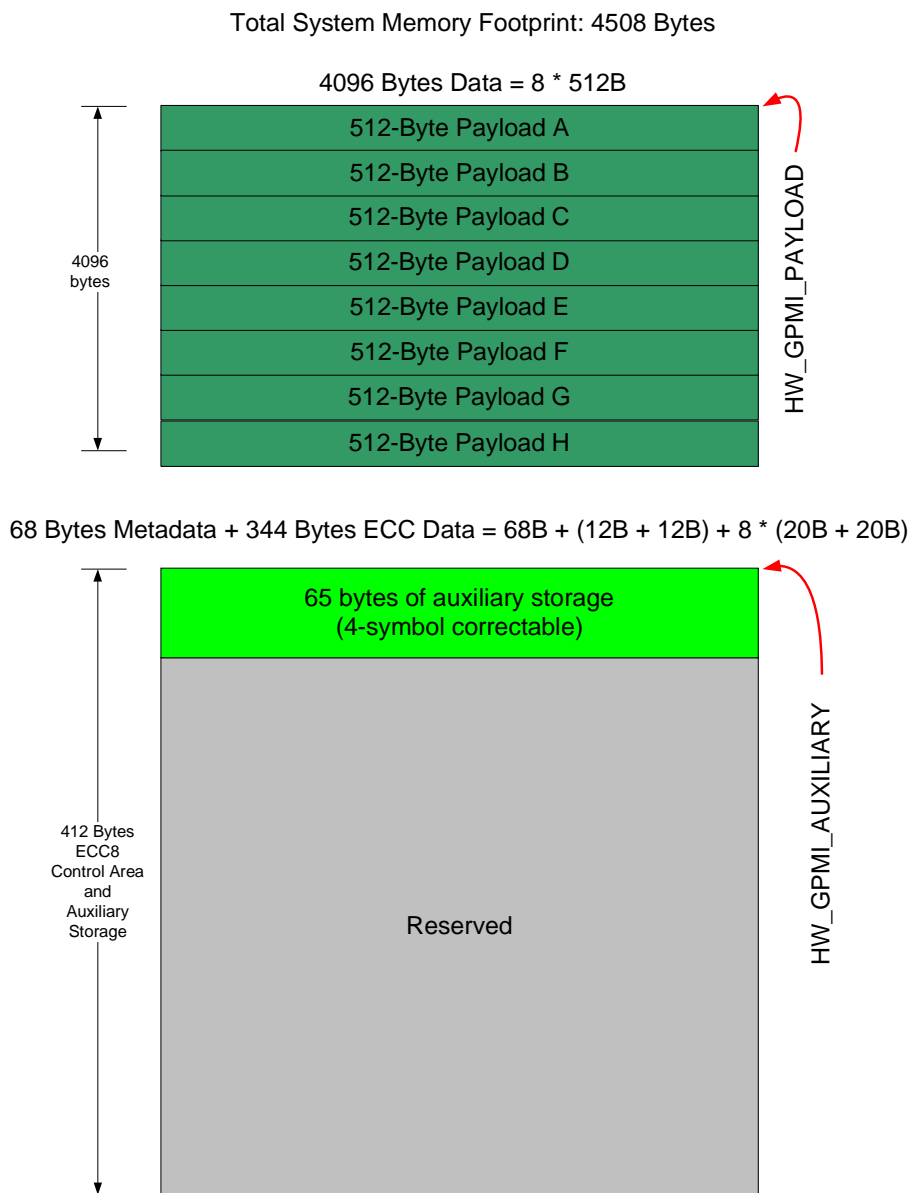


Figure 44. ECC-Protected 4K NAND Page Data—System Memory Footprint

14.2.1. Reed-Solomon ECC Accelerator

The Reed-Solomon algorithm used in ECC8 is capable of correcting up to 8 nine-bit symbols in a 512-byte block. Thus, up to 72 bits in error can be corrected in a 512-byte block, provided they are clustered within no more than 8 nine-bit symbols.

- 2K pages have four 512-byte data blocks (+ 9 bytes parity) and one 19-byte auxiliary block (+ 9 bytes parity).
- 4K pages have eight 512-byte data blocks (+ 18 bytes parity) and one 65-byte auxiliary block (+ 9 bytes parity).

To understand how the Reed-Solomon algorithm is implemented on the STMP3770, consider the case where there are eight 512-byte data blocks located in the on-chip RAM that need to be written to a NAND flash device. Further, assume that there is a 65-byte metadata block that needs to be written to the NAND device. Further assume that the NAND is a 4K page device.

Normal DMA channel command word processing in the APBH DMA allows buffers to start on arbitrary byte boundaries within system memory, i.e., buffers are byte-aligned. In operation with the ECC8 engine, the DMA channel command word processing requires the buffers to start on word boundaries within system memory. Specifically, the HW_GPMI_PAYLOAD and HW_GPMI_AUXILIARY pointers need to be word-aligned for proper ECC8 operation. If those pointers are non-word-aligned, then the ECC8 engine will not operate properly and could possibly corrupt system memory in the adjoining memory regions.

Assume that the data is stored in system memory in the layout shown in the memory footprint of [Figure 44](#). (Note that the data residing in system memory needs to be word-aligned.) In programming the GPMI to write to the NAND, the DMA must be programmed with two DMA descriptors: one that points to the beginning of the PAYLOAD data area and a second to point to the AUXILIARY metadata block. This programming is set up exactly as for the STMP36xx version of the GPMI. Program the GPMI to write these blocks to the NAND, and in addition, program the GPMI to run in ECC8 write mode by setting the following:

```
HW_GPMI_ECCCTRL = BV_FLD(GPMI_ECCCTRL,ECC_CMD,ENCODE_8_BIT) |
                  BV_FLD(GPMI_ECCCTRL,ENABLE_ECC,ENABLE) |
                  BF_GPMI_ECCCTRL_BUFFER_MASK (0x1FF);
HW_GPMI_ECCCOUNT = BF_GPMI_ECCCOUNT_COUNT (8*(512+18) + (65+9))
```

NOTE: The buffer mask value used to specify which data blocks and/or auxiliary block is involved in a transaction must be contiguous i.e., the data blocks and/or auxiliary block need to be consecutive. For example, a transaction involving only data blocks 0, 1, and 2 (buffer mask value = 0x007) is legal, while a transaction of data blocks 1, 2, 4, 6, plus the auxiliary block (buffer mask value = 0x155) is illegal. Illegal buffer mask values will cause improper and undefined system behavior.

Set the first DMA command transfer size to (8*512) bytes. Set the second DMA command transfer size to 65 bytes.

In this mode, the GPMI and the ECC8 collaborate to compute the 16-symbol (18-byte) parity values that must be written to the NAND at the end of each of the eight payload data blocks. In addition, the ECC8 calculates the 8-symbol (9-byte) parity value to be appended to the 65-byte metadata block on the NAND device.

Programming the ECC8 for NAND writes consists largely of removing the soft reset and clock gates from HW_ECC8_CTRL_SFTRST and HW_ECC8_CTRL_CLKGATE. The bulk of the programming is actually applied to

the GPMI via PIO operations embedded in its DMA command structures. This has a subtle implication when writing to the GPMI ECC8 registers: access to the ECC8 registers must be written in progressive register order. Thus, to write to the HW_GPMI_ECCCOUNT register, write first (in order) to registers HW_GPMI_CTRL0, HW_GPMI_COMPARE, and HW_GPMI_ECCCTRL before writing to HW_GPMI_ECCCOUNT. These additional register writes need to be accounted for in the CMDWORDS field of the respective DMA channel command register. See [Section 13.4](#) for the GPMI register descriptions.

When the DMA commands complete, the 4K NAND page will have been written in the format shown in [Figure 43](#). Except for diagnostic operations, normal transfers would never read or write the NAND in any mode other than its ECC mode. It is possible to bypass the parity generation and write “RAW” data to the NAND by not turning on the ECC functions.

To summarize the detailed operation, an 18-byte Reed-Solomon parity field is appended in a 4K page at the end of each of the eight 512-byte blocks. Notice that 4K NAND devices have 4096 byte data areas plus 218-byte spare area for each “4KB” NAND flash page. In addition, a 9-byte parity field is written to the end of the 65-byte metadata block.

Assume that the GPMI media interface is used to write the resultant $((8 \times 512) + 65)$ bytes of data from on-chip memory to the NAND flash device. The GPMI and ECC8 then collaborate to generate an additional $((8 \times 18) + 9)$ bytes of parity information.

- Channel commands in APBH DMA Channels 4, 5, 6, or 7 are used to point to the data block in either on-chip or off-chip RAM (as shown in [Figure 47](#)).

To program the GPMI and ECC8 to read that same 4K page of NAND data back from the NAND to a buffer in the system memory, first reserve a system memory buffer like the one depicted in [Figure 44](#). (Note that the reserved system memory buffers need to be word-aligned.) The GPMI DMA engine is not used for the data transfer, see below. Instead, it is used to convey a sequence of commands to the GPMI as DMA PIO operations. Some of the information conveyed to the GPMI is made available to the ECC8 engine to process the NAND read.

In particular, the address of the PAYLOAD BUFFER and the address of the AUXILIARY buffer are written to the GPMI PIO space, not to the ECC8 PIO space. Thus, the normal multi-NAND DMA based device interleaving is preserved, i.e., four NANDs on four separate chip selects can be scheduled for read or write operations using the ECC8. Whichever channel finishes its ready wait first and enters the DMA arbiter with its lock bit set will “own” the GPMI command interface and through it will own the ECC8 resources for the duration of its processing.

So, a nearly standard read DMA descriptor chain is used for the NAND read transfer, including the ready wait commands. The DMA command that kicks off the GPMI has a few extra PIO words attached to preload the HW_GPMI_ECCCTRL, HW_GPMI_ECCCOUNT, HW_GPMI_PAYLOADm and HW_GPMI_AUXILIARY registers.

When the data is read from the NAND by the GPMI, it is passed to the ECC8. Inside the ECC8, the data is copied to the payload buffer or auxiliary buffer using the AHB bus master in the ECC8. The ECC8 needs some work space in system memory to hold intermediate results. These elements are allocated in the auxiliary buffer pointed to by HW_GPMI_AUXILIARY.

Notice that programming the ECC8 for NAND reads consists largely of removing the soft reset and clock gates from HW_ECC8_CTRL and clearing the HW_ECC8_CTRL_COMPLETE_IRQ, since most of the actual programming is accomplished through PIO operations included in GPMI DMA command structures.

Set HW_ECC8_CTRL_COMPLETE_IRQ_EN to 1, then start the GPMI's DMA, and let it run. The ECC8 interrupts the CPU after completing the entire transaction. This could be a single 512-byte block if desired or the entire 4K page of payload data and the 65 bytes of metadata. It also could be just the metadata block. Note that the metadata is protected by its own 9-byte parity so that reading metadata is very efficient.

The ECC8 status registers indicate the quality of the data read into each of the nine blocks with a four-bit code.

- A value of 0 means no errors occurred on the block.
- A value of 1–8 means that correctable errors occurred but the data was repaired by the bus master.
- A value of 0xC means that this block was not specified on the read transaction.
- A value of 0xE means that an uncorrectable error occurred on that block.
- A value of 0xF means that this block contains all ones and is therefore considered to be an ERASED block.
- A summary status quickly tells if **any** block in the page had an uncorrectable error.

14.2.2. *Reed-Solomon ECC Encoding for NAND Writes*

The RS encoder flowchart in [Figure 45](#) shows the detailed steps involved in programming and using the ECC8 encoder. This flowchart shows how to use the ECC8 block with the GPMI.

To use the ECC8 encoder with the GPMI's DMA, create a DMA command chain containing ten descriptor structures, as shown in [Figure 47](#) and detailed in the DMA structure code example that follows it in [Section 14.2.2.1](#). The ten descriptors perform the following tasks:

1. Disable the ECC8 block (in case it was enabled) and issue NAND write setup command byte (under "CLE") and address bytes (under "ALE").
2. Configure and enable the ECC8 block and write the data payload.
3. Write the auxiliary payload.
4. Disable the ECC8 block and issue NAND write execute command byte (under "CLE").
5. Wait for the NAND device to finish writing the data by watching the ready signal.
6. Check for NAND time-out via "PSENSE".
7. Issue NAND status command byte (under "CLE").
8. Read the status and compare against expected.
9. If status is incorrect/incomplete, branch to error handling descriptor chain.
10. Otherwise, write is complete and emit GPMI interrupt.

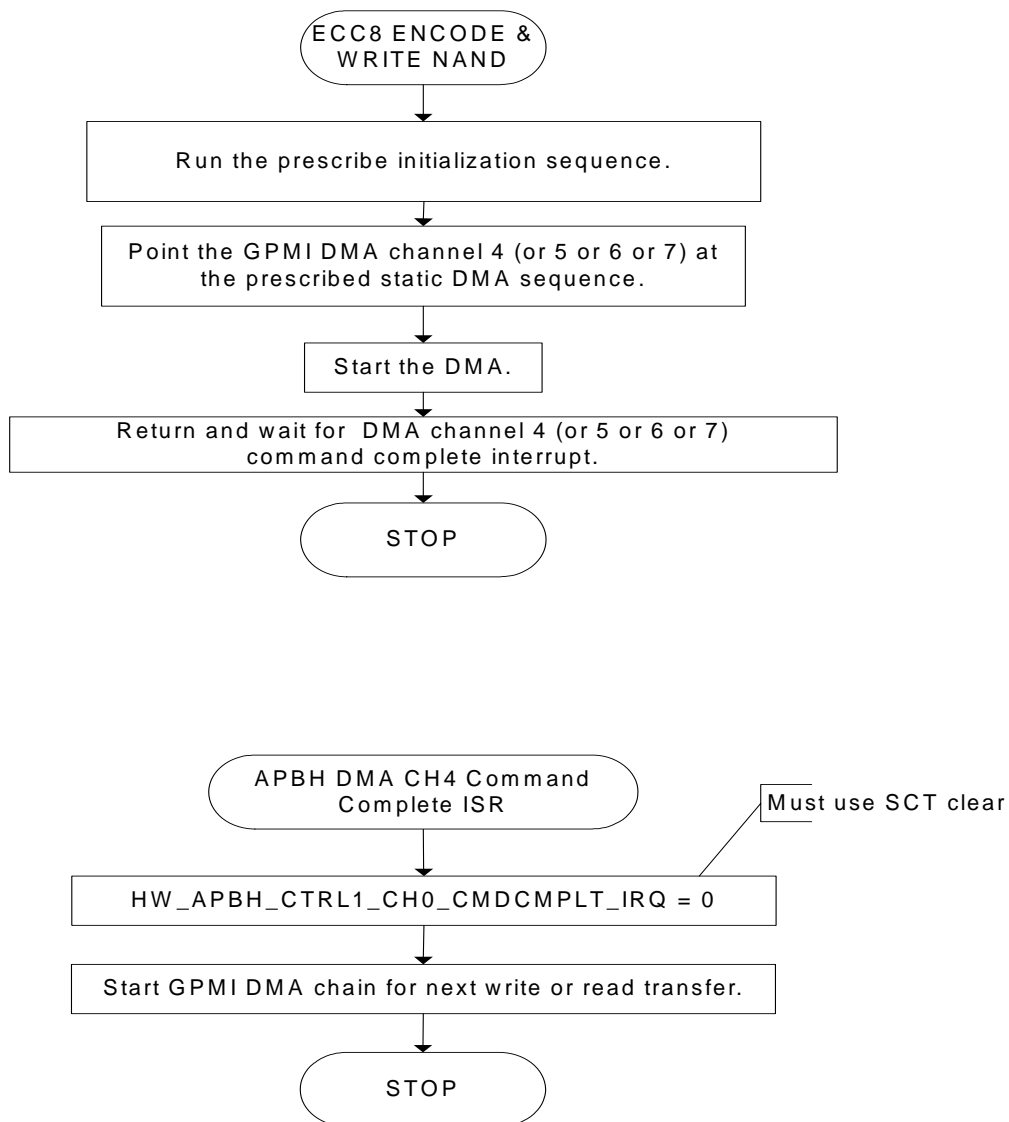


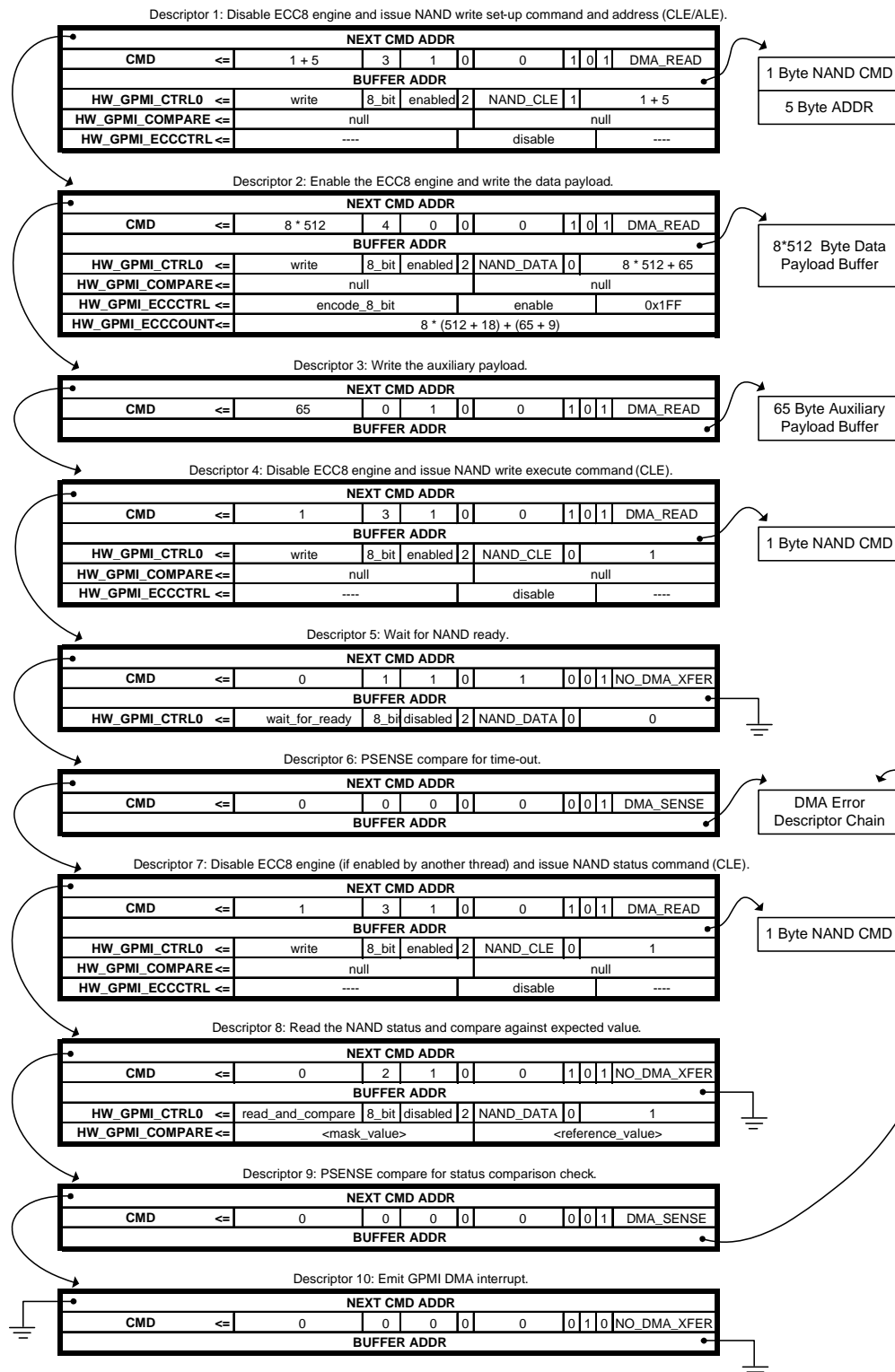
Figure 45. ECC8 Reed-Solomon Encode Flowchart

Descriptor Legend										
NEXT CMD ADDR										
CMD	<=	xfer_count	cmdwords	wait4endcmd	semaphore	nandwait4ready	nandlock	irqoncmplt	chain	command
BUFFER ADDR										
HW_GPMI_CTRL0	<=	command_mode	word_length	lock_cs	CS	address	address_increment	xfer_count		
HW_GPMI_COMPARE	<=	mask				reference				
HW_GPMI_ECCCTRL	<=	ecc_cmd			enable_ecc				buffer_mask	
HW_GPMI_ECCCOUNT										
HW_GPMI_PAYLOAD										
HW_GPMI_AUXILIARY										

Note: Refer to this legend when examining [Figure 47](#) and [Figure 50](#).

Figure 46. ECC8 DMA Descriptor Legend

STMP3770



Note: To interpret the fields in this diagram, see [Figure 46](#) for the descriptor legend.

Figure 47. ECC8 Reed-Solomon Encode DMA Descriptor Chain

14.2.2.1. DMA Structure Code Example

The following code sample illustrates the coding for one write transaction involving 4096 bytes of data payload (eight 512-byte blocks) and 65 bytes of auxiliary payload (also referred to as *metadata*) to a 4K NAND page sitting on GPMI CS2.

```
//-----
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----
typedef struct {
    // DMA related fields
    unsigned int dma_nxtcmdar;
    unsigned int dma_cmd;
    unsigned int dma_bar;

    // GPMI related fields
    unsigned int gpmi_ctrl0;
    unsigned int gpmi_compare;
    unsigned int gpmi_eccctrl;
    unsigned int gpmi_ecccount;
    unsigned int gpmi_data_ptr;
    unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;

//-----
// allocate 10 descriptors for doing a NAND ECC Write
//-----
GENERIC_DESCRIPTOR write[10];

//-----
// DMA descriptor pointer to handle error conditions from psense checks
//-----
unsigned int * dma_error_handler;

//-----
// 8 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
// byte 0 is write setup command
// bytes 1-5 is the NAND address
// byte 6 is write execute command
// byte 7 is status command
//-----
unsigned char nand_cmd_addr_buffer[8];

//-----
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----
unsigned int write_payload_buffer[(4096/4)];

//-----
// 65 byte meta-data to be written to NAND
// needs to be word aligned
//-----
unsigned int write_aux_buffer[65];

//-----
// Descriptor 1: issue NAND write setup command (CLE/ALE)
//-----
write[0].dma_nxtcmdar = &write[1]; // point to the next descriptor

write[0].dma_cmd = BF_APBH_ChN_CMD_XFER_COUNT (1 + 5) | // 1 byte command, 5 byte address
                  BF_APBH_ChN_CMD_CMDWORDS (3) | // send 3 words to the GPMI
                  BF_APBH_ChN_CMD_WAIT4ENDCMD (1) | // wait for command to finish before continuing
                  BF_APBH_ChN_CMD_SEMAPHORE (0) |
                  BF_APBH_ChN_CMD_NANDWAIT4READY(0) |
                  BF_APBH_ChN_CMD_NANDLOCK (1) | // prevent other DMA channels from taking over
                  BF_APBH_ChN_CMD_IRQONCMPLT (0) |
                  BF_APBH_ChN_CMD_CHAIN (1) | // follow chain to next command
                  BV_FLD(APBH_ChN_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND

write[0].dma_bar = &nand_cmd_addr_buffer; // byte 0 write setup, bytes 1 - 5 NAND address

// 3 words sent to the GPMI
write[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE) | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT) |
                     BV_FLD(GPMI_CTRL0, LOCK_CS, ENABLED) |
                     BF_GPMI_CTRL0_CS (2) | // must correspond to NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS, NAND_CLE) |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT (1) | // send command and address
                     BF_GPMI_CTRL0_XFER_COUNT (1 + 5); // 1 byte command, 5 byte address

write[0].gpmi_compare = NULL; // field not used but necessary to set eccctrl

write[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//-----
```

STMP3770

```

// Descriptor 2: write the data payload (DATA)
//-----
write[1].dma_nextcmdar = &write[2]; // point to the next descriptor

write[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (8*512) | // NOTE: DMA transfer only the data payload
                  BF_APBH_CHn_CMD_CMDWORDS (4) | // send 4 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (0) | // DON'T wait to end, wait in the next descriptor
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)
                  BF_APBH_CHn_CMD_NANDLOCK (1) // maintain resource lock
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND

write[1].dma_bar = &write_payload_buffer; // pointer for the 4K byte data area

// 4 words sent to the GPMI
write[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE) // write to the NAND
                    BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                    BV_FLD(GPMI_CTRL0, LOCK_CS, ENABLED)
                    BF_GPMI_CTRL0_CS (2) // must correspond to NAND CS used
                    BV_FLD(GPMI_CTRL0, ADDRESS, NAND_DATA)
                    BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                    BF_GPMI_CTRL0_XFER_COUNT (8*512+65); // NOTE: this field contains the total amount
                                                         // DMA transferred (8 data and 1 aux blocks)
                                                         // to GPMI!

write[1].gpmi_compare = NULL; // field not used but necessary to set eccctrl

write[1].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD, ENCODE_8_BIT) // specify t = 8 mode
                      BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE) // enable ECC module
                      BF_GPMI_ECCCTRL_BUFFER_MASK (0x1FF); // write all 8 data blocks and 1 aux block

write[1].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(8*(512+18)+(65+9)); // specify number of bytes written to NAND
                                                                    // NOTE: the extra 8*(18)+9 bytes are parity
                                                                    // bytes generated by the ECC block.

//-----
// Descriptor 3: write the aux payload (DATA)
//-----
write[2].dma_nextcmdar = &write[3]; // point to the next descriptor

write[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (65) // NOTE: DMA transfer only the aux block
                  BF_APBH_CHn_CMD_CMDWORDS (0) // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (1) // wait for command to finish before continuing
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)
                  BF_APBH_CHn_CMD_NANDLOCK (1) // maintain resource lock
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND

write[2].dma_bar = &write_aux_buffer; // pointer for the 65 byte meta data area

//-----
// Descriptor 4: issue NAND write execute command (CLE)
//-----
write[3].dma_nextcmdar = &write[4]; // point to the next descriptor

write[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (1) // 1 byte command
                  BF_APBH_CHn_CMD_CMDWORDS (3) // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (1) // wait for command to finish before continuing
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)
                  BF_APBH_CHn_CMD_NANDLOCK (1) // maintain resource lock
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND

write[3].dma_bar = &nand_cmd_addr_buffer[6]; // point to byte 6, write execute command

// 3 words sent to the GPMI
write[3].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE) // write to the NAND
                    BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                    BV_FLD(GPMI_CTRL0, LOCK_CS, ENABLED)
                    BF_GPMI_CTRL0_CS (2) // must correspond to NAND CS used
                    BV_FLD(GPMI_CTRL0, ADDRESS, NAND_CLE)
                    BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                    BF_GPMI_CTRL0_XFER_COUNT (1); // 1 byte command

write[3].gpmi_compare = NULL; // field not used but necessary to set eccctrl

write[3].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block

//-----
// Descriptor 5: wait for ready (CLE)
//-----
write[4].dma_nextcmdar = &write[5]; // point to the next descriptor

```

```

write[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (0)           // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS        (1)           // send 1 word to the GPMI
BF_APBH_CHn_CMD_WAIT4ENDCMD (1)           | // wait for command to finish before continuing
BF_APBH_CHn_CMD_SEMAPHORE   (0)           |
BF_APBH_CHn_CMD_NANDWAIT4READY(1)         | // wait for nand to be ready
BF_APBH_CHn_CMD_NANDLOCK    (0)           | // relinquish nand lock
BF_APBH_CHn_CMD_IRQONCMPLT  (0)           |
BF_APBH_CHn_CMD_CHAIN       (1)           | // follow chain to next command
BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER); // no dma transfer

write[4].dma_bar = NULL; // field not used

// 1 word sent to the GPMI
write[4].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) // wait for NAND ready
                   BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                   BV_FLD(GPMI_CTRL0, LOCK_CS, DISABLED)
                   BF_GPMI_CTRL0_CS (2) // must correspond to NAND CS used
                   BV_FLD(GPMI_CTRL0, ADDRESS, NAND_DATA)
                   BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                   BF_GPMI_CTRL0_XFER_COUNT (0);

//-----
// Descriptor 6: psense compare (time out check)
//-----
write[5].dma_nxtcmdar = &write[6]; // point to the next descriptor

write[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (0)           // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS        (0)           // no words sent to GPMI
BF_APBH_CHn_CMD_WAIT4ENDCMD (0)           | // do not wait to continue
BF_APBH_CHn_CMD_SEMAPHORE   (0)           |
BF_APBH_CHn_CMD_NANDWAIT4READY(0)         |
BF_APBH_CHn_CMD_NANDLOCK    (0)           |
BF_APBH_CHn_CMD_IRQONCMPLT  (0)           |
BF_APBH_CHn_CMD_CHAIN       (1)           | // follow chain to next command
BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE); // perform a sense check

write[5].dma_bar = dma_error_handler; // if sense check fails, branch to error handler

//-----
// Descriptor 7: issue NAND status command (CLE)
//-----
write[6].dma_nxtcmdar = &write[7]; // point to the next descriptor

write[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (1)           // 1 byte command
                  BF_APBH_CHn_CMD_CMDWORDS        (3)           // send 3 words to the GPMI
BF_APBH_CHn_CMD_WAIT4ENDCMD (1)           | // wait for command to finish before continuing
BF_APBH_CHn_CMD_SEMAPHORE   (0)           |
BF_APBH_CHn_CMD_NANDWAIT4READY(0)         |
BF_APBH_CHn_CMD_NANDLOCK    (1)           | // prevent other DMA channels from taking over
BF_APBH_CHn_CMD_IRQONCMPLT  (0)           |
BF_APBH_CHn_CMD_CHAIN       (1)           | // follow chain to next command
BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND

write[6].dma_bar = &nand_cmd_addr_buffer[7]; // point to byte 7, status command

write[6].gpmi_compare = NULL; // field not used but necessary to set eccctrl

write[6].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block

// 3 words sent to the GPMI
write[6].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE) // write to the NAND
                   BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                   BV_FLD(GPMI_CTRL0, LOCK_CS, ENABLED)
                   BF_GPMI_CTRL0_CS (2) // must correspond to NAND CS used
                   BV_FLD(GPMI_CTRL0, ADDRESS, NAND_CLE)
                   BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                   BF_GPMI_CTRL0_XFER_COUNT (1); // 1 byte command

//-----
// Descriptor 8: read status and compare (DATA)
//-----
write[7].dma_nxtcmdar = &write[8]; // point to the next descriptor

write[7].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (0)           // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS        (2)           // send 2 words to the GPMI
BF_APBH_CHn_CMD_WAIT4ENDCMD (1)           | // wait for command to finish before
continuing
BF_APBH_CHn_CMD_SEMAPHORE   (0)           |
BF_APBH_CHn_CMD_NANDWAIT4READY(0)         |
BF_APBH_CHn_CMD_NANDLOCK    (1)           | // maintain resource lock
BF_APBH_CHn_CMD_IRQONCMPLT  (0)           |
BF_APBH_CHn_CMD_CHAIN       (1)           | // follow chain to next command
BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER); // no dma transfer

write[7].dma_bar = NULL; // field not used

// 2 word sent to the GPMI
write[7].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ_AND_COMPARE) | // read from the NAND and compare to
expect

```



```

        BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
        BV_FLD(GPMI_CTRL0, LOCK_CS, DISABLED)
        BF_GPMI_CTRL0_CS (2) // must correspond to NAND CS used
        BV_FLD(GPMI_CTRL0, ADDRESS, NAND_DATA)
        BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
        BF_GPMI_CTRL0_XFER_COUNT (1);

write[7].gpmi_compare = <MASK_AND_REFERENCE_VALUE>; // NOTE: mask and reference values are
NAND // SPECIFIC to evaluate the NAND status

//-----
// Descriptor 9: psense compare (time out check)
//-----
write[8].dma_nextcmdar = &write[9]; // point to the next descriptor

write[8].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (0) // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS (0) // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (0) // do not wait to continue
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)
                  BF_APBH_CHn_CMD_NANDLOCK (0) // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE); // perform a sense check

write[8].dma_bar = dma_error_handler; // if sense check fails, branch to error handler

//-----
// Descriptor 10: emit GPMI interrupt
//-----
write[9].dma_nextcmdar = NULL; // not used since this is last descriptor

write[9].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (0) // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS (0) // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (0) // do not wait to continue
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)
                  BF_APBH_CHn_CMD_NANDLOCK (0)
                  BF_APBH_CHn_CMD_IRQONCMPLT (1) // emit GPMI interrupt
                  BF_APBH_CHn_CMD_CHAIN (0) // terminate DMA chain processing
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER); // no dma transfer

```

14.2.2.2. Using the ECC8 Encoder

To use the ECC8 encoder, first turn off the module-wide soft reset bit in both the GPMI and ECC8 blocks 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 registers. Turn off the ECC8 bus master soft reset bit (bit 29). Turn off the clock gate bits.

Program the remainder of the GPMI, ECC8 and APBH DMA as follows:

```

// bring APBH out of reset
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_SFRST);
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_CLKGATE);

// bring ecc8 out of reset
HW_ECC8_CTRL_CLR(BM_ECC8_CTRL_SFTRST);
HW_ECC8_CTRL_CLR(BM_ECC8_CTRL_CLKGATE);
HW_ECC8_CTRL_CLR(BM_ECC8_CTRL_AHBM_SFTRST);

// bring gpmi out of reset
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
HW_GPMI_CTRL1_SET(BM_GPMI_CTRL1_DEV_RESET); // deassert in case
// anyone's hooked up to the reset pin

// enable pinctrl
HW_PINCTRL_CTRL_WR(0x00000000);

// enable GPMI through alt pin wiring
HW_PINCTRL_MUXSEL0_CLR(0xff000000);
HW_PINCTRL_MUXSEL0_SET(0xaa000000);

// to use the primary pins do the following
// HW_PINCTRL_MUXSEL4_CLR(0xff000000);
// HW_PINCTRL_MUXSEL4_SET(0x55000000);

```



```
// enable gpmi pins
HW_PINCTRL_MUXSEL0_CLR(0x0000ffff); // data bits
HW_PINCTRL_MUXSEL1_CLR(0x000fffff); // control bits
```

Note that for writing NANDs (ECC encoding), only GPMI DMA command complete interrupts are used. The ECC8 engine is used for writing to the NAND but never produces an interrupt. From the sample code in [Section 14.2.2.1](#):

- DMA descriptor 1 prepares the NAND for data write by using the GPMI to issue a write setup command byte under “CLE”, then sends a 5-byte address under “ALE”. The ECC8 engine is disabled and not used for these commands.
- DMA descriptor 2 enables the ECC8 engine for t=8 encoding to begin the initial writing of the NAND data by specifying where the data payload is coming from in system memory.
- DMA descriptor 3 continues the writing of NAND data by specifying where the auxiliary data is coming from in system memory.
- DMA descriptor 4 issues the write commit command byte under “CLE” to the NAND.
- DMA descriptor 5 waits for the NAND to complete the write commit/transfer by watching the NAND’s ready line status. This descriptor relinquishes the NANDLOCK on the GPMI to enable the other DMA channels to initiate NAND transactions on different NAND CS lines.
- DMA descriptor 7 issues a NAND status command byte under “CLE” to check the status of the NAND device following the page write.
- DMA descriptor 8 reads back the NAND status and compares the status with an expected value. If there are differences, then the DMA processing engine follows an error-handling DMA descriptor path.
- DMA descriptor 9 disables the ECC8 engine and emits a GPMI interrupt to indicate that the NAND write has been completed.

14.2.3. Reed-Solomon ECC Decoding for NAND Reads

When a page is read from NAND flash, RS syndromes will be computed and, if correctable errors are found, they will be corrected on a per block basis within the NAND page. This decoding process is fully overlapped with other NAND data reads and with CPU execution. The RS decoder flowchart in Figure 48 shows the steps involved in programming the ECC8 Reed-Solomon decoder. The hardware flow of reading and decoding a 512-byte page encoded for $t=8$ error correction (18 parity bytes) is shown in Figure 49.

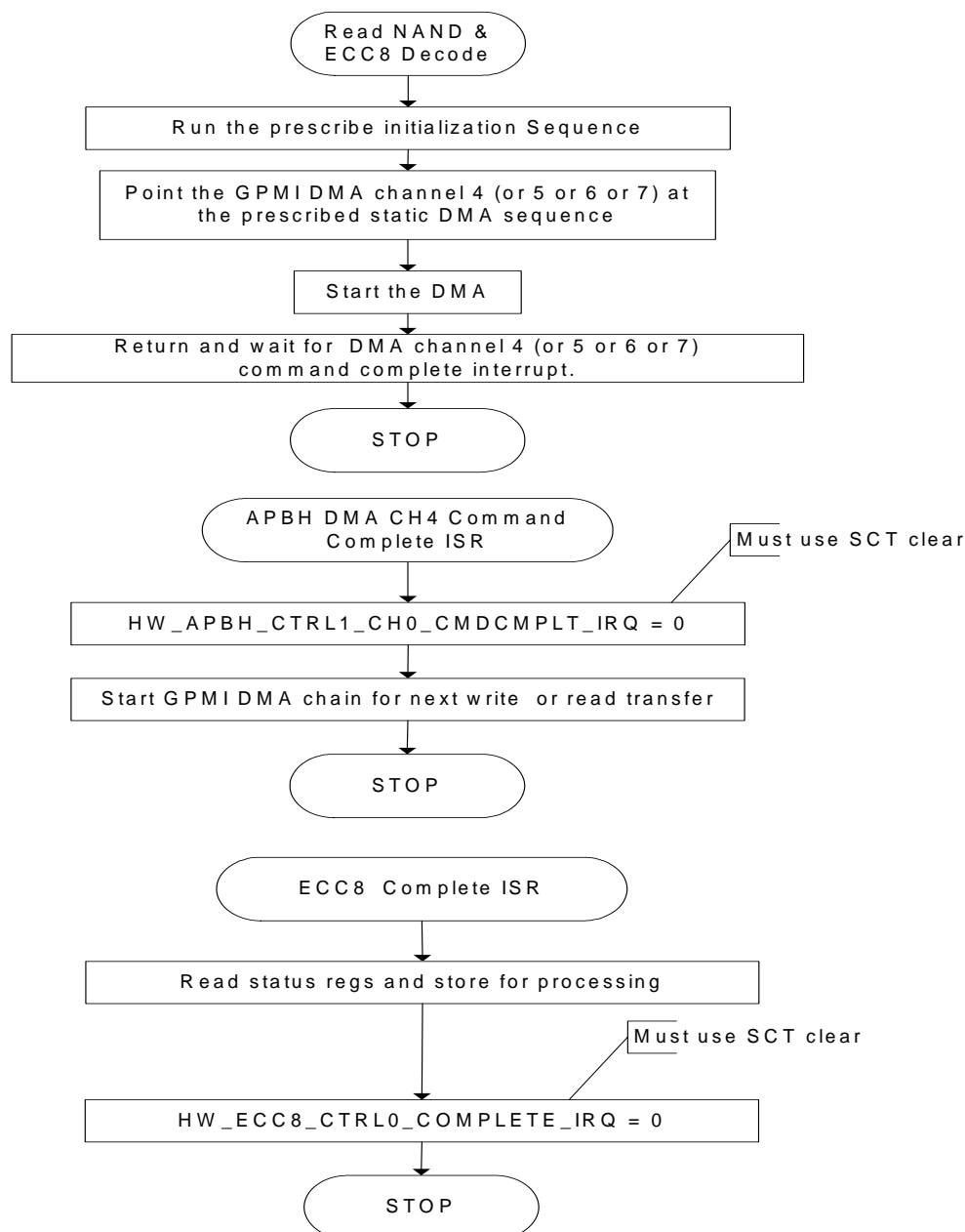
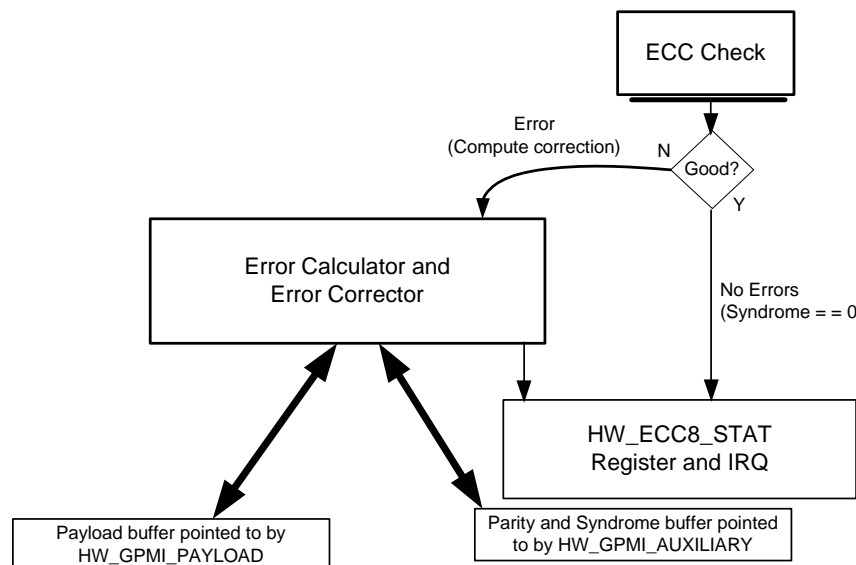


Figure 48. ECC8 Reed-Solomon Decode Flowchart



NOTES:

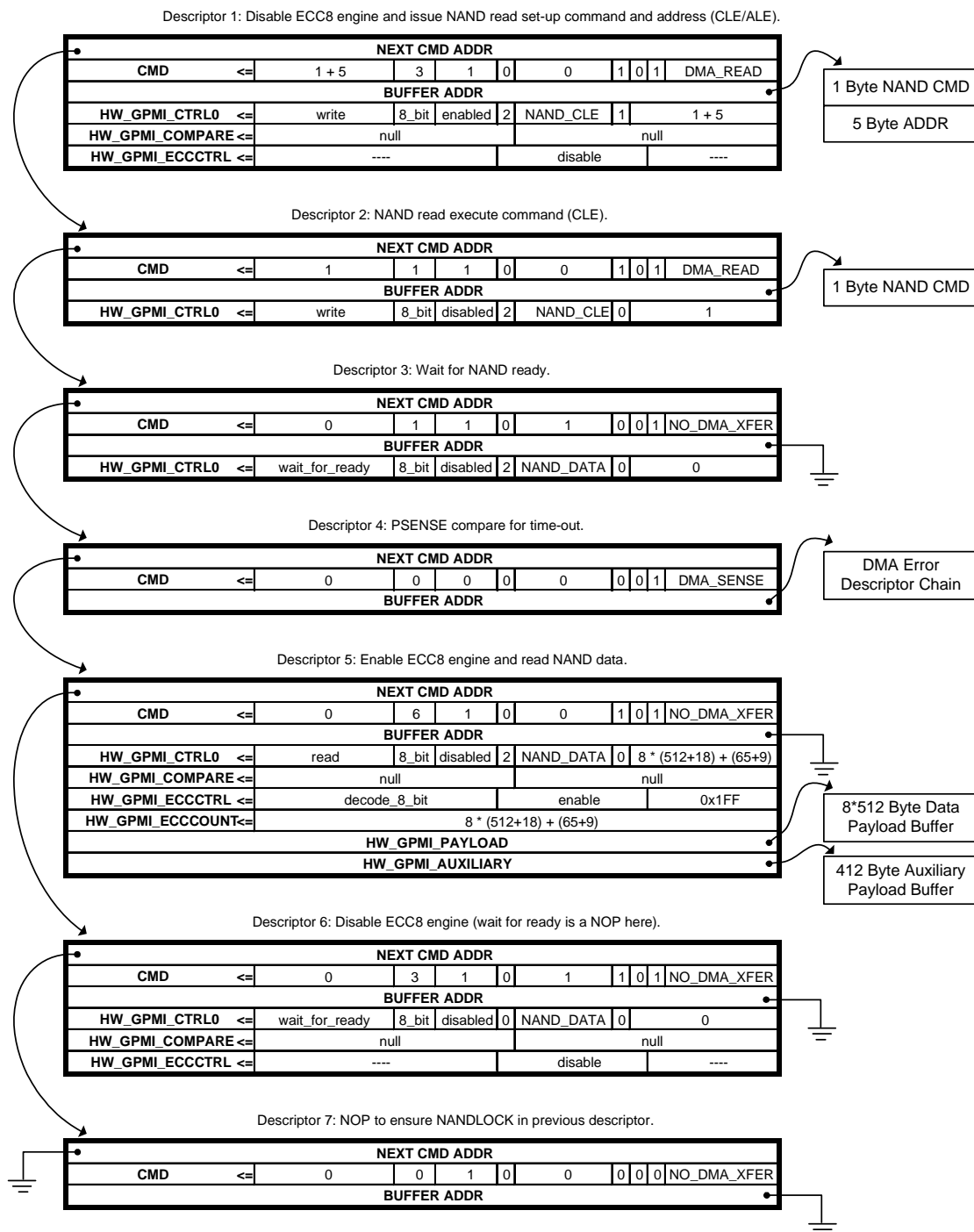
This diagram describes a decoder for reading and correcting a 512-byte data block encoded with $t=8$ error correction. ECC8 bus master writes the bytes from the NAND to system memory. The error corrector performs read-modify-writes on these system memory buffers as necessary.

Figure 49. ECC8 Reed-Solomon Block Coding—Decoder for $t=8$

Conceptually, an APHB DMA Channel 4, 5, 6, or 7 command chain with seven command structures linked together is used to perform the RS decode operation (as shown in [Figure 50](#)). NOTE: The GPMI's DMA command structures controls the ECC8 decode operation.

To use the ECC8 decoder with the GPMI's DMA, create a DMA command chain containing seven descriptor structures, as shown in [Figure 50](#) and detailed in the DMA structure code example that follows it in [Section 14.2.3.1](#). The seven DMA descriptors perform the following tasks:

1. Issue NAND read setup command byte (under "CLE") and address bytes (under "ALE").
2. Issue NAND read execute command byte (under "CLE").
3. Wait for the NAND device to complete accessing the block data by watching the ready signal.
4. Check for NAND time-out via "PSENSE".
5. Configure and enable the ECC8 block and read the NAND block data.
6. Disable the ECC8 block.
7. Descriptor NOP to allow NANDLOCK in the previous descriptor to the thread-safe.



Note: To interpret the fields in this diagram, see [Figure 46](#) for the descriptor legend.

Figure 50. ECC8 Reed-Solomon Decode DMA Descriptor Chain

14.2.3.1. DMA Structure Code Example

The following sample code illustrates the coding for one read transaction, consisting of a seven DMA command structure chain for reading all 4096 bytes of payload data (eight 512-byte blocks) and 65 bytes of metadata with the associative parity bytes ($8 * (18) + 9$) from a 4K NAND page sitting on GPMI CS2.

```
//-----
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//-----
typedef struct {
    // DMA related fields
    unsigned int dma_nxtcmdar;
    unsigned int dma_cmd;
    unsigned int dma_bar;

    // GPMI related fields
    unsigned int gpmi_ctrl0;
    unsigned int gpmi_compare;
    unsigned int gpmi_eccctrl;
    unsigned int gpmi_ecccount;
    unsigned int gpmi_data_ptr;
    unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;

//-----
// allocate 7 descriptors for doing a NAND ECC Read
//-----
GENERIC_DESCRIPTOR read[7];

//-----
// DMA descriptor pointer to handle error conditions from psense checks
//-----
unsigned int * dma_error_handler;

//-----
// 7 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
// byte 0 is read setup command
// bytes 1-5 is the NAND address
// byte 6 is read execute command
//-----
unsigned char nand_cmd_addr_buffer[7];

//-----
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//-----
unsigned int read_payload_buffer[(4096/4)];

//-----
// 412 byte auxiliary buffer used for reads
// needs to be word aligned
//-----
unsigned int read_aux_buffer[(412/4)];

//-----
// Descriptor 1: issue NAND read setup command (CLE/ALE)
//-----
read[0].dma_nxtcmdar = &read[1]; // point to the next descriptor

read[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (1 + 5) | // 1 byte command, 5 byte address
                  BF_APBH_CHn_CMD_CMDWORDS (3) | // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (1) | // wait for command to finish before continuing
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY (0)
                  BF_APBH_CHn_CMD_NANDLOCK (1) | // prevent other DMA channels from taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND

read[0].dma_bar = &nand_cmd_addr_buffer; // byte 0 read setup, bytes 1 - 5 NAND address

// 3 words sent to the GPMI
read[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE) | // write to the NAND
                    BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                    BV_FLD(GPMI_CTRL0, LOCK_CS, ENABLED)
                    BF_GPMI_CTRL0_CS (2) | // must correspond to NAND CS used
                    BV_FLD(GPMI_CTRL0, ADDRESS, NAND_CLE)
                    BF_GPMI_CTRL0_ADDRESS_INCREMENT (1) | // send command and address
                    BF_GPMI_CTRL0_XFER_COUNT (1 + 5); // 1 byte command, 5 byte address

read[0].gpmi_compare = NULL; // field not used but necessary to set eccctrl

read[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
```

STMP3770

```

//-----
// Descriptor 2: issue NAND read execute command (CLE)
//-----
read[1].dma_nextcmdar = &read[2]; // point to the next descriptor

read[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (1) // 1 byte read command
                  BF_APBH_CHn_CMD_CMDWORDS (1) // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (1) // wait for command to finish before continuing
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY (0)
                  BF_APBH_CHn_CMD_NANDLOCK (1) // prevent other DMA channels from taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND

read[1].dma_bar = &nand_cmd_addr_buffer[6]; // point to byte 6, read execute command

// 1 word sent to the GPMI
read[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE) // write to the NAND
                    BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                    BV_FLD(GPMI_CTRL0, LOCK_CS, DISABLED)
                    BF_GPMI_CTRL0_CS (2) // must correspond to NAND CS used
                    BV_FLD(GPMI_CTRL0, ADDRESS, NAND_CLE)
                    BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                    BF_GPMI_CTRL0_XFER_COUNT (1); // 1 byte command

//-----
// Descriptor 3: wait for ready (DATA)
//-----
read[2].dma_nextcmdar = &read[3]; // point to the next descriptor

read[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (0) // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS (1) // send 1 word to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (1) // wait for command to finish before
continuing
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY (1) // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK (0) // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER); // no dma transfer

read[2].dma_bar = NULL; // field not used

// 1 word sent to the GPMI
read[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) // wait for NAND ready
                    BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                    BV_FLD(GPMI_CTRL0, LOCK_CS, DISABLED)
                    BF_GPMI_CTRL0_CS (2) // must correspond to NAND CS used
                    BV_FLD(GPMI_CTRL0, ADDRESS, NAND_DATA)
                    BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                    BF_GPMI_CTRL0_XFER_COUNT (0);

//-----
// Descriptor 4: psense compare (time out check)
//-----
read[3].dma_nextcmdar = &read[4]; // point to the next descriptor

read[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (0) // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS (0) // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (0) // do not wait to continue
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY (0)
                  BF_APBH_CHn_CMD_NANDLOCK (0)
                  BF_APBH_CHn_CMD_IRQONCMPLT (0)
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE); // perform a sense check

read[3].dma_bar = dma_error_handler; // if sense check fails, branch to error handler

//-----
// Descriptor 5: read 4K page plus 65 byte meta-data Nand data
// and send it to ECC block (DATA)
//-----
read[4].dma_nextcmdar = &read[5]; // point to the next descriptor

read[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (0) // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS (6) // send 6 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD (1) // wait for command to finish before
continuing
                  BF_APBH_CHn_CMD_SEMAPHORE (0)
                  BF_APBH_CHn_CMD_NANDWAIT4READY (0)
                  BF_APBH_CHn_CMD_NANDLOCK (1) // prevent other DMA channels from
taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT (0) // ECC block generates ecc8 interrupt
on completion
                  BF_APBH_CHn_CMD_CHAIN (1) // follow chain to next command

```

```

        BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);           // no DMA transfer, ECC block handles
transfer
read[4].dma_bar = NULL;                                       // field not used

// 6 words sent to the GPMI
read[4].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ) // read from the NAND
                   BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                   BV_FLD(GPMI_CTRL0, LOCK_CS, DISABLED)
                   BF_GPMI_CTRL0_CS (2)                      // must correspond to NAND CS used
                   BV_FLD(GPMI_CTRL0, ADDRESS, NAND_DATA)
                   BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                   BF_GPMI_CTRL0_XFER_COUNT (8*(512+18)+(65+9)); // eight 512 byte data blocks (plus
parity, t = 8)                                                // and one 65 byte aux block (plus
parity, t = 4)

read[4].gpmi_compare = NULL;                                   // field not used but necessary to set
eccctrl

// GPMI ECCCTRL PIO This launches the 4K byte transfer through ECC8's
// bus master. Setting the ECC_ENABLE bit redirects the data flow
// within the GPMI so that read data flows to the ECC8 engine instead
// of flowing to the GPMI's DMA channel.
read[4].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD, DECODE_8_BIT) // specify t = 8 mode
                    BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       // enable ECC module
                    BF_GPMI_ECCCTRL_BUFFER_MASK (0X1FF);           // read all 8 data blocks and 1 aux block

read[4].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(8*(512+18)+(65+9)); // specify number of bytes read from
NAND

read[4].gpmi_data_ptr = &read_payload_buffer;                 // pointer for the 4K byte data area
read[4].gpmi_aux_ptr = &read_aux_buffer;                       // pointer for the 65 byte aux area +
                                                                // parity and syndrome bytes for both
                                                                // data and aux blocks.

//-----
// Descriptor 6: disable ECC block
//-----
read[5].dma_nextcmdar = &read[6];                             // point to the next descriptor

read[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (0)              // no dma transfer
                 BF_APBH_CHn_CMD_CMDWORDS (3)                // send 3 words to GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD (1)              // wait for command to finish before continuing
                 BF_APBH_CHn_CMD_SEMAPHORE (0)
                 BF_APBH_CHn_CMD_NANDWAIT4READY(1)            // wait for nand to be ready
                 BF_APBH_CHn_CMD_NANDLOCK (1)                 // need nand lock to be thread safe while turn-
off ECC8
                 BF_APBH_CHn_CMD_IRQONCMPLT (0)
                 BF_APBH_CHn_CMD_CHAIN (1)                    // follow chain to next command
                 BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);   // no dma transfer

read[5].dma_bar = NULL;                                       // field not used

// 3 words sent to the GPMI
read[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)
                   BV_FLD(GPMI_CTRL0, WORD_LENGTH, 8_BIT)
                   BV_FLD(GPMI_CTRL0, LOCK_CS, DISABLED)
                   BF_GPMI_CTRL0_CS (2)                      // must correspond to NAND CS used
                   BV_FLD(GPMI_CTRL0, ADDRESS, NAND_DATA)
                   BF_GPMI_CTRL0_ADDRESS_INCREMENT (0)
                   BF_GPMI_CTRL0_XFER_COUNT (0);

read[5].gpmi_compare = NULL;                                   // field not used but necessary to set eccctrl
read[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block

//-----
// Descriptor 7: deassert nand lock
//-----
read[6].dma_nextcmdar = NULL;                                  // not used since this is last descriptor

read[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT (0)              // no dma transfer
                 BF_APBH_CHn_CMD_CMDWORDS (0)                // no words sent to GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD (1)              // wait for command to finish before continuing
                 BF_APBH_CHn_CMD_SEMAPHORE (0)
                 BF_APBH_CHn_CMD_NANDWAIT4READY(0)
                 BF_APBH_CHn_CMD_NANDLOCK (0)                 // relinquish nand lock
                 BF_APBH_CHn_CMD_IRQONCMPLT (0)               // ECC8 engine generates interrupt
                 BF_APBH_CHn_CMD_CHAIN (0)                    // terminate DMA chain processing
                 BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);   // no dma transfer

read[6].dma_bar = NULL;                                       // field not used

```

14.2.3.2. Using the Decoder

As illustrated in [Figure 50](#) and the sample code in [Section 14.2.3.1](#):

- DMA descriptor 1 prepares the NAND for data read by using the GPML to issue a NAND read setup command byte under “CLE”, then sends a 5-byte address under “ALE”. The ECC8 engine is not used for these commands.
- DMA descriptor 2 issues a one-byte read execute command to the NAND device that triggers its read access. The NAND then goes not ready.
- DMA descriptor 3 performs a wait for ready operation allowing the DMA chain to remain dormant until the NAND device completes its read access time.
- DMA descriptor 5 handles the reading and error correction of the NAND data. This command’s PIOs activate the ECC8 engine to write the read NAND data to system memory and to process it for any errors that need to be corrected. This DMA descriptor contains two PIO values that are system memory addresses pointing to the PAYLOAD data area and to the AUXILIARY data area. These addresses are used by the ECC8 engine’s AHB master to move data into system memory and to correct it. While this example is reading an entire 4K page—payload plus metadata—it is equally possible to read just one 512-byte payload block or just the uniquely protected metadata block in a single 7 DMA structure transfer.
- DMA descriptor 6 disables the ECC8 engine with the NANDLOCK asserted. This is necessary to ensure that the GPML resource is not arbitrated to another DMA channel when multiple DMA channels are active concurrently.
- DMA descriptor 7 deasserts the NANDLOCK to free up the GPML resource to another channel.

As the ECC8 block receives data from the GPML:

- The decoder transforms the read NAND data block into an RS code word and computes the codeword syndrome.
- If no errors are present, then the ECC8 block can immediately report back to firmware. This report is passed as the HW_ECC8_CTRL_COMPLETE_IRQ interrupt status bit and the associated status registers in HW_ECC8_STATUS0/1 registers.
- If an error is present, then the ECC8 block corrects the necessary data block or parity block bytes, if possible (not all errors are correctable).

As the RS decoder reads the data block and the 9-byte or 18-byte parity block, it records a special condition, i.e., that all of the bits of a payload data block or metadata block are 1, including any associated parity bytes. The “all-ones” case for both parity and data indicates an erased block in the NAND device.

The HW_ECC8_STATUS0 register contains a 4-bit field that indicates the final status of the auxiliary block. HW_ECC8_STATUS1 contains a similar 4-bit field for each of the 512-byte payload data blocks.

- A value of 0x0 indicates no errors found for a block.
- A value of 1 to 8 inclusive indicates that many correctable errors were found and fixed.
- A value of 0xC is reported for any block that was not actually transferred during a specific transaction, i.e., its BLOCK_MASK bit was a 0 for the transaction.
- A value of 0xE indicates uncorrectable errors detected on the block.
- A value of 0xF indicates that the block was in the special ALL ONES state and is therefore considered to be an ERASED block.

- All other values are disallowed by the hardware design.

Recall that up to four NAND devices can have DMA chains in-flight at once, i.e., they can all be contending for access to the GPPI data bus. It is impossible to predict which NAND device will enter the ECC8 engine with a transfer first, because each chain includes a wait4ready command structure. As a result, firmware should look at the HW_ECC8_STATUS0_COMPLETED_CE bit field to determine which block is being reported in the status register. There is also a 16 bit HANDLE field in the HW_GPPI_ECCCTRL register that is passed down the pipeline with each transaction. This handle field can be used to speed firmware's detection of which transaction is being reported.

These examples of reading and writing have focused on full page transfers of 4K page NAND devices. Set HW_GPPI_ECCCTRL_ECC_CMD to a value of HW_GPPI_ECCCTRL_ECC_CMD_ENCODE_4_BIT or to a value of HW_GPPI_ECCCTRL_ECC_CMD_DECODE_4_BIT to enable encode and decode of up to full page transfers of 2K page NAND devices.

To reiterate, you can select a single block to transfer within one transaction by setting only one bit in the HW_GPPI_ECCCTRL_BUFFER_MASK bit field. There is a 1:1 correspondence between a bit in this bit field and a 512-byte buffer address offset into the payload area pointed to by the HW_GPPI_PAYLOAD register. The ECC8 and GPPI blocks are designed to be very efficient at reading single 512-byte pages in one transaction. With no errors, the transaction takes less than 20 HCLKs longer than the time to read the raw data from the NAND.

Additionally, you can select multiple contiguous blocks to transfer within one transaction by setting the respective bits in the HW_GPPI_BUFFER_MASK bit fields. The selected bits must represent a contiguous block of data in the NAND.

To summarize, the APBH DMA command chain for a Reed-Solomon decode operation is shown in [Figure 50](#). Seven DMA command structures must be present for each NAND read transaction decoded by the ECC8. The seven DMA command structures for multiple NAND read transaction blocks can be chained together to make larger units of work for the ECC8, and each will produce an appropriate error report in the ECC8 PIO space. Multiple NAND devices can have such multiple chains scheduled. The results can come back out of order with respect to the multiple chains.

The RS decoder processes the code word in four phases. All phases may not be necessary, for example when no errors are found or when uncorrectable errors are detected.

The four phases include the following:

1. **Syndrome Calculation Phase (SC)**—This is the process of reading in all of the symbols of the block and continuously dividing the code word by the generator polynomial that is a function of the number of symbols to be corrected. The remainder of this division is the syndrome polynomial. If the remainder is 0, i.e., the syndrome symbols are all 0, then the RS code word is correct and no bit errors were detected in the read NAND data, and an ECC8 interrupt is generated upon completion of this phase. Otherwise, we proceed to the next phase. This phase takes place completely on the GPPI clock domain and is fully overlapped with NAND reads from the NAND device. There are approximately two gppl_clk cycles that are not overlapped. The data is passed to the HCLK domain and there are approximately 20 HCLK cycles that are not overlapped on the final block transferred.

2. **Key Equation Solver Phase (KES)**—After the eight (or sixteen) symbol syndromes have been calculated, a set of eight (or sixteen) linear equations with eight (or sixteen) unknowns is formed. The process of solving these equations and selecting from the numerous possible solutions constitutes the KES phase. The hardware block uses the Berlekamp-Massey algorithm to solve the key equations from the syndrome symbols. The resulting λ and Ω polynomials are used in the next phase to determine symbol error locations and the respective correction mask. If the hardware detects an uncorrectable scenario while computing the λ and Ω polynomials, it will terminate and report the appropriate status. This phase takes up to 288 GPMI clocks and 20 hclks, with no planned DMA wait states added.
3. **Chien Search and Forney Evaluator Phase (EVAL)**—This phase takes the λ and Ω polynomials from the KES phase and uses Chien's algorithm for finding the locations of the errors based on the λ polynomial. The method basically involves substituting all 511 nine-bit symbols into the λ polynomial. All non-zero results of these substitutions represent the locations of the various symbol errors. At this point, another calculation involving the λ and Ω polynomials determines the error value or the correction to apply at the symbol in the error locations. This phase consumes approximately 550 gpml_clk and no HCLKs, with no planned DMA wait states added.
4. **Error Correction Phase (CORR)**—The CORR phase applies any required read-modify-write cycles to the data payload and/or auxiliary payload to correct any correctable errors. An ECC8 interrupt is generated upon completion of this phase. Firmware examines the error status registers *and then* clears the interrupt status bit (order is important).

If uncorrectable errors occur, it is up to software to determine how to deal with the 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 possible on a second pass.

14.2.4. Interrupts

There are two interrupt sources used in processing ECC8 protected NAND read and write transfers. Since all ECC8 operations are initiated by GPMI DMA command structures, the DMA completion interrupt for the GPMI is an important ISR. Both of the flow charts of [Figure 45 on page 429](#) and [Figure 48 on page 436](#) show the GPMI DMA complete ISR skeleton. In both reads and writes, the GPMI DMA completion interrupt is used to schedule work *INTO* the error correction pipeline. As the front end processing completes, the DMA interrupt is generated and additional work, i.e., DMA chains, are passed to the GPMI DMA to keep it “fed”. For write operations, this is the only interrupt that will be generated for processing the NAND write transfer.

For reads, however, two interrupts are needed. Every read is started by a GPMI DMA command chain and the front end queue is fed as described above. The back end of the read pipeline is “drained” by monitoring the ECC8 completion interrupt found in `HW_ECC8_CTRL_COMPLETE_IRQ`.

An ECC8 transaction consists of reading or writing all of the blocks requested in the `HW_GPMI_ECCCTRL_BUFFER_MASK` bit field. As every read transaction completes, it posts the status of all of the blocks to the `HW_ECC8_STATUS0` and `HW_ECC8_STATUS1` registers and sets the completion interrupt. The five stages of the ECC8 read pipeline completes, one in the GPMI and four in the ECC8, are independently stalled as they complete and try to deliver to the next stage in the data

flow. Several of these stages can be skipped if no-errors are found or once an uncorrectable error is found in a block.

In any case, the final stage will stall if the status register is busy waiting for the CPU to take status register results. The hardware monitors the state of the HW_ECC8_CTRL_COMPLETE_IRQ bit. If it is still set when the last pipeline stage is ready to post data, then the stage will stall. It follows that the next previous stage will stall when it is ready to hand off work to the final stage, and so on up the pipeline.

WARNING: It is important that firmware read the STATUS0/1 results and save them before clearing the interrupt request bit otherwise a transaction and its results could be completely lost.

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 34.4.10, "Correct Way to Soft Reset a Block" on page 1013](#) 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 ECC8 hardware 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_ECC8_CTRL	0x80008000
HW_ECC8_CTRL_SET	0x80008004
HW_ECC8_CTRL_CLR	0x80008008
HW_ECC8_CTRL_TOG	0x8000800C

Table 614. HW_ECC8_CTRL

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
SFTRST	CLKGATE	AHBM_SFTRST	RSVD	THROTTLE				RSVD														DEBUG_STALL_IRQ_EN	DEBUG_WRITE_IRQ_EN	COMPLETE_IRQ_EN	RSVD				BM_ERROR_IRQ	DEBUG_STALL_IRQ	DEBUG_WRITE_IRQ	COMPLETE_IRQ

Table 615. HW_ECC8_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	0 = Normal ECC8 operation. 1 = Disable clocking with the ECC8 and hold it in its reset (lowest power) state (default). This bit can be turned on and then off to reset the ECC8 block to its default state. This bit resets all state machines except for the AHB master state machine. RUN = 0x0 Allow ECC8 to operate normally. RESET = 0x1 Hold ECC8 in reset.
30	CLKGATE	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block. RUN = 0x0 Allow ECC8 to operate normally. NO_CLKS = 0x1 Do not clock ECC8 gates in order to minimize power consumption.
29	AHBM_SFTRST	RW	0x1	Resets the AHB state machine. 0 = Normal ECC8 operation. 1 = Disable clocking with the ECC8 and hold it in its reset (lowest power) state (default). This bit can be turned on and then off to reset the ECC8 block to its default state. Do not use this bit for normal device soft-resets unless instructed to do so by SigmaTel. RUN = 0x0 Allow ECC8 to operate normally. RESET = 0x1 Hold ECC8 in reset.
28	RSVD	RO	0x0	Reserved.
27:24	THROTTLE	RW	0x0	Non-zero values will hold off that number of HCLKs between success burst requests on the AHB.
23:11	RSVD	RO	0x0	Reserved.
10	DEBUG_STALL_IRQ_EN	RW	0x0	1 = Interrupt on debug stall mode is enabled. The IRQ is raised on every block
9	DEBUG_WRITE_IRQ_EN	RW	0x0	1 = Interrupt on debug write mode is enabled. The IRQ is raised on every transfer. In this mode, no correction occurs.
8	COMPLETE_IRQ_EN	RW	0x0	1 = Interrupt on completion of correction is enabled.
7:4	RSVD	RO	0x0	Reserved.
3	BM_ERROR_IRQ	RW	0x0	AHB Bus Interface Error Interrupt Status. Write a 1 to the SCT clear address to clear the interrupt status bit.
2	DEBUG_STALL_IRQ	RW	0x0	Debug Stall Interrupt Status. Write a 1 to the SCT clear address to clear the interrupt status bit.
1	DEBUG_WRITE_IRQ	RW	0x0	Debug Write Interrupt Status. Write a 1 to the SCT clear address to clear the interrupt status bit.
0	COMPLETE_IRQ	RW	0x0	External Interrupt Line Status. Write a 1 to the SCT clear address to clear the interrupt status bit. Note: Subsequent ECC completions will be held off as long as this bit is set. Be sure to read the data from HW_ECC8_STATUS0/1 before clearing this interrupt bit.

14.4.2. Hardware ECC Accelerator Status Register 0 Description

The Hardware ECC Accelerator Status Register 0 provides overall status of the hardware ECC accelerator.

HW_ECC8_STATUS0

0x80008010

Table 616. HW_ECC8_STATUS0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
HANDLE																RS8ECC_ENC_PRESENT	RS8ECC_DEC_PRESENT	RS4ECC_ENC_PRESENT	RS4ECC_DEC_PRESENT	STATUS_AUX				RSVD		ALLONES	CORRECTED	UNCORRECTABLE	COMPLETED_CE		

Table 617. HW_ECC8_STATUS0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	HANDLE	RO	0x0	Software supplies a 16-bit handle for this transfer as part of the GPPI DMA PIO operation that started the transaction. That handle passes down the pipeline and ends up here at the time the ECC8 interrupt is signaled.
15	RS8ECC_ENC_PRESENT	RO	0x1	Reserved.
14	RS8ECC_DEC_PRESENT	RO	0x1	Reserved.
13	RS4ECC_ENC_PRESENT	RO	0x1	Reserved.
12	RS4ECC_DEC_PRESENT	RO	0x1	Reserved.
11:8	STATUS_AUX	RO	0xc	Count of symbols in error during processing of auxiliary data area. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.
7:5	RSVD	RO	0x0	Reserved.
4	ALLONES	RO	0x1	1 = All data bits of this transaction are 1.
3	CORRECTED	RO	0x0	1 = At least one correctable error encountered during last processing cycle.
2	UNCORRECTABLE	RO	0x0	1 = Uncorrectable error encountered during last processing cycle.
1:0	COMPLETED_CE	RO	0x0	Chip enable number, 0 to 3, corresponding to the NAND device from which this data came.

STMP3770

DESCRIPTION:

The Hardware ECC Accelerator Status Register 0 provides visibility into the run-time status of the ECC8. The register also reflects the ECC8 configurations supported in this version of the STMP3770.

EXAMPLE:

```
Have8BitRSEncode = HW_ECC8_STAT.B.RS8ENC_PRESENT;
```

14.4.3. Hardware ECC Accelerator Status Register 1 Description

The Hardware ECC Accelerator Status Register 1 provides overall status of the hardware ECC accelerator.

HW_ECC8_STATUS1

0x80008020

Table 618. HW_ECC8_STATUS1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
STATUS_PAYLOAD7				STATUS_PAYLOAD6				STATUS_PAYLOAD5				STATUS_PAYLOAD4				STATUS_PAYLOAD3				STATUS_PAYLOAD2				STATUS_PAYLOAD1				STATUS_PAYLOAD0			

Table 619. HW_ECC8_STATUS1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	STATUS_PAYLOAD7	RO	0xc	Count of symbols in error during processing of payload area 7. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.
27:24	STATUS_PAYLOAD6	RO	0xc	Count of symbols in error during processing of payload area 6. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.

Table 619. HW_ECC8_STATUS1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:20	STATUS_PAYLOAD5	RO	0xc	Count of symbols in error during processing of payload area 5. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.
19:16	STATUS_PAYLOAD4	RO	0xc	Count of symbols in error during processing of payload area 4. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.
15:12	STATUS_PAYLOAD3	RO	0xc	Count of symbols in error during processing of payload area 3. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.
11:8	STATUS_PAYLOAD2	RO	0xc	Count of symbols in error during processing of payload area 2. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.

Table 619. HW_ECC8_STATUS1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:4	STATUS_PAYLOAD1	RO	0xc	Count of symbols in error during processing of payload area 1. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.
3:0	STATUS_PAYLOAD0	RO	0xc	Count of symbols in error during processing of payload area 0. 0xF indicates uncorrectable. NO_ERRORS = 0x0 No errors occurred. ONE_CORRECTABLE = 0x1 One correctable error occurred. TWO_CORRECTABLE = 0x2 Two correctable errors occurred. THREE_CORRECTABLE = 0x3 Three correctable errors occurred. FOUR_CORRECTABLE = 0x4 Four correctable errors occurred. FIVE_CORRECTABLE = 0x5 Five correctable errors occurred. SIX_CORRECTABLE = 0x6 Six correctable errors occurred. SEVEN_CORRECTABLE = 0x7 Seven correctable errors occurred. EIGHT_CORRECTABLE = 0x8 Eight correctable errors occurred. NOT_CHECKED = 0xC This block was not examined by the ECC8. UNCORRECTABLE = 0xE Errors occurred that were uncorrectable. ALL_ONES = 0xF All bits are 1.

DESCRIPTION:

The Hardware ECC Accelerator Status Register 1 provides visibility into the run-time status of the ECC8. The register also reflects the ECC8 configurations supported in this version of the STMP3770.

EXAMPLE:

```
if (HW_ECC8_STAT1.B._PAYLOAD0) LifeIsGood();
```

14.4.4. Hardware ECC Accelerator Debug Register 0

The ECC8 internal state machines and signals can be seen in the Hardware ECC Accelerator Debug Register 0.

HW_ECC8_DEBUG0	0x80008030
HW_ECC8_DEBUG0_SET	0x80008034
HW_ECC8_DEBUG0_CLR	0x80008038
HW_ECC8_DEBUG0_TOG	0x8000803C

STMP3770

Table 621. HW_ECC8_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12	KES_DEBUG_KICK	RW	0x0	Toggling causes KES engine FSM to start as if kicked by the bus master. This allows standalone testing of the KES and Chien Search engines. Be sure to set KES_ECC8_DEBUG0_KES_STANDALONE mode to 1 before kicking.
11	KES_STANDALONE	RW	0x0	Set to 1 to cause the KES engine to suppress toggling the KES_BM_DONE signal to the bus master and to suppress toggling the CF_BM_DONE signal by the CF engine. NORMAL = 0x0 Bus master address generator for synd_gen writes operates normally. TEST_MODE = 0x1 Bus master address generator always addresses last four bytes in auxiliary block.
10	KES_DEBUG_STEP	RW	0x0	Toggling this bit causes the KES FSM to skip past the stall state if it is in DEBUG_STALL mode and it has completed processing a block.
9	KES_DEBUG_STALL	RW	0x0	Set to 1 to cause KES FSM to stall after notifying the Chien search engine to start processing its block but before notifying the bus master that the KES computation is complete. This allows a diagnostic to stall the FSM after each block's key equations are solved. This also has the effect of stalling the CSFE search engine so its state can be examined after it finishes processing the KES stalled block. NORMAL = 0x0 KES FSM proceeds to next block supplied by bus master. WAIT = 0x1 KES FSM waits after current equations are solved and the search engine is started.
8	BM_KES_TEST_BYPASS	RW	0x0	1 = Point all synd_gen writes to dummy area at the end of the auxiliary block so that diagnostics can preload all payload, parity bytes, and computed syndrome bytes for test the KES engine. NORMAL = 0x0 Bus master address generator for synd_gen writes operates normally. TEST_MODE = 0x1 Bus master address generator always addresses last four bytes in auxiliary block.
7:6	RSVD	RO	0x0	Reserved.
5:0	DEBUG_REG_SELECT	RW	0x0	The value loaded in this bit field is used to select the internal register state view of KES engine or the Chien search engine.

DESCRIPTION:

The HW_ECC8_DEBUG0 register provides access to various internal state information which might prove useful during hardware debug and validation.

EXAMPLE:

```
Value = HW_ECC8_DEBUG0.U; // diagnostic programs can read and act upon various bit fields.
```

BITS	LABEL	RW	RESET	DEFINITION
31:0	VALUES	RO	0x0	Reserved.

STMP3770



15. DATA CO-PROCESSOR (DCP)

This chapter describes the data co-processor (DCP) block included on the STMP3770 and how to use it. It includes sections on memory copy functionality, Advanced Encryption Standard (AES), hashing, color-space conversion, and arbitration. Sections on programming channel operations and the color-space converter are also provided, along with example code. Programmable registers are described in [Section 15.3](#).

15.1. Overview

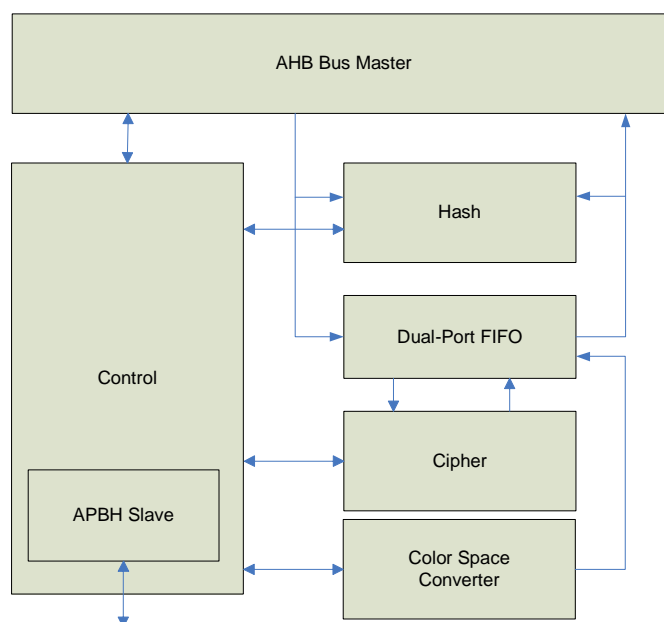


Figure 51. Data Co-Processor (DCP) Block Diagram

The DCP module provides support for general encryption and hashing functions typically used for security functions. Because its basic job is moving data from memory to memory, it also incorporates memory-copy (memcpy) function for both debugging and as a more efficient method of copying data between memory blocks than the DMA-based approach. The memcpy function also has a “blit” mode of operation where it can transfer a rectangular block of data to a video frame buffer. Finally, the DCP module incorporates a color-space conversion (CSC) block to assist software in performing video decode.

The DCP has been designed to support a wide variety of encryption and hashing algorithms, and the control structures have been designed to allow flexibility in adding additional algorithms and modes in the future. It supports up to 16 encryption algorithms (for example DES, TDEA, AES-128, etc.) with 16 different modes of operation (ECB, CBC, etc.) as well as 16 hashing algorithms (for example MD5, SHA-1, SHA-2, etc.). While the DCP module has been designed to support numerous algorithms, only a subset may be implemented in any given implementation (see the Capability Register).

The DCP module processes data based on chained command structures written to system memory by software (in a manner similar to the DMA engine). The control

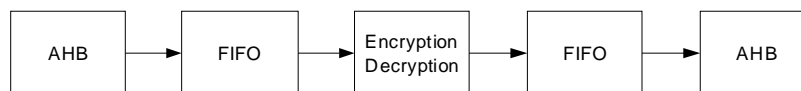
block maintains registers to support four independent and concurrent chains, allowing software to virtualize access to the DCP block. Each command in a chain represents a work unit that the module will process to completion. At the end of each work unit, the control logic arbitrates among chains with outstanding commands and processes a command from that chain. Arbitration among the channels is round-robin, allowing all active channels equal access to the data engine. Each channel also supports a “high-priority” mode that allows it to have priority over the remaining channels. If multiple channels are selected as high-priority, the channel arbiter selects among the high-priority channels in round-robin fashion.

The data flow through the DCP module can be configured in one of five fashions, depending on the functionality activated by the control packet:

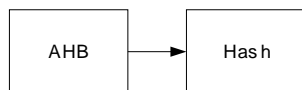
- **Memcopy/Blit Mode**—Data is moved unchanged from one memory buffer to another.



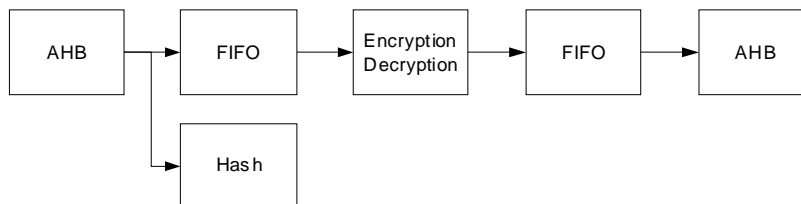
- **Encryption Only**—Data from source buffer is encrypted/decrypted into the destination buffer.



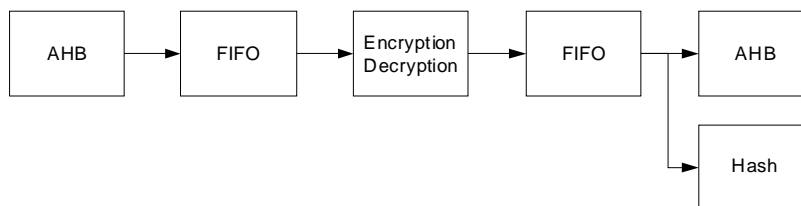
- **Hashing Only**—Data from source buffer is read, and a hash is generated.



- **Encryption and Input Hashing**—Data from source buffer is encrypted/decrypted into destination, and source buffer is hashed.



- **Encryption and Output Hashing**—Data from source buffer is encrypted/decrypted into destination, and output data is hashed.



15.1.1. DCP Limitations for Software

While the DCP module has been designed to be as flexible as possible, there are a few limitations to which software must adhere:

- Buffer sizes for all operations **MUST** be aligned to the natural size of the transfer algorithm used. Memcopy operations can transfer any number of bytes (one-byte granularity) and AES operations must be multiples of 16 bytes (four-word granularity). For all operations, if the byte count is not a word granularity, the hardware rounds up to the next word. Hashing is supported at a byte granularity.
- The DCP module supports buffer operations to any byte alignment, but performance will be improved if buffers are aligned to a four-byte boundary, since fetch/store operations can be performed without having to do byte operations to accommodate the misaligned addresses.
- Hash operations are limited to a 512-Mbyte buffer size. The hardware only implements a 32-bit hash length counter instead of the 64-bit counter supported by the SHA-1 algorithm (counter counts bits, not bytes, therefore a total of 512 Mbytes).
- For chained hashing operations (operations involving multiple descriptors), every descriptor except the last must have a byte count that is a 16-word multiple (granularity of the hash algorithm).
- Key values cannot be written while the AES block is active. This limitation exists because the key RAM is in use while AES is operational. Any writes from the APB cannot be held in wait states; therefore, the RAM must be accessible during key writes.
- The byte-swap controls can only be used with modulo-4 length buffers. For non-modulo-4 lengths, the final partial word will contain incorrect data. Any address alignment can be used with byte swapping, however.
- The word-swap controls are only useful with cipher operations, because the logic forms the 128-bit cipher data from four words from system memory. The word-swap controls are ignored for memcopy or hashing operations.

15.2. Operation

The top-level DCP module contains the AHB master, APB slave bus interface units, the main control block and FIFO, and any encryption or hashing blocks included with the design.

The controller manages the fetching of work blocks, the fetching/storing of context information when switching between chain pointers, and the managing of the data flow through the FIFO, SHA, and AES blocks. Data entering the block from the AHB master is placed in the FIFO for consumption by the cipher block. After the cipher module has finished its operation, data is placed back into the FIFO and stored back to memory via the AHB master. When hashing is enabled, the SHA block takes its inputs from the bus side of the FIFO to allow it to operate without waiting for the cipher block to complete. The APB slave provides all register controls and interfaces mainly with the control block.

15.2.1. Memory Copy, Blit, and Fill Functionality

In its most basic operation, the DCP supports moving unmodified data from one place in system memory to another. This functionality is referred to as “memcopy”, because it operates only on memory and it copies data from one place to another. Typical uses for memcopy might be for fast virtual memory page moves or repositioning data blocks in memory. Memcopy buffers can be aligned to any

memory address and can be of any length (byte granularity). For best performance, buffers should be word-aligned, although the DCP includes enhancements to improve performance for unaligned cases.

The DCP also has the ability to do basic “blit” operations that are typical in graphics operations. To specify a blit, the control packet must have the `ENABLE_BLIT` bit set in the packet control register. Blit source buffers must be contiguous. The output destination buffer for a blit operation is defined as a “M runs of N bytes” that define a rectangular region in a frame buffer. For blit operations, each line of the blit may consist of any number of bytes. After performing a “run”, the DCP updates the destination pointer such that the next destination address falls on the pixel below the start of the previous run operation. This is done by incrementing the starting pointer by the frame buffer width, which is specified in the `Control1` field.

In addition to being able to copy data within memory, the DCP also provides a “fill” operation, where source data comes not from another memory location, but from an internal register (the source buffer address in the control packet). This is done whenever the `CONSTANT_FILL` flag is set in the packet control register. This feature may be used with memcopy to prefill memory with a specified value or during a blit operation to fill a rectangular region with a constant color.

15.2.2. Advanced Encryption Standard (AES)

The AES block implements a 128-bit key/data encryption/decryption block as defined by the National Institute of Standards and Technology (NIST) as US FIPS PUB 197, dated November 2001 (see references for specifications and toolkits)¹.

There are three variations of AES, each corresponding to the key size used: AES-128, AES-192 or AES-256. AES always operates on 128 bits of data at a time. Only the AES-128 algorithm is implemented at this time.

15.2.2.1. Key Storage

The DCP implements four SRAM-based keys that may be used by software to securely store keys on a semi-permanent basis. The keys may be written via the PIO interface by specifying a key index to specify which key to load and a subword pointer that indicates which word to write within the key. After a subword is written, the logic automatically increments the subword pointer so that software can program the higher-order words of the key without rewriting the key index. Keys written into the key storage are not readable.

To use a key in the key storage, the cipher descriptor packet should select the key by setting the `KEY_SELECT` field in the `Control1` descriptor field without setting the `OTP_KEY` or `PAYLOAD_KEY` fields in the `Control0` register.

15.2.2.2. OTP Key

After a system reset, the OTP controller reads the e-fuse devices and provides OTP key information over a parallel 128-bit interface. The key transfer interface runs on `HCLK` and provides the key over the serialized `otp_data` signal. The `otp_crypto_key_smpl` signal indicates when the key value is valid and causes the control logic to capture the key into the key RAM.

1. The AES core used in the design was derived from the AES design available from OpenCores.org under a modified BSD license as described here: http://www.opencores.org/projects.cgi/web/aes_core/overview
The license for this code is documented on page 523 for compliance.

To use the OTP key, the descriptor packet should set the OTP_KEY field in the Control1 register.

15.2.2.3. Encryption Modes

The most basic form of encryption is the Electronic Code Book (ECB) mode. In this mode, the encryption output is a function only of the key and the plaintext, thus it can be visualized as a giant lookup table. While this provides a great deal of security, there are a few limitations. For instance, if the same plaintext appears more than once in a block of data, the same ciphertext will also appear. This can be very evident if the plaintext contains large blocks of constant data (0s for example) and can be used to formulate attacks against a key.

In order to make ciphers stronger, several modes of operation can be implemented around the basic ECB cipher to provide additional security. One such mode is CBC mode (Cipher Block Chaining), which takes the previous encrypted data and logically XORs it with the next incoming plaintext before performing the encryption. During decryption, the process is reversed and the previous encrypted data is XORed with the decrypted ECB data to provide the plaintext again.

The AES engine supports handling the various modes of operation. The core AES block supports ECB mode, and other algorithms are handled in the wrapper around the encryption blocks. The DCP module supports Cipher Block Chaining (CBC), which chains the data blocks by XORing the previously encrypted data with the plaintext before encryption. Cipher block chaining encryption is illustrated in [Figure 52](#).

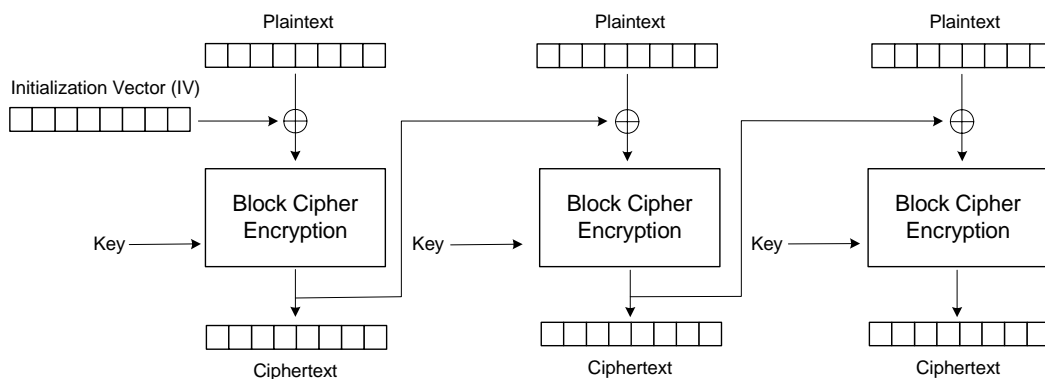


Figure 52. Cipher Block Chaining (CBC) Mode Encryption

Decryption (shown in [Figure 53](#)) works in a similar manner, where the cipher text is first decrypted and then XORed with the previous ciphertext. For the first encryption/decryption operation, an initialization vector (IV) is used to seed the operation. The IV must be the same for both the encryption and decryption steps; otherwise, decrypted data will not match the original plaintext.

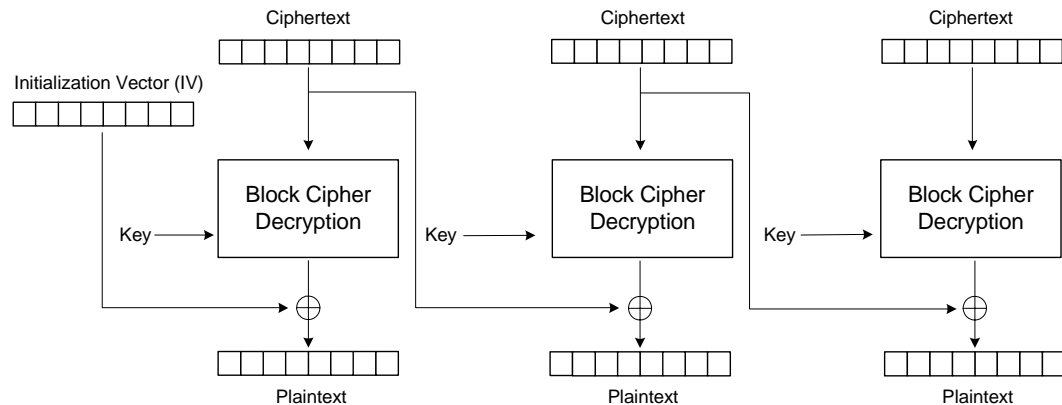


Figure 53. Cipher Block Chaining (CBC) Mode Decryption

15.2.3. Hashing

The hashing module implements the SHA-1 hashing algorithm and a modified CRC-32 checksum algorithm. These algorithms produce a signature for a block of data that can be used to determine whether the data is intact.

The CRC-32 algorithm implements a 32-bit CRC algorithm similar to the one used by Ethernet and many other protocols. The CRC differs from the Unix `cksum()` function in three ways:

- The CRC is initialized as 0xFFFFFFFF instead of 0x00000000.
- Logic pads zeros to a 32-bit boundary for trailing bytes.
- Logic does not post-pend the file length.

The SHA-1 block implements a 160-bit hashing algorithm that operates on 512-bit (64-byte) blocks as defined by US FIPS PUB 180-1 in 1995. The purpose of the hashing module is to generate a unique signature for a block of data that can be used to validate the integrity of the data by comparing the resulting “digest” with the original digest.

Results from hash operations are written to the beginning of the payload for the descriptor. The DCP also has the ability to check the resulting hash against a value in the payload and issue an interrupt if a mismatch occurs.

15.2.4. Color-Space Conversion (YUV/YCbCr to RGB)

The DCP module supports color-space conversion to assist software in video processing. The function of the color-space converter is to reformat the YUV-formatted video buffer created by the video decode algorithm into the RGB color format required for display on an LCD.

- The color-space converter accepts YUV or YCbCr data in planar 4:2:0 or 4:2:2 format. The output of the CSC is either a 16-bit RGB frame buffer (565 format), a 24-bit unpacked frame buffer (24-bit pixels stored as a 32-bit word), or an interleaved YCbCr 4:2:2 format to support LCD displays that accept YCbCr data.
- In addition, the logic supports 8-bit delta pixel displays by simple reordering of RGB components on odd scan lines (selectable via mode bits).

Since the CSC may have a high latency when processing a frame buffer, it supports suspending its current operation to allow the four data channels to perform an

operation. The control logic supports priority levels such that the CSC can be forced to suspend on either outstanding high-priority requests or any requests from the other channels.

The color-space converter is not a generic matrix-multiply, but instead implements the subset of equations necessary to perform the conversion process. The CSC also does not work on chained command structures, but is register-driven because the buffer processing latency requirement is greater than 5 ms and video buffers are typically multi-buffered.

Source data for the CSC will generally be planar YUV/YCbCr data, where associated Y/U/V values come from independent buffers in system memory. Typically, the buffers are contiguous, with chrominance (Cb and Cr) data following the luminosity (Y) data. The CSC includes registers with information on the frame buffer, such as the number of pixels per line and the number of lines per field.

The CSC supports 4:2:0 or 4:2:2 color subsampling as a baseline. [Figure 54](#) illustrates the supported chroma subsampling modes.

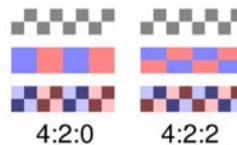


Figure 54. Supported Chroma Subsampling Modes

Table 634. 4:2:2 Interleaved YUV Output Buffer Format

REPRESENTATION	FORMAT
4:2:2	[Y0 U0] [Y1 V1] [Y2 U2] [Y3 U3]
Pixels	[Y0 U0 V1] [Y1 U0 V1] [Y2 U2 V3] [Y3 U2 V3]
Byte storage in memory	0x00: U0 0x01: Y0 0x02: V1 0x03: Y1

As shown in [Table 635](#), the CSC supports two RGB frame buffer formats: a packed 16-bit value with 5 bits of red and blue and 6 bits of green, or a 24-bit frame buffer unpacked in a 32-bit word. The logic supports both 24-bit striped and delta configuration LCDs by shifting the RGB components on odd scan lines to match the color stream expected by the LCD.

Table 635. RGB Frame Buffer Formats

CONFIGURATION	FORMAT
565 RGB	{5{red}, 6{green}, 5{blue}}
888 RGB (striped)	{8'h00, 8{red}, 8{green}, 8{blue}}
888 RGB (delta)	even lines: {8'h00, 8{red}, 8{green}, 8{blue}}, ... odd lines: {8'h00, 8{green}, 8{blue}, 8{red}}...

The following equations are used to perform the YUV-to-RGB conversion:

$$R = C_0(Y - Y_{\text{OFFSET}}) + C_1(Cr - 128)$$

$$G = C_0(Y - Y_{\text{OFFSET}}) - C_2(Cr - 128) - C_3(Cb - 128)$$

$$B = C_0(Y - Y_{\text{OFFSET}}) + C_4(Cb - 128)$$

The constants shown in [Table 636](#) are stored in registers to allow flexibility in the implementation and to allow for differences in YUV and YCbCr computations. In addition, this allows software some flexibility in modifying constant values to increase or decrease brightness or contrast.

Table 636. Constants for Color-Space Conversion

CONVERSION TYPE	Y OFFSET	COEFFICIENT	DEC	HEX ¹
YCbCr-to-RGB, where saturation is: Y: 16–240 R/G/B: 0–255	16	C ₀	1.164	0x12A
		C ₁	1.596	0x198
		C ₂	0.813	0x0D0
		C ₃	0.392	0x064
		C ₄	2.017	0x204
YCbCr-to-RGB, where saturation is: Y: 16–240 R/G/B: 16–235	0	C ₀	1	0x100
		C ₁	1.336	0x156
		C ₂	0.700	0x0B3
		C ₃	0.334	0x055
		C ₄	1.732	0x1BB
YUV-to-RGB, where saturation is: Y: 0–255 R/G/B: 0–255	0	C ₀	1	0x100
		C ₁	1.140	0x123
		C ₂	0.395	0x065
		C ₃	0.581	0x094
		C ₄	2.032	0x208

Notes:

1. Hexadecimal values for the coefficients can be calculated as:
 $256 * (\text{decimal value})$ converted into hexadecimal

15.2.5. Managing DCP Channel/CSC Arbitration and Performance

The DCP can have four channels and the CSC block all competition for DCP resources to complete their operations. Depending on the situation, critical operations may need to be prioritized above less important operations to ensure smooth system operation. To help software achieve this goal, the DCP implements a programmable arbiter for internal DCP operations and provides “recovery” timers on each channel to throttle channel activity.

15.2.5.1. DCP Arbitration

The DCP implements a multi-tiered arbitration policy that allows software a lot of flexibility in scheduling DCP operations. Figure 55 illustrates the arbitration levels and where each channel and the CSC fit into the arbitration scheme.

		Channels				CSC
		0	1	2	3	
Priority Level	High	1	1	1	1	3
	Medium					2
	Low	0	0	0	0	1
	Background					0

Figure 55. DCP Arbitration

Each channel can be programmed as being in either the high-priority or low-priority arbitration pool, depending on the setting in the HIGH_PRIORITY_CHANNEL field of the HW_DCP_CHANNELCTRL register. When the corresponding bit is programmed as a 1, the channel arbitrates in the high-priority pool; otherwise it arbitrates in the low-priority pool. Once a channel has been selected, it completes one packet and then the arbiter re-arbitrates. The channel that just completed participates in the new arbitration round.

The color-space converter can be programmed to operate in any of the four arbitration pools, which is selected by the CSC_PRIORITY field in the HW_DCP_CHANNELCTRL register. The additional granularity allows software to better control how much of the DCP's resources the CSC operations may consume. After the CSC is granted control of the DCP resources, it processes one line of video and then allows arbitration to occur again to ensure fair resource sharing. Each arbitration pool is arbitrated independently in a round-robin fashion. This ensures that the arbiter is perfectly fair in its distribution of resources. For each arbitration cycle, any pending requests in the high-priority pool are serviced first, followed by requests in the medium, low, and background pools.

15.2.5.2. Channel Recovery Timers

Each channel also contains a channel recovery timer in its HW_DCP_CHnOPTS register. The purpose of the recovery timer is to keep the channel inactive for a period of time after it completes an operation. This capability could be used for a high-priority channel to ensure that at least some lower-priority requests get serviced between packets or to simply allow more timeslices for other channels to perform operations. The value programmed into the recovery timers register delays the channel from operations until 16 times the programmed value. Any non-zero value should prevent the channel from participating in the next arbitration cycle.

Note: The CSC function does not support HCLK auto-slow mode.

15.2.6. Programming Channel Operations

The control logic block maintains the channel pointers and manages arbitration and context switching between the different channels. It also manages the fetching of

work packets and data fetch/store operations from the AHB master interface and coordinates the actions of the hashing and encryption blocks.

The control logic maintains four channels that allow software to effectively create four independent work sets for the DCP module. Software can construct chained control packets in memory that describe encryption/hashing/memcopy operations to the hardware unit. The address for this first control packet can be written to one of the four virtual channels and enabled. When one or more of the channels is enabled, the controller fetches the control packet pointed to by that channel and initiate data fetches from the source buffer. Data is then processed as described in the control packet and stored back to system memory.

Once the processing for a control packet is complete, the controller writes completion status information back to the control packet, and optionally stores relevant state information to the context buffer. If the control packet specifies a subsequent control packet, the channel's pointer is updated to the address for the next packet and an optional interrupt can be issued to the processor.

At this point, the DCP module arbitrates among the virtual channels for the next operation and processes its control packet. If a subsequent operation is continued from a previous operation, the controller automatically loads the context from the previous session into the working registers before resuming operation for that channel.

15.2.6.1. Virtual Channels

The DCP module processes data via work packets stored in memory. Each of the channels contains a pointer to the current work packet and enough control logic to determine whether the channel is active and to provide status to the processor. Each channel also provides a recovery timer to help throttle processing by a particular channel. After processing a packet, the channel enables its 16-bit recovery timer (if the recovery time is set to a non-zero value). The channel will not become active again until its recovery timer has expired. The recovery timer timebase is 16 HCLK cycles, so the timer acts as a 20-bit timer with the bottom four bits implicitly tied to 0. This provides an effective range of 0 to 2^{20-1} clocks or 0 ns to 7.8 ms at 133 MHz.

A channel is activated any time its semaphore is non-zero and its recovery timer is cleared. The semaphore can be incremented by software to indicate that the chain pointer has been loaded with a valid pointer. As the hardware completes the work packets, it decrements the semaphore if the Decrement Semaphore flag in the Control 0 field set. Work packets may be chained together using the CHAIN or CHAIN_CONTIGUOUS bits in the Control0 field, which causes the channel to automatically update the work packet pointer to the value in the NEXT_COMMAND_ADDRESS field at the end of the current work packet.

All channels have the same capabilities, but channel 0 is special in that it has a private interrupt line (dcp_vmi_irq). This allows software to use it for VMI (virtual memory) page-copy operations and have a dedicated interrupt vector to reduce latency. All other channels (and the color-space converter) share the other interrupt (dcp_irq).

15.2.6.2. Context Switching

The control logic maintains four virtual channels that allow the DCP block to time-multiplex encryption, hashing, and memcopy operations, it must also retain state information when changing channels so that when a channel is resumed, it can resume the operation from where it left off. This process is called context-switching.

To minimize the number of registers used in the design, the controller saves context information from each channel into a private context area in system memory. When initializing the DCP module, software must allocate memory for the context buffer and write the address into the Context Buffer Pointer register. As the DCP module processes packets, it saves the context information for each channel to the buffer after completing each control packet. When the channel is subsequently activated, the DCP module's internal registers are then reloaded with the proper context before resuming the operation.

Each channel reserves one-fourth the context buffer area for its context storage. The context buffer consumes 160 bytes of system memory and is formatted as shown in [Table 637](#).

Table 637. DCP Context Buffer Layout

RANGE	CHANNEL	DATA	SIZE
0x00–0x0C	3	Cipher Context	16 bytes
0x10–0x24		Hash Context	24 bytes
0x28–0x34	2	Cipher Context	16 bytes
0x38–0x4C		Hash Context	24 bytes
0x50–0x5C	1	Cipher Context	16 bytes
0x60–0x74		Hash Context	24 bytes
0x78–0x84	0	Cipher Context	16 bytes
0x88–0x9C		Hash Context	24 bytes

The control logic writes to the context buffer only if the function is being used. This effectively means that the cipher context is stored for CBC encryption/decryption operations only, and the hash context is written only if SHA-1 is utilized. If neither of these modes are used for a given channel, the memory for the context buffer need not be allocated by software.

Since channel 0 is likely to be used for VMI in an SDRAM-based system-to-page data from the SDRAM to on-chip SRAM, the buffer allocation table has been organized so that the *highest* numbered channel uses the *lowest* area in the context buffer. For this reason, software should allocate the higher numbered channels for encryption/hashing operations and the lower numbered channels for memcopy operations to reduce the size of the context buffer.

If only a single channel is used for CBC mode operations or hashing operations, the controller provides a bit in the control register to disable context switching. In this scenario, context switching is not required, because other channels will not corrupt the state of the hashing or cipher modes. Thus, when the channel resumes, a context load is not required.

Additionally, the control logic monitors the use of CBC/hashing, so that a context reload is not done if the previous channel resumes an operation without an intermediate operation from another channel.

15.2.6.3. Working with Semaphores

Each channel has a semaphore register to indicate that control packets are ready to be processed. Several techniques can be used when programming the semaphores

to control the execution of packets within a channel. The channel will continue to execute packets as long as the semaphore is non-zero, a chain bit is set in the descriptor, and no error has occurred.

- Software can write the number of pending packets into the semaphore register with the Decrement Semaphore bit in each control packet set. In this scenario, the channel simply decrements the semaphore for each packet set and terminates at the end of the packet chain. The benefit of this method is that software can easily determine how many packets have executed by reading the semaphore status register.
- Software can create a packet chain with the Decrement Semaphore bit set only in the last packet. In this case, software would write a 1 to the semaphore register to start the chains and the DCP will terminate after executing the last packet.
- Software can create a packet chain with the Decrement Semaphore bit set for each packet and write either a 1 or a number less than the number of packets to the semaphore register. This can be useful when debugging, because it allows the channel to execute only a portion of the packets and software can inspect intermediate values before restarting the channel again.

If an error occurs, the channel issues an interrupt and clears the semaphore register. The channel does not perform any further operations until the error bits in the status register have been cleared. Software can manually clear a non-zero semaphore by writing 0xFF to the CLR (clear) address of the semaphore register.

15.2.6.4. Work Packet Structure

Work packets for the DCP module are created in memory by the processor. Each work packet includes all information required for the hardware to process the data. The general structure of the packet is shown in [Figure 56](#).

Word 0	Next Cmd Addr
Word 1	Control 0
Word 2	Control 1
Word 3	Source Buffer Addr
Word 4	Destination Buffer Addr
Word 5	Buffer Size
Word 6	Payload Pointer
Word 7	Status

Figure 56. DCP Work Packet Structure

For some operations, additional information is required to process the data in the packet. This additional information is provided in the variable-sized payload buffer, which can be found at the address specified in the payload pointer field. When encryption is specified, the payload may include the encryption key (if the key selected resides in the packet), an initialization vector (for modes of operation such

as CBC), and expected hash values when data hashing is indicated. The hardware can automatically compare the expected hash with the actual hash and interrupt the processor only on a mismatch. The payload area is used by software to store the calculated hash value at the end of hashing operations (when the HASH_TERM control bit is set).

15.2.6.4.1. Next Command Address Field

The NEXT_COMMAND_ADDRESS field (as shown in [Table 638](#)) is used to point to the next work packet in the chain. This field is loaded into the channel's command pointer after the current packet has completed processing if the CHAIN bit in the CONTROL0 field (see [Table 639](#)) is set. The Next Command Address field should be programmed to a non-zero value when the CHAIN bit is set; otherwise, the channel will flag an invalid setup error.

Table 638. DCP Next Command Address Field

31	30	29	28	27	26	25	24	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																															

15.2.6.4.2. Control0 Field

The main functions of the DCP module are enabled with the ENABLE_MEMCOPY, ENABLE_BLIT, ENABLE_CIPHER, and ENABLE_HASH bits from the Control0 field in the work packet. The combinations of these bits determine the function performed by the DCP. [Table 640](#) summarizes the function performed for each combination.

Table 639. DCP Control0 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TAG								OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDWRAP	KEY_BYTEWRAP	TEST_SEMA_IRQ	CONSTANT_FILL	HASH_OUTPUT	HASH_CHECK	HASH_TERM	HASH_INIT	PAYLOAD_KEY	OTP_KEY	CIPHER_INIT	CIPHER_ENCRYPT	ENABLE_BLIT	ENABLE_HASH	ENABLE_CIPHER	ENABLE_MEMCOPY	CHAIN_CONTINUOUS	CHAIN	DECR_SEMAPHORE	INTERRUPT_ENABLE

Table 640. DCP Function Enable Bits

HASH	CIPHER	BLIT	MEMCOPY	DESCRIPTION
0	0	0	X	Simple memcopy operation.
0	0	1	0	Blit operation.
0	1	0	0	Simple encrypt/decrypt operation.
1	0	0	0	Simple hash. Only reads performed.
1	0	0	1	Memcopy and hash operation.
1	1	0	0	Hash with encryption/decryption.
All Others				Invalid setup.

The CHAIN bit should be set if the NEXT_COMMAND_ADDRESS field has a valid pointer to the next work packet. When set, this bit causes the channel to update its pointer to the next packet. The CHAIN_CONTINUOUS bit is similar, but it indicates that the next packet follows immediately after the current packet. This allows software to generate templates of operations without regard to the actual physical addresses used, which makes programming easier, especially in a virtual memory environment.

If the INTERRUPT_ENABLE bit is set, the channel generates an interrupt to the processor after completing the work for this packet. When the interrupt is issued, the packet will have been completely processed, including the update of the status/payload fields in the work packet.

Each channel contains 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. Whenever a command finishes its operation, it checks the DECR_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 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, it then uses the value in the NEXT_COMMAND_ADDRESS field to begin processing.

If the ENABLE_CIPHER bit is set, the data is encrypted or decrypted based on the CIPHER_ENCRYPT bit using the key/encryption settings from the Control1 field. The OTP_KEY and PAYLOAD_KEY indicate the source of the keys for the operation. If the OTP_KEY value is set, the KEY_SELECT field from the Control1 register indicates which OTP key is to be used. If the PAYLOAD_KEY bit is set, the first entry in the payload is the key to be used for the operation. If neither bit is set, the KEY_SELECT field indicates an index in the key RAM that contains the key to be used. (For cases when both the OTP_KEY and PAYLOAD_KEY bits are set, the PAYLOAD_KEY takes precedence.)

The HASH_ENABLE enables hashing of the data with the HASH_OUTPUT bit controlling whether the input data (HASH_OUTPUT=0) or output data (HASH_OUTPUT=1) is hashed. In addition, the hardware can be programmed to automatically check the hashed data if the HASH_CHECK bit is set. If the hash does not match, an interrupt is generated and the channel terminates operation on the current chain. The hashing algorithms also require two additional fields in order to operate properly. The HASH_INIT bit should be set for the first block in a hashing pass to properly initialize the hashing function. The HASH_TERM bit must be set on the final block of a hash to notify the hardware that the hashing operation is complete so that it can properly pad the tail of the buffer according to the hashing algorithm. This flag also triggers the write-back of the hash output to the work packet's payload area.

The CONSTANT_FILL flag may be used for both memcpy and blit operations. When this is set, the source address field is used instead as a constant that is written to all locations in the output buffer.

The WORDSWAP and BYTESWAP bits enable muxes around the FIFO to swap data to handle little-endian and big-endian storage of data in system memory. When these bits are cleared, data is assumed to be in little-endian format. When all bits are set, data is assumed to be in big-endian format.

The TAG field is used to identify the work packet and the associated completion status. As each packet is completed, the channel provides the status and tag information for the last packet processed.

15.2.6.4.3. Control1 Field

The Control1 field (shown in [Table 641](#)) provides additional values used when hashing or encrypting/decrypting data. For blit operations, this field indicates the number of bytes in a frame buffer line, which is used to calculate the address for successive lines in each blit operation.

Table 641. DCP Control1 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CIPHER_CONFIG												HASH_SELECT				KEY_SELECT						CIPHER_MODE				CIPHER_SELECT					
																FRAMEBUFFER_LENGTH (in bytes, blit mode)															

The CIPHER_SELECT field selects from one of sixteen possible encryption algorithms (0=AES128). The CIPHER_MODE selects the mode of operation for that cipher (0=ECB, 1=CBC).

The KEY_SELECT field allows key selection for one of several sources. Keys can be included in the packet payload or may come from OTP or write-only registers.

The HASH_SELECT field selects a hashing algorithm for the operation (0=SHA-1, 1=CRC32).

The CIPHER_CONFIG field provides optional configuration information for the selected cipher. An example would be a key length for the RC4 algorithm.

15.2.6.4.4. Source Buffer

The SOURCE_BUFFER pointer (shown in [Table 642](#)) specifies the location of the source buffer in memory. The buffer may reside at any byte alignment and should refer to an on-chip SRAM or off-chip SDRAM location. For optimal performance, buffers should be word aligned. When the CONSTANT_FILL flag is set in the Control0 field, the value in this field is used as the data written to all destination buffer locations.

Table 642. DCP Source Buffer Field

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

15.2.6.4.5. Destination Buffer

The DESTINATION_BUFFER (shown in [Table 643](#)) specifies the location of the destination buffer in on-chip SRAM or off-chip SDRAM memory and may be set to any byte alignment. For in-place encryption, the destination buffer and source buffer should be the same value. For optimal performance, the buffer location should be word-aligned.

Table 643. DCP Destination Buffer Field

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

15.2.6.4.6. Buffer Size Field

The BUFFER_SIZE field (shown in [Table 644](#)) indicates the size of the buffers for memcopy, encryption, and hashing modes. For memcopy and hashing operations, the value may be any number of bytes (byte granularity), and for encryption modes, the length must be a multiple of the selected encryption algorithm's natural data size (16 bytes for AES).

Table 644. DCP Buffer Size Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NUMBER_LINES (blit mode)																BLIT_WIDTH (in bytes, blit mode)															
BUFFER_SIZE (in bytes, memcpy, encryption, hashing modes)																															

For blit operations, the buffer size field is split into two portions, a BLIT_WIDTH value, which specifies the number of bytes in each "line" of the blit operation, and a NUMBER_LINES value, which specifies how many vertical rows of pixels are in the image buffer.

15.2.6.4.7. Payload Pointer

Some operations require additional control values that are stored in a Payload Buffer (shown in [Table 645](#)), which is pointed to by this field. After the DCP reads the control packet, it examines the Control0 register and determines whether any payload information is required. If so, the DCP loads the payload from the address specified in this field. (See [Section 15.2.6.4.9](#) for more details.)

Table 645. DCP Payload Buffer Pointer

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

The payload area is also written to by the DCP at the completion of a hash operation (when the HASH_TERM) bit is set. Software must allocate the appropriate amount of storage (20 bytes for SHA-1 and 4 bytes for CRC32) in the payload or risk having the DCP write to an unallocated address.

15.2.6.4.8. Status

After the DCP engine has completed processing of a descriptor, it writes the packet status (shown in [Table 646](#)) back to the descriptor In the Status field. The packet status is the value of the channel status register at the time the packet completed processing.

Table 646. DCP Status Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
TAG								ERROR_CODE																				ERROR_DST	ERROR_SRC	ERROR_PACKET	ERROR_SETUP	HASH_MISMATCH	COMPLETE

For various error conditions, the DCP encodes additional error information in the ERROR_CODE field. See [Section 15.3](#) for more details on assignments of error codes. For any completion codes besides the COMPLETE flag, the channel suspends processing of the command chain until the error status values are cleared in the channel status register.

The format for this field is the same as for the channel status register, with the exception that the COMPLETE bit is written to the payload status but is not present in the channel status register.

15.2.6.4.9. Payload

The payload is a variable-length field that is used to provide key, initialization values, and expected results from hashing operations. The payload may consist of the data fields listed in [Table 647](#).

Table 647. DCP Payload Field

FIELD NAME	SIZE	DESCRIPTION	CONDITION
Cipher Key	16 bytes	AES key	Cipher enable with PAYLOAD_KEY
Cipher IV	16 bytes	CBC initialization vector	Cipher enable with CBC mode.
Hash Check	20 bytes	Hash completion value	Hash enabled with HASH_TERM fields set.

If fields are not used, they do not appear in the payload and the other payload values move upwards in the payload area. For instance, if only hashing is used, then the HASH_CHECK value would appear at offset 0 in the payload area. [Table 648](#) should be used by software to determine the amount of payload to allocate and initialize.

Table 648. DCP Payload Allocation by Software

CONTROL BITS PRESENT			PAYLOAD SIZE
HASH_CHECK	CIPHER_INIT	PAYLOAD_KEY	
0	0	0	0 words / 0 bytes
0	0	1	4 words / 16 bytes
0	1	0	4 words / 16 bytes
0	1	1	8 words / 32 bytes
1	0	0	5 words / 20 bytes

Table 648. DCP Payload Allocation by Software (Continued)

CONTROL BITS PRESENT			PAYLOAD SIZE
HASH_CHECK	CIPHER_INIT	PAYLOAD_KEY	
1	0	1	9 words / 36 bytes
1	1	0	9 words / 36 bytes
1	1	1	13 words / 52 bytes

For hashing operations, the DCP module writes the final hash value back to the start of the payload area for descriptors with the HASH_TERM bit set in the control packet. It is important that software allocate the required payload space, even though it is not required to set up the payload for control purposes.

15.2.7. Programming the Color-Space Converter

The color-space converter operates on planar YUV/YCbCr data that is output from the video decoder. (Planar indicates that each component is stored in a separate buffer.) As the CSC processes the data, it reads from each buffer and performs the necessary computations to convert from the YUV color space to the RGB color space. Output RGB data is then written to the RGB frame buffer.

Setting up the CSC for an operation is fairly straightforward. Software should first set up the HW_DCP_CSCLUMA register to point to the Y (luma) buffer and the HW_DCP_CSCCHROMAU and HW_DCP_CSCCHROMAV registers to point to the UV (chroma) buffers. The HW_DCP_CSCRGB register should then be written to point to the output RGB buffer. Each of these addresses are required to be 32-bit aligned. Software should then set up the coefficients for the operation for the desired conversion calculations. Typically, the coefficients could be set up one time and would likely not require adjustment.

15.2.7.1. Limitations

The following limitations apply to programming the color-space converter:

- The color-space pointers Y, U, V, and RGB must all be word-aligned.
- Image sizes should be multiples of four pixels in the X direction and multiples of two pixels in the Y direction.

15.2.7.2. Video Clipping

Video clipping is supported in the CSC by adjusting a few of the frame buffer parameters. The output frame buffer values (field/line size) should be programmed to the resolution of the attached LCD. These values provide a run-length for horizontal scanning, as well as the number of vertical lines to process. The CSC contains a separate register for the input frame buffer parameters that can be used to specify the line size of the input frame buffer. Since this value is independent of the output frame buffer line length, horizontal clipping is possible. After the logic processes pixels at the end of the line, it updates the output and input pointers according to their respective line lengths.

Vertical clipping can be achieved by simply offsetting the YCbCr buffer pointers to a non-zero line in the frame buffer. If 4:2:0 subsampling is used, this must be a multiple of 2 to ensure that the correct CrCb values are associated with the proper Y values.

For horizontal clipping, the YCbCr buffer pointers should be set to point to a non-zero offset within the line of the frame buffer. Again, it is important that the pointers be offset by the appropriate amount based on the color subsampling mode in use. YUV 4:1:1 should be offset in four-pixel increments, and 4:2:0 and 4:2:2 modes should be offset in two-pixel increments.

15.2.7.3. Video Rotation

The color-space converter also supports a simple video rotation mode to accommodate portrait-oriented displays that are used in landscape mode. In such scenarios, the line updates are done vertically versus horizontally, requiring that the frame buffer be rotated before being sent to the LCD interface. The color-space converter accomplishes this by processing the input YUV buffers in the normal manner, but then addressing the output RGB pixels so that they are stored in a vertical column in the RGB buffer instead of a horizontal row. The rotation function implemented produces an output RGB buffer that is mirrored around the top-left to bottom-right diagonal. It can also be envisioned as a 90-degree clockwise rotation with horizontal flip. This rotation was selected to simplify the address calculations required for the output pixels. Rotation is enabled by setting the ROTATE bit in the CSC control register.

The video rotation algorithm is performance-optimized for the non-scaled case. In this scenario, the CSC processes two horizontal scan lines of video at a time, which allows more efficient computation because the organization of the computational pipeline can accommodate the second pixel with no additional cycles since the second multiplier is idle for the luma multiplication. This results in fewer cycles required for the overall CSC conversion, as well as more optimal storage of the RGB buffer by allowing the CSC to generate two horizontal pixels per rotated line, which allows it to burst the two pixels to the RGB buffer.

15.2.7.4. Video Scaling

The color-space converter also supports a simple video scaling algorithm based on the Bresenham algorithm (replicate closest source pixel based on accumulating fractional steps across the source image). As the CSC processes each line, it either replicates the previous pixel, moves to the next pixel, or skips several pixels, depending on whether the incremental step pushes it to the next pixel in the source buffer. After each line is processed, the CSC determines whether the vertical pointers need to be incremented for the next line or remain the same, which results in the duplication of the previous line of output video. For power-of-two upscaling or downscaling, the algorithm produces a decimated or replicated image. For other steps, it produces a relatively good-looking scaled output without the additional computational (and hardware storage) requirements of a multi-tap filter.

Scaling is enabled by the SCALE field in the CSC control register. When this is enabled, two additional registers (XSCALE and YSCALE) are used to determine the output buffer width and height. In addition, these two registers provide integer/fractional counts that are used by the hardware when performing the scaling.

The INT value is determined by dividing the source resolution by the target resolution and rounding down. For downscaling operations, the INT value will be greater than 0 and for upscaling it will be 0. This inherently makes sense, since when downscaling, the hardware skips more than one pixel between RGB buffer computations. The hardware only supports downscaling by a factor of ~3, so the largest INT value supported is 0x02.

The FRAC value is determined by computing the modulo of the source and target buffer sizes. After processing a pixel in a line, the FRAC value is added to a running sum. The CSC then uses the INT and the overflow from the running sum (overflow is when the sum is greater than the scaled line length) to determine which is the next source pixel to process. When the INT+overflow is 1, the next pixel is processed; when greater than 1, pixels are skipped (downscaling), and when less than 1, the same pixel is processed again (upscaling).

Processing in the vertical direction works similarly, except that the pointers to the source line buffers are either incremented by 0, 1, 2, etc., for each new line.

15.2.7.5. CSC Programming Example

```

u32
    inputformat = 0,           // YCbCr 420 input data
    rotate      = 1,           // rotate image
    scale       = 1,           // no scaling
    outputmode  = 2,           // output format is 24-bit RGB
    width       = 320,         // original image width
    height      = 240,         // original image height
    clip_width  = 304,         // clipped image width (clip 16 pixels)
    clip_height = 224,         // clipped image height (clip 16 pixels)
    clip_xoff   = (width - clip_width) >> 1, // center clipped area (8)
    clip_yoff   = (height - clip_height) >> 1, // center clipped area (8)
    swidth      = 220,         // scaled width
    sheight     = 176;         // scaled height

volatile u32
    *rgbbuf,           // RGB output buffer pointer
    *csc_luma_data,    // YCbCr luma (Y) input buffer pointer
    *csc_chroma_u_data, // YCbCr chroma (U/Cb) buffer pointer
    *csc_chroma_v_data; // YCbCr chroma (V/Cr) buffer pointer

// The output buffer is defined by the clipped image size.
HW_DCP_CSCOUTBUFFPARAM_WR ((clip_height << 12) | clip_width);

// The input buffer width is determined by the original image. The height is determined by
// how many lines of video we want processed into the output buffer.
// NOTE: Height field here is ignored by the hardware.
// It is left here to be consistent with the OUTBUFFPARAM register.
HW_DCP_CSCINBUFFPARAM_WR ((clip_height << 12) | width);

// The RGB buffer is simply programmed into the RGB pointer register.
HW_DCP_CSCRGB_WR ((u32)&rgbbuf[0]);

// The Luma pointer must be shifted into the original buffer because of clipping.
// It is offset by the original line width * number of clipped lines
// PLUS the offset in the horizontal direction.
HW_DCP_CSCLUMA_WR ((u32)&csc_luma_data[clip_xoff + (clip_yoff * width)]);

// The chroma buffers are in 4:2:0 format, so they have half the resolution in each
// direction. The computation here is similar to luma, but takes into account the
// nature of the 4:2:0 data.
HW_DCP_CSCCHROMAU_WR ((u32)&csc_chroma_u_data[(clip_xoff >> 1) + ((clip_yoff * width) >> 2)]);
HW_DCP_CSCCHROMAV_WR ((u32)&csc_chroma_v_data[(clip_xoff >> 1) + ((clip_yoff * width) >> 2)]);

// For scaling, the clipped and scaled sizes are used for the scaling computations.
// The integer step is the clipped size / scaled size, and
// the fractional step is the clipped size modulo the scaled size.
// The DCP also needs to know the end scaled sizes in each direction.
HW_DCP_CSCXSCALE_WR ((clip_width / swidth) << 24 | (clip_width % swidth) << 12 | swidth);
HW_DCP_CSCYSCALE_WR ((clip_height / sheight) << 24 | (clip_height % sheight) << 12 | sheight);

// After the parameters are set up, software enables the CSC with the following options.
HW_DCP_CSCCTRL0_WR ((scale << 13) | (rotate << 12) | (outputmode << 8) | (inputformat << 4) | 1);

```

15.2.8. Programming Other DCP Functions

15.2.8.1. Basic Memory Copy Programming Example

To perform a basic memcpy operation, only a single descriptor is required. The DCP simply copies data from the source to the destination buffer. This process is illustrated in Figure 57.

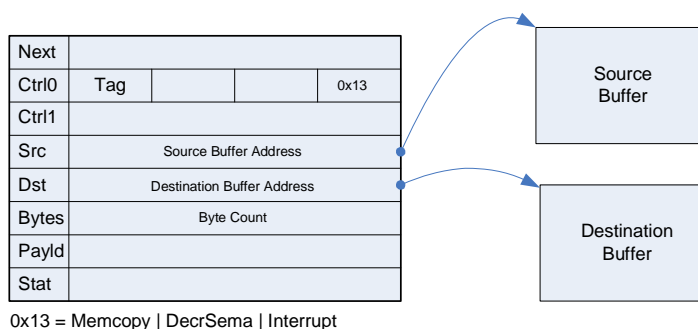


Figure 57. Basic Memory Copy Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32          *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32          *src,
                *dst,
                buf_size,
                *payload,
                stat;
} DCP_DESCRIPTOR;

DCP_DESCRIPTOR dcp1;
u32 *srcbuffer, *dstbuffer;

// set up control packet
dcp1.next = 0; // single packet in chain
dcp1.ctrl0.U = 0; // clear ctrl0 field
dcp1.ctrl0.B.ENABLE_MEMCOPY = 1; // enable memcopy
dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp1.ctrl0.B.INTERRUPT = 1; // interrupt
dcp1.ctrl1.U = 0; // clear ctrl1
dcp1.src = srcbuffer; // source buffer
dcp1.dst = dstbuffer; // destination buffer
dcp1.buf_size = 512; // 512 bytes
dcp1.payload = NULL; // not required
dcp1.status = 0; // clear status

// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1); // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 1); // increment semaphore by 1

// now wait for interrupt or poll
// polling code
while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );

// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>
```

15.2.8.2. Basic Hash Operation Programming Example

To perform a basic hash operation, only a single descriptor is required. The DCP simply reads data from the source buffer and computes the hash value on the contents. This process is illustrated in [Figure 58](#).

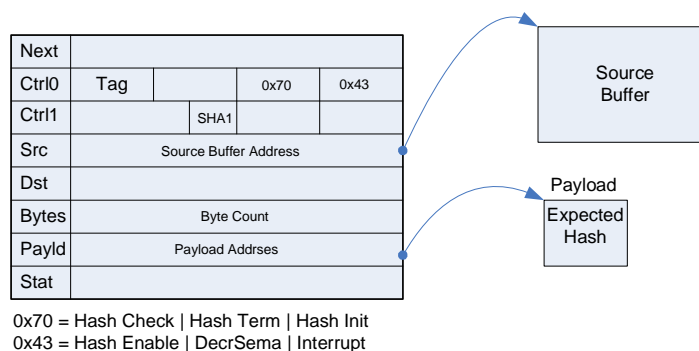


Figure 58. Basic Hash Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32          *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32          *src,
                *dst,
                buf_size,
                *payload,
                stat;
} DCP_DESCRIPTOR;

DCP_DESCRIPTOR dcp1;
u32 *srcbuffer;
u32 payload[5];

// set up expected hash check value
payload[0]=0x01234567;
payload[1]=0x89ABCDEF;
payload[2]=0x00112233;
payload[3]=0x44556677;
payload[4]=0x8899aabb;

// set up control packet
dcp1.next = 0; // single packet in chain
dcp1.ctrl0.U = 0; // clear ctrl0 field
dcp1.ctrl0.B.HASH_CHECK = 1; // check hash when complete
dcp1.ctrl0.B.HASH_INIT = 1; // initialize hash with this block
dcp1.ctrl0.B.HASH_TERM = 1; // terminate hash with this block
dcp1.ctrl0.B.ENABLE_HASH = 1; // enable hash
dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp1.ctrl0.B.INTERRUPT = 1; // interrupt
dcp1.ctrl1.U = 0; // clear ctrl1
dcp1.src = srcbuffer; // source buffer
dcp1.dst = 0; // no destination buffer
dcp1.buf_size = 512; // 512 bytes
dcp1.payload = payload; // holds expected hash value
dcp1.status = 0; // clear status

// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1); // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 1); // increment semaphore by 1

// now wait for interrupt or poll
// polling code
```

STMP3770

```

while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );

// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>

```

15.2.8.3. Basic Cipher Operation Programming Example

To perform a basic cipher operation, only a single descriptor is required. The DCP reads data from the source buffer, encrypts it, and writes it to the destination buffer. For this example, the key is provided in the payload and the algorithm uses the AES CBC mode of operation. This requires a payload with both the key and CBC initialization value. This process is illustrated in [Figure 59](#).

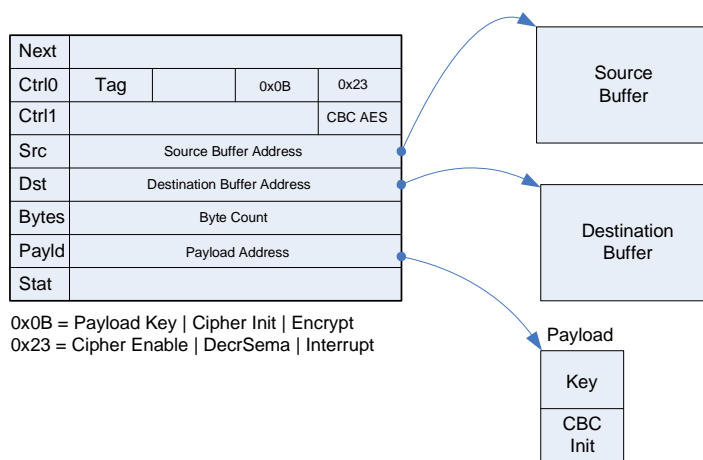


Figure 59. Basic Cipher Operation

```

<code>
typedef struct _dcp_descriptor
{
    u32          *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32          *src,
                *dst,
                buf_size,
                *payload,
                stat;
} DCP_DESCRIPTOR;

DCP_DESCRIPTOR dcp1;
u32 *srcbuffer;
u32 *dstbuffer;
u32 payload[8];

// set up key/CBC init in the payload
payload[0]=0x01234567;           // key
payload[1]=0x89ABCDEF;
payload[2]=0xfedcba98;
payload[3]=0x76543210;
payload[4]=0x00112233;           // CBC initialization
payload[5]=0x44556677;
payload[6]=0x8899aabb;
payload[7]=0xccddeeff;

```

```

// set up control packet
dcpl.next = 0; // single packet in chain
dcpl.ctrl0.U = 0; // clear ctrl0 field
dcpl.ctrl0.B.PAYLOAD_KEY = 1; // key is located in payload
dcpl.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcpl.ctrl0.B.CIPHER_INIT = 1; // init CBC for this block (from payload)
dcpl.ctrl0.B.ENABLE_CIPHER = 1; // enable cipher
dcpl.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcpl.ctrl0.B.INTERRUPT = 1; // interrupt
dcpl.ctrl1.U = 0; // clear ctrl1
dcpl.ctrl1.B.CIPHER_MODE = 1; // select CBC mode of operation
dcpl.src = srcbuffer; // source buffer
dcpl.dst = dstbuffer; // destination buffer
dcpl.buf_size = 512; // 512 bytes
dcpl.payload = payload; // holds key/CBC init
dcpl.status = 0; // clear status

// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcpl); // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 1); // increment semaphore by 1

// now wait for interrupt or poll
// polling code
while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );

// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>

```

15.2.8.4. Multi-Buffer Scatter/Gather Cipher and Hash Operation Programming Example

For this example, three separate buffers are encrypted and hashed with the results being directed to a unified buffer (gather operation). Three descriptors are used for the operation because there are three separate source buffer pointers. The DCP reads data from the source buffer and computes a hash on the unencrypted data. It then encrypts the data and writes it to the destination buffer. For this example, the key is located in the key RAM, and the algorithm uses the AES CBC mode of operation with a SHA-1 hash. The payload for the first operation contains the CBC initialization value, and the payload for the last packet contains the expected hash value. The middle packet requires no payload. This process is illustrated in [Figure 60](#).

STMP3770

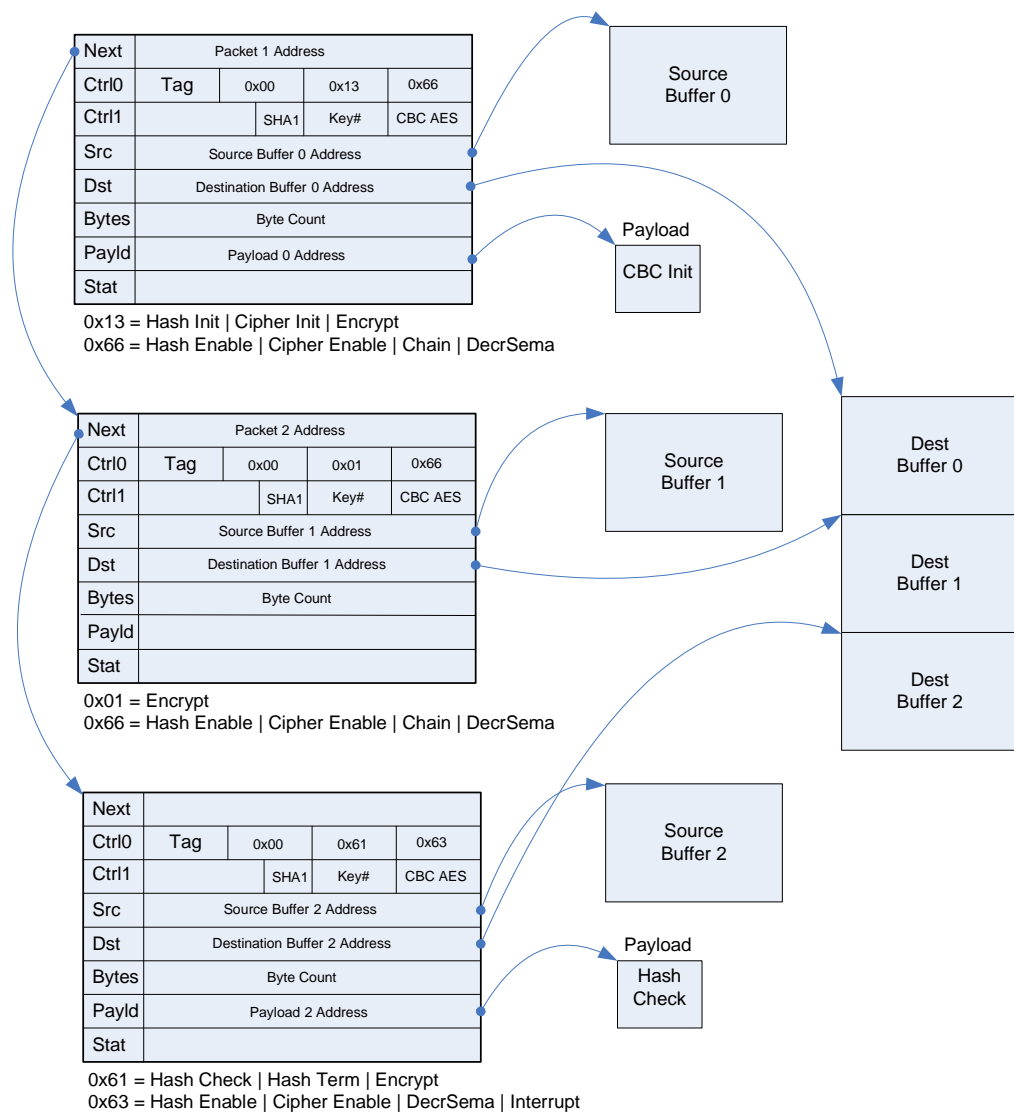


Figure 60. Multi-Buffer Scatter/Gather Cipher and Hash Operation

<code>

```
typedef struct _dcp_descriptor
{
    u32          *next;
    hw_dcp_packet1_t ctrl0;
    hw_dcp_packet2_t ctrl1;
    u32          *src,
    *dst,
    buf_size,
    *payload,
    stat;
} DCP_DESCRIPTOR;

DCP_DESCRIPTOR dcp1, dcp2, dcp3;
u32 *srcbuffer0, *srcbuffer2, *srcbuffer3;
u32 *dstbuffer;
u32 payload0[4], payload2[5];

// set up CBC init in the payload
payload0[0]=0x01234567; // key
```



```

payload0[1]=0x89ABCDEF;
payload0[2]=0xfedcba98;
payload0[3]=0x76543210;

payload2[0]=0x00112233;          // CBC initialization
payload2[1]=0x44556677;
payload2[2]=0x8899aabb;
payload2[3]=0xccddeeff;
payload2[3]=0xaabbccdd;

// set up control packet
dcp1.next = dcp2;                // point to packet 2
dcp1.ctrl0.U = 0;                // clear ctrl0 field
dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcp1.ctrl0.B.CIPHER_INIT = 1;    // init CBC for this block (from payload)
dcp1.ctrl0.B.HASH_INIT = 1;      // init hash this block
dcp1.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
dcp1.ctrl0.B.ENABLE_HASH = 1;    // enable cipher
dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp1.ctrl0.B.CHAIN = 1;          // chain to next packet
dcp1.ctrl1.U = 0;                // clear ctrl1
dcp1.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of operation
dcp1.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1; // select SHA1 hash
dcp1.ctrl1.B.KEY_SELECT = 2;     // select key #2
dcp1.src = srcbuffer0;           // source buffer
dcp1.dst = dstbuffer;            // destination buffer
dcp1.buf_size = 512;             // 512 bytes
dcp1.payload = payload0;         // holds key/CBC init
dcp1.status = 0;                 // clear status

// set up control packet
dcp2.next = dcp3;                // point to packet 2
dcp2.ctrl0.U = 0;                // clear ctrl0 field
dcp2.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcp2.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
dcp2.ctrl0.B.ENABLE_HASH = 1;    // enable cipher
dcp2.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp2.ctrl0.B.CHAIN = 1;          // chain to next packet
dcp2.ctrl1.U = 0;                // clear ctrl1
dcp2.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of operation
dcp2.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1; // select SHA1 hash
dcp2.ctrl1.B.KEY_SELECT = 2;     // select key #2
dcp2.src = srcbuffer1;           // source buffer
dcp2.dst = dstbuffer+512;        // destination buffer
dcp2.buf_size = 512;             // 512 bytes
dcp2.payload = NULL;             // no payload required
dcp2.status = 0;                 // clear status

// set up control packet
dcp3.next = dcp3;                // point to packet 2
dcp3.ctrl0.U = 0;                // clear ctrl0 field
dcp3.ctrl0.B.HASH_TERM = 1;      // terminate hash block
dcp3.ctrl0.B.HASH_CHECK = 1;     // check hash against payload value
dcp3.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcp3.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
dcp3.ctrl0.B.ENABLE_HASH = 1;    // enable cipher
dcp3.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp3.ctrl0.B.INTERRUPT = 1;      // interrupt on completion
dcp3.ctrl1.U = 0;                // clear ctrl1
dcp3.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of operation
dcp3.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1; // select SHA1 hash
dcp3.ctrl1.B.KEY_SELECT = 2;     // select key #2
dcp3.src = srcbuffer1;           // source buffer
dcp3.dst = dstbuffer+1024;       // destination buffer
dcp3.buf_size = 512;             // 512 bytes
dcp3.payload = payload2;         // payload is hash check value
dcp3.status = 0;                 // clear status

// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1);    // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 3);         // increment semaphore by 3 (for 3 packets)

// now wait for interrupt or poll
// polling code
while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );

// now check/clear channel status
if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {

```

STMP3770

```
// an error occurred
HW_DCP_CHnSTAT_CLR(0, 0xff);
}
// clear interrupt register
HW_DCP_STAT_CLR(1);
</code>
```

15.3. Programmable Registers

The following registers are available for programmer access and control of the DCP.

15.3.1. DCP Control Register 0 Description

The DCP Control Register 0 contains controls for the DCP module.

HW_DCP_CTRL	0x80028000
HW_DCP_CTRL_SET	0x80028004
HW_DCP_CTRL_CLR	0x80028008
HW_DCP_CTRL_TOG	0x8002800C

Table 649. HW_DCP_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				RSVD				GATHER_RESIDUAL_WRITES	ENABLE_CONTEXT_CACHING	ENABLE_CONTEXT_SWITCHING	RSVD										CSC_INTERRUPT_ENABLE		CHANNEL_INTERRUPT_ENABLE																														
CLKGATE																																																					
PRESENT_CRYPTO																																																					
PRESENT_CSC																																																					

Table 650. HW_DCP_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	Clear this bit to 0 to enable normal DCP operation. Set this bit to 1 (default) to disable clocking with the DCP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the DCP block to its default state.
30	CLKGATE	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block.
29	PRESENT_CRYPTO	RO	0x1	Indicates whether the crypto (cipher/hash) functions are present. Present = 0x1 Absent = 0x0
28	PRESENT_CSC	RO	0x1	Indicates whether the color-space conversion (CSC) function is present. Present = 0x1 Absent = 0x0
27:24	RSVD	RO	0x000000	Reserved.

Table 650. HW_DCP_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23	GATHER_RESIDUAL_WRITES	RW	0x1	Software should set this bit to enable ragged writes to unaligned buffers to be gathered between multiple write operations. This improves performance by removing several byte operations between write bursts. Trailing byte writes are held in a residual write data buffer and combined with a subsequent write to the buffer to form a word write.
22	ENABLE_CONTEXT_CACHING	RW	0x0	Software should set this bit to enable caching of contexts between operations. If only a single channel is used for encryption/hashing, enabling caching causes the context to not be reloaded if the channel was the last to be used.
21	ENABLE_CONTEXT_SWITCHING	RW	0x0	Enable automatic context switching for the channels. Software should set this bit if more than one channel is doing hashing or cipher operations that require context to be saved (for instance, when CBC mode is enabled). By disabling context switching, software can save the 160 bytes used for the context buffer.
20:9	RSVD	RO	0x000000	Reserved.
8	CSC_INTERRUPT_ENABLE	RW	0x0	Interrupt enable bit for the color-space converter (CSC). The CSC interrupt bit is combined with the interrupts from channels 1-3 to form the dcp_irq signal to the interrupt controller.
7:0	CHANNEL_INTERRUPT_ENABLE	RW	0x0	Per-channel interrupt enable bit. When set, the channel's interrupt gets routed to the interrupt controller. Channel 0 is routed to the dcp_vmi_irq signal and the other channels are combined (along with the CRC interrupt) into the dcp_irq signal. CH0 = 0x01 CH1 = 0x02 CH2 = 0x04 CH3 = 0x08

DESCRIPTION:

The DCP Control Register 0 contains the primary controls for the DCP block. The Present bits indicate which of the sub-features of the block are present in the hardware. The Context bits control how the DCP utilizes its context buffer, and the Gather Residual Writes bit controls how the master handles writing misaligned data to the bus. Each channel and the color-space converter contain an independent interrupt enable.

EXAMPLE:

```
HW_DCP_CTRL_SET(BM_DCP_CTRL_SFTRST);
HW_DCP_CTRL_CLR(BM_DCP_CTRL_SFTRST | BM_DCP_CTRL_CLKGATE);
```

15.3.2. DCP Status Register Description

The DCP Status Register provides interrupt status information for each channel.

HW_DCP_STAT	0x80028010
HW_DCP_STAT_SET	0x80028014
HW_DCP_STAT_CLR	0x80028018
HW_DCP_STAT_TOG	0x8002801C

STMP3770

Table 651. HW_DCP_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
RSVD				OTP_KEY_READY	CUR_CHANNEL				READY_CHANNELS								RSVD				CSCIRQ	RSVD				IRQ					

Table 652. HW_DCP_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x000000	Reserved.
28	OTP_KEY_READY	RO	0x1	When set, indicates that the OTP key has been shifted from the fuse block and is ready for use.
27:24	CUR_CHANNEL	RO	0x0	Current (active) channel (encoded). None = 0x0 CH0 = 0x1 CH1 = 0x2 CH2 = 0x3 CH3 = 0x4 CSC = 0x8
23:16	READY_CHANNELS	RO	0x0	Indicates which channels are ready to proceed with a transfer (active channel also included). Each bit is a one-hot, indicating the request status for the associated channel. CH0 = 0x01 CH1 = 0x02 CH2 = 0x04 CH3 = 0x08
15:9	RSVD	RO	0x000000	Reserved.
8	CSCIRQ	RW	0x0	Indicates that the color-space-converter logic (CSC) has a pending interrupt request. The CSC interrupt appears on the dcp_irq line along with interrupts from channels 1-3.
7:4	RSVD	RO	0x000000	Reserved.
3:0	IRQ	RW	0x0	Indicates which channels have pending interrupt requests. Channel 0's interrupt is routed through the dcp_vmi_irq and the other interrupt bits are routed through the dcp_irq.

DESCRIPTION:

This register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE:

```
HW_DCP_STAT_CLR(BM_DCP_STAT_CSCIRQ); // clear CSC interrupt
```

15.3.3. DCP Channel Control Register Description

The DCP Channel Control Register provides controls for channel arbitration and channel enables.

HW_DCP_CHANNELCTRL 0x80028020
 HW_DCP_CHANNELCTRL_SET 0x80028024
 HW_DCP_CHANNELCTRL_CLR 0x80028028
 HW_DCP_CHANNELCTRL_TOG 0x8002802C

Table 653. HW_DCP_CHANNELCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00				
RSVD													CSC_PRIORITY		CH0_IRQ_MERGED	HIGH_PRIORITY_CHANNEL										ENABLE_CHANNEL									

Table 654. HW_DCP_CHANNELCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:19	RSVD	RO	0x0000	Reserved.
18:17	CSC_PRIORITY	RW	0x0	Sets the CSC priority within the channel arbitration scheme: HIGH = 0x3 Enables CSC at same level as a high-priority channel MED = 0x2 Enables CSC at a level below high-priority channels and above low-priority channels LOW = 0x1 Enables CSC at same level as a normal-priority channel BACKGROUND = 0x0 Enables CSC at a level below normal-priority channels (in the background)
16	CH0_IRQ_MERGED	RW	0x0	Indicates that the interrupt for Channel 0 should be merged with the other interrupts on the shared dcp_irq interrupt. 0 = Channel 0's interrupt is routed to the separate dcp_vmi_irq. 1 = The interrupt is routed to the shared DCP interrupt.

STMP3770

Table 654. HW_DCP_CHANNELCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	HIGH_PRIORITY_CHANNEL	RW	0x0	Setting a bit in this field causes the corresponding channel to have high-priority arbitration. High-priority channels are arbitrated round-robin and take precedence over other channels that are not marked as high-priority. CH0 = 0x01 CH1 = 0x02 CH2 = 0x04 CH3 = 0x08
7:0	ENABLE_CHANNEL	RW	0x0	Setting a bit in this field enables the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When not enabled, the channel is denied access to the central DMA resources. CH0 = 0x01 CH1 = 0x02 CH2 = 0x04 CH3 = 0x08

DESCRIPTION:

The DCP Channel Control Register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE:

```
BW_DCP_CHANNELCTRL_ENABLE_CHANNEL(BV_DCP_CHANNELCTRL_ENABLE_CHANNEL__CH0); // enable channel 0
```

15.3.4. DCP Capability 0 Register Description

The DCP Capability 0 Register contains additional information about the DCP module implementation parameters.

HW_DCP_CAPABILITY0 0x80028030

Table 655. HW_DCP_CAPABILITY0

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

Table 656. HW_DCP_CAPABILITY0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x000000	Reserved.
11:8	NUM_CHANNELS	RO	0x4	Encoded value indicating the number of channels implemented in the design.
7:0	NUM_KEYS	RO	0x4	Encoded value indicating the number of key storage locations implemented in the design.

DESCRIPTION:

The DCP Capability 0 Register provides capability information for the DCP block. It indicates the number of channels implemented as well as the number of key storage locations available for software use.

15.3.5. DCP Capability 1 Register Description

The DCP Capability 1 Register contains information about the algorithms available in the implementation.

HW_DCP_CAPABILITY1 0x80028040

Table 657. HW_DCP_CAPABILITY1

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

Table 658. HW_DCP_CAPABILITY1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	HASH_ALGORITHMS	RO	0x1	One-hot field indicating which hashing algorithms are available. SHA1 = 0x0001 CRC32 = 0x0002
15:0	CIPHER_ALGORITHMS	RO	0x1	One-hot field indicating which cipher algorithms are available. AES128 = 0x0001

DESCRIPTION:

The DCP Capability 1 Register provides capability information for the DCP block. It contains two fields indicating which encryption and hashing algorithms are present in the design. Each bit set indicates that support for the associated function is present.

15.3.6. DCP Context Buffer Pointer Description

The DCP Context Buffer Pointer contains a pointer to the memory region to be used for DCP context swap operations.

HW_DCP_CONTEXT 0x80028050

Table 659. HW_DCP_CONTEXT

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

STMP3770

Table 660. HW_DCP_CONTEXT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x0	Context pointer address. Address should be located in system RAM and should be word-aligned for optimal performance.

DESCRIPTION:

The DCP Context Buffer Pointer contains a pointer to the start of the context pointer memory in on-chip SRAM or off-chip SDRAM. This buffer stores state information when the DCP module changes from one channel to another.

15.3.7. DCP Key Index Register Description

The DCP Key Index Register contains a pointer to the key location to be written.

HW_DCP_KEY

0x80028060

Table 661. HW_DCP_KEY

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

Table 662. HW_DCP_KEY Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:8	RSVD	RO	0x000000	Reserved.
7:6	RSVDINDEX	RO	0x000000	Reserved.
5:4	INDEX	RW	0x0	Key index pointer. Valid indices are 0-[number_keys].
3:2	RSVDSUBWORD	RO	0x0	Reserved.
1:0	SUBWORD	RW	0x0	Key subword pointer. Valid indices are 0-3. After each write to the key data register, this field increments.

DESCRIPTION:

The DCP module maintains a set of write-only keys that may be used by software. To write a key, software must first write the desired key index/subword to the DCP Key Index Register and then write the key values to the key registers (below). After each write to the key data register, the SUBWORD field increments to allow software to write the subsequent key to be written without having to rewrite the key index.

EXAMPLE:

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
HW_DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0)); // set key index to key 0, subword 0
HW_DCP_KEYDATA_WR(0xccddeeff); // write key values (subword 0)
HW_DCP_KEYDATA_WR(0x8899aabb); // write key values (subword 1)
HW_DCP_KEYDATA_WR(0x44556677); // write key values (subword 2)
HW_DCP_KEYDATA_WR(0x00112233); // write key values (subword 3)
```


15.3.8. DCP Key Data Description

The DCP Key Data Register provides write access to the key/key subword specified by the DCP Key Index Register.

HW_DCP_KEYDATA 0x80028070

Table 663. HW_DCP_KEYDATA

31	30	29	28	27	26	25	24	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 664. HW_DCP_KEYDATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	DATA	RW	0x0	Word 0 data for key. This is the least-significant word.

DESCRIPTION:

Writing this location updates the selected subword for the key located at the index specified by the DCP Key Index Register. A write also triggers the SUBWORD field of the DCP Key Index Register to increment to the next higher word in the key.

EXAMPLE:

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
HW_DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0)); // set key index to key 0, subword 0
HW_DCP_KEYDATA_WR(0xccddeeff); // write key values (subword 0)
HW_DCP_KEYDATA_WR(0x8899aabb); // write key values (subword 1)
HW_DCP_KEYDATA_WR(0x44556677); // write key values (subword 2)
HW_DCP_KEYDATA_WR(0x00112233); // write key values (subword 3)
```

15.3.9. DCP Work Packet 0 Status Register Description

This register displays the values for the current work packet offset 0x00 (Next Command) field.

HW_DCP_PACKET0 0x80028080

Table 665. HW_DCP_PACKET0

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

Table 666. HW_DCP_PACKET0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RO	0x0	Next Pointer Register,

DESCRIPTION:

The Work Packet Status registers show the contents of the currently executing packet. When the channels are inactive (or the CSC is active), the packet status register returns 0. The register bits are fully documented here to document the packet structure in memory.

STMP3770

15.3.10. DCP Work Packet 1 Status Register Description

The DCP Work Packet 1 Status Register displays the values for the current work packet offset 0x04 (control) field.

HW_DCP_PACKET1

0x80028090

Table 667. HW_DCP_PACKET1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TAG								OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDSWAP	KEY_BYTESWAP	TEST_SEMA_IRQ	CONSTANT_FILL	HASH_OUTPUT	CHECK_HASH	HASH_TERM	HASH_INIT	PAYLOAD_KEY	OTP_KEY	CIPHER_INIT	CIPHER_ENCRYPT	ENABLE_BLIT	ENABLE_HASH	ENABLE_CIPHER	ENABLE_MEMCOPY	CHAIN_CONTIGUOUS	CHAIN	DECR_SEMAPHORE	INTERRUPT

Table 668. HW_DCP_PACKET1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	TAG	RO	0x0	Packet Tag
23	OUTPUT_WORDSWAP	RO	0x0	Reflects whether the DCP engine word-swaps output data (big-endian data).
22	OUTPUT_BYTESWAP	RO	0x0	Reflects whether the DCP engine byte-swaps output data (big-endian data).
21	INPUT_WORDSWAP	RO	0x0	Reflects whether the DCP engine word-swaps input data (big-endian data).
20	INPUT_BYTESWAP	RO	0x0	Reflects whether the DCP engine byte-swaps input data (big-endian data).
19	KEY_WORDSWAP	RO	0x0	Reflects whether the DCP engine swaps key words (big-endian key).
18	KEY_BYTESWAP	RO	0x0	Reflects whether the DCP engine swaps key bytes (big-endian key).
17	TEST_SEMA_IRQ	RO	0x0	This bit is used to test the channel semaphore transition to 0. FOR TEST USE ONLY!
16	CONSTANT_FILL	RO	0x0	When this bit is set (MEMCOPY and BLIT modes only), the DCP simply fills the destination buffer with the value found in the Source Address field.
15	HASH_OUTPUT	RO	0x0	When hashing is enabled, this bit controls whether the input or output data is hashed. INPUT = 0x00 OUTPUT = 0x01
14	CHECK_HASH	RO	0x0	Reflects whether the calculated hash value should be compared against the hash provided in the payload.
13	HASH_TERM	RO	0x0	Reflects whether the current hashing block is the final block in the hashing operation, so the hash padding should be applied by hardware.
12	HASH_INIT	RO	0x0	Reflects whether the current hashing block is the initial block in the hashing operation, so the hash registers should be initialized before the operation.

Table 668. HW_DCP_PACKET1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11	PAYLOAD_KEY	RO	0x0	When set, indicates the payload contains the key. This bit takes precedence over the OTP_KEY control.
10	OTP_KEY	RO	0x0	Reflects whether a hardware-based key should be used. The KEY_SELECT field from the Control1 field is used to select from multiple hardware keys. The PAYLOAD_KEY bit takes precedence over the OTP_KEY bit.
9	CIPHER_INIT	RO	0x0	Reflects whether the cipher block should load the initialization vector from the payload for this operation.
8	CIPHER_ENCRYPT	RO	0x0	When the cipher block is enabled, this bit indicates whether the operation is encryption or decryption. ENCRYPT = 0x01 DECRYPT = 0x00
7	ENABLE_BLIT	RO	0x0	Reflects whether the DCP should perform a blit operation. Source data is always continuous and the destination buffer is written in run/stride format. When set, the BUFFER_SIZE field is treated as two 16-bit values for the X-Y extents of the blit operation.
6	ENABLE_HASH	RO	0x0	Reflects whether the selected hashing function should be enabled for this operation.
5	ENABLE_CIPHER	RO	0x0	Reflects whether the selected cipher function should be enabled for this operation.
4	ENABLE_MEMCOPY	RO	0x0	Reflects whether the selected hashing function should be enabled for this operation.
3	CHAIN_CONTIGUOUS	RO	0x000000	Reflects whether the next packet's address is located following this packet's payload.
2	CHAIN	RO	0x0	Reflects whether the next command pointer register should be loaded into the channel's current descriptor pointer.
1	DECR_SEMAPHORE	RO	0x0	Reflects whether the channel's semaphore should be decremented at the end of the current operation. When the semaphore reaches a value of 0, no more operations will be issued from the channel.
0	INTERRUPT	RO	0x0	Reflects whether the channel should issue an interrupt upon completion of the packet.

DESCRIPTION:

The DCP Work Packet 1 Status Register shows the contents of the Control0 register from the packet being processed.

15.3.11. DCP Work Packet 2 Status Register Description

The DCP Work Packet 2 Status Register displays the values for the current work packet offset 0x08 (Control1) field.

HW_DCP_PACKET2

0x800280A0

STMP3770

Table 669. HW_DCP_PACKET2

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

Table 670. HW_DCP_PACKET2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	CIPHER_CFG	RO	0x0	Cipher configuration bits. Optional configuration bits required for ciphers.
23:20	RSVD	RO	0x0	Reserved.
19:16	HASH_SELECT	RO	0x0	Hash Selection SHA1 = 0x00 CRC32 = 0x01
15:8	KEY_SELECT	RO	0x0	Key Selection. The value here reflects the key index for the cipher operation.
7:4	CIPHER_MODE	RO	0x0	Cipher Mode Selection. Reflects the mode of operation for cipher operations. ECB = 0x00 CBC = 0x01
3:0	CIPHER_SELECT	RO	0x0	Cipher Selection Field AES128 = 0x00

DESCRIPTION:

The DCP Work Packet 2 Status Register shows the contents of the Control0 register from the packet being processed.

15.3.12. DCP Work Packet 3 Status Register Description

The DCP Work Packet 3 Status Register displays the values for the current work packet offset 0x0C (Source Address) field.

HW_DCP_PACKET3 0x800280B0

Table 671. HW_DCP_PACKET3

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

Table 672. HW_DCP_PACKET3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RO	0x0	Source Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

DESCRIPTION:

The DCP Work Packet 3 Status Register shows the contents of the Source Address register from the packet being processed. When the CONSTANT_FILL bit in the Control0 field is set, this field contains the data written to the destination buffer.

15.3.13. DCP Work Packet 4 Status Register Description

The DCP Work Packet 4 Status Register displays the values for the current work packet offset 0x10 (Destination Address) field.

HW_DCP_PACKET4 0x800280C0

Table 673. HW_DCP_PACKET4

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

Table 674. HW_DCP_PACKET4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RO	0x0	Destination Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

DESCRIPTION:

The DCP Work Packet 4 Status Register shows the contents of the Destination Address register from the packet being processed.

15.3.14. DCP Work Packet 5 Status Register Description

The DCP Work Packet 5 Status Register displays the values for the current work packet offset 0x14 (Buffer Size) field.

HW_DCP_PACKET5 0x800280D0

Table 675. HW_DCP_PACKET5

31	30	29	28	27	26	25	24	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 676. HW_DCP_PACKET5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	COUNT	RO	0x0	Byte Count register. This value is the working value and will update as the operation proceeds.

DESCRIPTION:

The DCP Work Packet 5 Status Register shows the contents of the bytecount register from the packet being processed. The field can be considered either a byte count or a buffer size. The logic treats this as a decrementing count of bytes from the buffer size programmed into the field. As the transaction proceeds, the logic decrements the bytecount as data is written to the destination buffer. For blit operations,

the top 16-bits of this field represents the number of lines (y size) in the blit and the lower 16-bits represent the number of bytes in a line (x size).

15.3.15. DCP Work Packet 6 Status Register Description

The DCP Work Packet 6 Status Register displays the values for the current work packet offset 0x1C (Payload Pointer) field.

HW_DCP_PACKET6 0x800280E0

Table 677. HW_DCP_PACKET6

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

Table 678. HW_DCP_PACKET6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RO	0x0	This register reflects the payload pointer for the current control packet.

DESCRIPTION:

The DCP Work Packet 6 Status Register shows the contents of the payload pointer field from the packet being processed.

15.3.16. DCP Channel 0 Command Pointer Address Register Description

The DCP Channel 0 Command Pointer Address Register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value arbitrate for access to the engine for the subsequent operation.

HW_DCP_CH0CMDPTR 0x80028100

Table 679. HW_DCP_CH0CMDPTR

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

Table 680. HW_DCP_CH0CMDPTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x00000000	Pointer to descriptor structure to be processed for Channel 0.

DESCRIPTION:

DCP Channel 0 is controlled by a variable sized command structure. The DCP Channel 0 Command Pointer Address Register points to the command structure to be executed.

EXAMPLE:

```
HW_DCP_CHnCMDPTR_WR(0, v); // Write channel 0 command pointer
pCurptr = (hw_DCP_chncmdptr_t *) HW_DCP_CHnCMDPTR_RD(0); // Read current command pointer
```

15.3.17. DCP Channel 0 Semaphore Register Description

The DCP Channel 0 Semaphore Register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a decrement semaphore bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the decrement_semaphore bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error has occurred.

HW_DCP_CH0SEMA 0x80028110

Table 681. HW_DCP_CH0SEMA

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

Table 682. HW_DCP_CH0SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:16	VALUE	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	RSVD	RO	0x0	Reserved.
7:0	INCREMENT	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP 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 DCP channel decrements the count on the same clock, then the count is incremented by a net 1. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

DESCRIPTION:

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. After processing each control packet, the DCP decrements the semaphore if it is non-zero. The channel continues processing packets as long as the semaphore contains a non-zero value and the CHAIN or CHAIN_CONTIGUOUS control bits in the Control0 field are set.

15.3.18. DCP Channel 0 Status Register Description

The DCP Channel 0 Status Register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH0STAT	0x80028120
HW_DCP_CH0STAT_SET	0x80028124
HW_DCP_CH0STAT_CLR	0x80028128
HW_DCP_CH0STAT_TOG	0x8002812C

Table 683. HW_DCP_CH0STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TAG								ERROR_CODE								RSVD								ERROR_DST	ERROR_SRC	ERROR_PACKET	ERROR_SETUP	HASH_MISMATCH	RSVDCOMPLETE		

Table 684. HW_DCP_CH0STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	TAG	RO	0x00	Indicates the tag from the last completed packet in the command structure
23:16	ERROR_CODE	RW	0x0	Indicates additional error codes for some error conditions. NEXT_CHAIN_IS_0 = 0x01 Error signalled because the next pointer is 0x00000000 NO_CHAIN = 0x02 Error signalled because the semaphore is nonzero and neither chain bit is set CONTEXT_ERROR = 0x03 Error signalled because an error was reported reading/writing the context buffer PAYLOAD_ERROR = 0x04 Error signalled because an error was reported reading/writing the payload INVALID_MODE = 0x05 Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15:6	RSVD	RO	0x0000	Reserved.
5	ERROR_DST	RW	0x0	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by software.
4	ERROR_SRC	RW	0x0	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by software.

Table 684. HW_DCP_CH0STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	ERROR_PACKET	RW	0x0	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by software.
2	ERROR_SETUP	RW	0x0	This bit indicates that the hardware has detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing stops until the error is handled by software.
1	HASH_MISMATCH	RW	0x0	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by software.
0	RSVDCOMPLETE	RO	0x0	This bit always reads as 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

DESCRIPTION:

The DCP Channel 0 Status Register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

15.3.19. DCP Channel 0 Options Register Description

The DCP Channel 0 Options Register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH0OPTS	0x80028130
HW_DCP_CH0OPTS_SET	0x80028134
HW_DCP_CH0OPTS_CLR	0x80028138
HW_DCP_CH0OPTS_TOG	0x8002813C

Table 685. HW_DCP_CH0OPTS

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

Table 686. HW_DCP_CH0OPTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	RSVD	RO	0x0	Reserved.
15:0	RECOVERY_TIMER	RW	0x0	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initialized with this value and then decremented until the timer reaches 0. The channel will not initiate another operation for the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0 ns to 8.3 ms at 133-MHz operation.

DESCRIPTION:

The DCP Channel 0 Options Register can be used to control optional features of the channels.

15.3.20. DCP Channel 1 Command Pointer Address Register Description

The DCP Channel 1 Command Pointer Address Register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value arbitrate for access to the engine for the subsequent operation.

HW_DCP_CH1CMDPTR 0x80028140

Table 687. HW_DCP_CH1CMDPTR

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

Table 688. HW_DCP_CH1CMDPTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x00000000	Pointer to descriptor structure to be processed for Channel 1.

DESCRIPTION:

DCP Channel 1 is controlled by a variable sized command structure. The DCP Channel 1 Command Pointer Address Register points to the command structure to be executed.

EXAMPLE:

```
HW_DCP_CHn_CMDPTR_WR(1, v); // Write channel 1 command pointer
pCurptr = (hw_dcp_chn_cmdptr_t *) HW_DCP_CHn_CMDPTR_RD(1); // Read current command pointer
```

15.3.21. DCP Channel 1 Semaphore Register Description

The DCP Channel 1 Semaphore Register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address is not loaded into the CMDPTR register. Each packet also contains a decrement semaphore bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the decrement_semaphore bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error has occurred.

HW_DCP_CH1SEMA 0x80028150

Table 689. HW_DCP_CH1SEMA

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

Table 690. HW_DCP_CH1SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:16	VALUE	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.

STMP3770

Table 690. HW_DCP_CH1SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	RSVD	RO	0x0	Reserved.
7:0	INCREMENT	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP 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 DCP channel decrements the count on the same clock, then the count is incremented by a net 1. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

DESCRIPTION:

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of 0. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

15.3.22. DCP Channel 1 Status Register Description

The DCP Channel 1 Status Register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH1STAT	0x80028160
HW_DCP_CH1STAT_SET	0x80028164
HW_DCP_CH1STAT_CLR	0x80028168
HW_DCP_CH1STAT_TOG	0x8002816C

Table 691. HW_DCP_CH1STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00				
TAG								ERROR_CODE								RSVD								ERROR_DST		ERROR_SRC		ERROR_PACKET		ERROR_SETUP		HASH_MISMATCH		RSVDCOMPLETE	

Table 692. HW_DCP_CH1STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	TAG	RO	0x00	Indicates the tag from the last completed packet in the command structure
23:16	ERROR_CODE	RW	0x0	Indicates additional error codes for some error conditions. NEXT_CHAIN_IS_0 = 0x01 Error signalled because the next pointer is 0x00000000 NO_CHAIN = 0x02 Error signalled because the semaphore is nonzero and neither chain bit is set CONTEXT_ERROR = 0x03 Error signalled because an error was reported reading/writing the context buffer PAYLOAD_ERROR = 0x04 Error signalled because an error was reported reading/writing the payload INVALID_MODE = 0x05 Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15:6	RSVD	RO	0x0000	Reserved.
5	ERROR_DST	RW	0x0	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by software.
4	ERROR_SRC	RW	0x0	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by software.
3	ERROR_PACKET	RW	0x0	This bit indicates that a bus error occurs when reading the packet or payload or when writing status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by software.
2	ERROR_SETUP	RW	0x0	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing stops until the error is handled by software.
1	HASH_MISMATCH	RW	0x0	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by software.
0	RSVDCOMPLETE	RO	0x0	This bit always reads as 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

DESCRIPTION:

The DCP Channel 1 Status Register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was

the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

15.3.23. DCP Channel 1 Options Register Description

The DCP Channel 1 Options Register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH1OPTS	0x80028170
HW_DCP_CH1OPTS_SET	0x80028174
HW_DCP_CH1OPTS_CLR	0x80028178
HW_DCP_CH1OPTS_TOG	0x8002817C

Table 693. HW_DCP_CH1OPTS

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

Table 694. HW_DCP_CH1OPTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	RSVD	RO	0x0	Reserved.
15:0	RECOVERY_TIMER	RW	0x0	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initialized with this value and then decremented until the timer reaches 0. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0 ns to 8.3 ms at 133-MHz operation.

DESCRIPTION:

The DCP Channel 1 Options Register can be used to control optional features of the channels.

15.3.24. DCP Channel 2 Command Pointer Address Register Description

The DCP Channel 2 Command Pointer Address Register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value arbitrates for access to the engine for the subsequent operation.

HW_DCP_CH2CMDPTR	0x80028180
------------------	------------

Table 695. HW_DCP_CH2CMDPTR

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

Table 696. HW_DCP_CH2CMDPTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x00000000	Pointer to descriptor structure to be processed for Channel 2.

DESCRIPTION:

DCP Channel 2 is controlled by a variable sized command structure. The DCP Channel 2 Command Pointer Address Register points to the command structure to be executed.

EXAMPLE:

```
HW_DCP_CHn_CMDPTR_WR(2, v); // Write channel 2 command pointer
pCurptr = (hw_dcp_chn_cmdptr_t *) HW_DCP_CHn_CMDPTR_RD(2); // Read current command pointer
```

15.3.25. DCP Channel 2 Semaphore Register Description

The DCP Channel 2 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address is not loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error has occurred.

HW_DCP_CH2SEMA 0x80028190

Table 697. HW_DCP_CH2SEMA

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

Table 698. HW_DCP_CH2SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:16	VALUE	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	RSVD	RO	0x0	Reserved.
7:0	INCREMENT	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP 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 DCP channel decrements the count on the same clock, then the count is incremented by a net 1. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

DESCRIPTION:

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of 0. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

15.3.26. DCP Channel 2 Status Register Description

The DCP Channel 2 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH2STAT	0x800281A0
HW_DCP_CH2STAT_SET	0x800281A4
HW_DCP_CH2STAT_CLR	0x800281A8
HW_DCP_CH2STAT_TOG	0x800281AC

Table 699. HW_DCP_CH2STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
TAG								ERROR_CODE								RSVD												ERROR_DST	ERROR_SRC	ERROR_PACKET	ERROR_SETUP	HASH_MISMATCH	RSVDCOMPLETE

Table 700. HW_DCP_CH2STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	TAG	RO	0x00	Indicates the tag from the last completed packet in the command structure
23:16	ERROR_CODE	RW	0x0	Indicates additional error codes for some error conditions. NEXT_CHAIN_IS_0 = 0x01 Error signalled because the next pointer is 0x00000000 NO_CHAIN = 0x02 Error signalled because the semaphore is nonzero and neither chain bit is set CONTEXT_ERROR = 0x03 Error signalled because an error was reported reading/writing the context buffer PAYLOAD_ERROR = 0x04 Error signalled because an error was reported reading/writing the payload INVALID_MODE = 0x05 Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15:6	RSVD	RO	0x0000	Reserved.
5	ERROR_DST	RW	0x0	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by software.
4	ERROR_SRC	RW	0x0	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by software.
3	ERROR_PACKET	RW	0x0	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by software.
2	ERROR_SETUP	RW	0x0	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing stops until the error is handled by software.
1	HASH_MISMATCH	RW	0x0	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by software.
0	RSVDCOMPLETE	RO	0x0	This bit always reads as 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

DESCRIPTION:

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was

the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

15.3.27. DCP Channel 2 Options Register Description

The DCP Channel 2 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH2OPTS	0x800281B0
HW_DCP_CH2OPTS_SET	0x800281B4
HW_DCP_CH2OPTS_CLR	0x800281B8
HW_DCP_CH2OPTS_TOG	0x800281BC

Table 701. HW_DCP_CH2OPTS

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

Table 702. HW_DCP_CH2OPTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	RSVD	RO	0x0	Reserved.
15:0	RECOVERY_TIMER	RW	0x0	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initialized with this value and then decremented until the timer reaches 0. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0 ns to 8.3 ms at 133-MHz operation.

DESCRIPTION:

The options register can be used to control optional features of the channels.

15.3.28. DCP Channel 3 Command Pointer Address Register Description

The DCP Channel 3 Command Pointer Address Register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value arbitrate for access to the engine for the subsequent operation.

HW_DCP_CH3CMDPTR	0x800281C0
------------------	------------

Table 703. HW_DCP_CH3CMDPTR

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

Table 704. HW_DCP_CH3CMDPTR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x00000000	Pointer to descriptor structure to be processed for Channel 3.

DESCRIPTION:

DCP Channel 3 is controlled by a variable sized command structure. The DCP Channel 3 Command Pointer Address Register points to the command structure to be executed.

EXAMPLE:

```
HW_DCP_CHn_CMDPTR_WR(3, v); // Write channel 3 command pointer
pCurptr = (hw_dcp_chn_cmdptr_t *) HW_DCP_CHn_CMDPTR_RD(3); // Read current command pointer
```

15.3.29. DCP Channel 3 Semaphore Register Description

The DCP Channel 3 Semaphore Register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address is not loaded into the CMDPTR register. Each packet also contains a decrement semaphore bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the decrement_semaphore bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic also clears the semaphore if an error has occurred.

HW_DCP_CH3SEMA 0x800281D0

Table 705. HW_DCP_CH3SEMA

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

Table 706. HW_DCP_CH3SEMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:16	VALUE	RO	0x0	This read-only field shows the current (instantaneous) value of the semaphore counter.
15:8	RSVD	RO	0x0	Reserved.
7:0	INCREMENT	RW	0x00	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP 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 DCP channel decrements the count on the same clock, then the count is incremented by a net 1. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

DESCRIPTION:

Each DCP channel has an 8-bit counting semaphore that is used to synchronize between the program stream and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of 0. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

15.3.30. DCP Channel 3 Status Register Description

The DCP Channel 3 Status Register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH3STAT	0x800281E0
HW_DCP_CH3STAT_SET	0x800281E4
HW_DCP_CH3STAT_CLR	0x800281E8
HW_DCP_CH3STAT_TOG	0x800281EC

Table 707. HW_DCP_CH3STAT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
TAG								ERROR_CODE								RSVD												ERROR_DST	ERROR_SRC	ERROR_PACKET	ERROR_SETUP	HASH_MISMATCH	RSVDCOMPLETE

Table 708. HW_DCP_CH3STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	TAG	RO	0x00	Indicates the tag from the last completed packet in the command structure
23:16	ERROR_CODE	RW	0x0	Indicates additional error codes for some error conditions. NEXT_CHAIN_IS_0 = 0x01 Error signalled because the next pointer is 0x00000000 NO_CHAIN = 0x02 Error signalled because the semaphore is nonzero and neither chain bit is set CONTEXT_ERROR = 0x03 Error signalled because an error was reported reading/writing the context buffer PAYLOAD_ERROR = 0x04 Error signalled because an error was reported reading/writing the payload INVALID_MODE = 0x05 Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15:6	RSVD	RO	0x0000	Reserved.
5	ERROR_DST	RW	0x0	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing stops until the error is handled by software.
4	ERROR_SRC	RW	0x0	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing stops until the error is handled by software.
3	ERROR_PACKET	RW	0x0	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet payload. When an error is detected, the channel's processing stops until the error is handled by software.
2	ERROR_SETUP	RW	0x0	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing stops until the error is handled by software.
1	HASH_MISMATCH	RW	0x0	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing stops until the error is handled by software.
0	RSVDCOMPLETE	RO	0x0	This bit always reads as 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

DESCRIPTION:

The DCP Channel 3 Status Register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt is generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was

the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

15.3.31. DCP Channel 3 Options Register Description

The DCP Channel 3 Options Register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH3OPTS	0x800281F0
HW_DCP_CH3OPTS_SET	0x800281F4
HW_DCP_CH3OPTS_CLR	0x800281F8
HW_DCP_CH3OPTS_TOG	0x800281FC

Table 709. HW_DCP_CH3OPTS

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

Table 710. HW_DCP_CH3OPTS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	RSVD	RO	0x0	Reserved.
15:0	RECOVERY_TIMER	RW	0x0	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initialized with this value and then decremented until the timer reaches 0. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0 ns to 8.3 ms at 133-MHz operation.

DESCRIPTION:

The DCP Channel 3 Options Register can be used to control optional features of the channels.

15.3.32. Color-Space Conversion Control Register 0 Description

The Color-Space Conversion Control Register 0 contains several control bits that control the color-space conversion logic.

HW_DCP_CSCCTRL0	0x80028300
HW_DCP_CSCCTRL0_SET	0x80028304
HW_DCP_CSCCTRL0_CLR	0x80028308
HW_DCP_CSCCTRL0_TOG	0x8002830C

Table 711. HW_DCP_CSCCTRL0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD																	UPSAMPLE	SCALE	ROTATE	SUBSAMPLE	DELTA	RGB_FORMAT	YUV_FORMAT			RSVD			ENABLE		

Table 712. HW_DCP_CSCCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:15	RSVD	RO	0x00000	Reserved.
14	UPSAMPLE	RW	0x0	Indicates that the color conversion process should first upsample the subsampled chroma values at the luma locations before performing the color conversion. Note that the implementation only upsamples in the X (horizontal) direction since the CSC does not load multiple lines of chroma data. The overall effect of enabling this bit is to smooth color transitions in the horizontal direction.
13	SCALE	RW	0x0	Indicates that the output image should be scaled. The CSCXSCALE and CSCYSCALE registers should be programmed accordingly. When this bit is 0, the contents of the scaling registers are ignored.
12	ROTATE	RW	0x0	Indicates that the output image should be rotated/flipped. Rotation is defined as a clockwise 90 degree rotation with horizontal flip, which results in the image being mirrored about the top-left/bottom-right diagonal.
11	SUBSAMPLE	RW	0x0	Indicates that pixel subsampling should be enabled. Pixel sub-sampling should only be enabled when the DELTA bit is set. Enabling this bit causes the CSC to resample the green color component based on its actual position in a Delta display module.
10	DELTA	RW	0x0	Indicates framebuffer color ordering should correspond to delta pixel configuration. This should only be enabled for 24-bit RGB frame buffer configurations.
9:8	RGB_FORMAT	RW	0x0	Target RGB framebuffer format. RGB16_565 = 0x0 RGB24 = 0x2 YUV422I = 0x3
7:4	YUV_FORMAT	RW	0x0	YUV Input Buffer format. YUV420 = 0x0 YUV422 = 0x2
3:1	RSVD	RO	0x00	Reserved.
0	ENABLE	RW	0x0	Enables color-space conversion with specified parameters.

STMP3770

DESCRIPTION:

The Color-Space Conversion Control Register 0 contains the primary controls for the CSC block. All other CSC registers should be programmed before writing the ENABLE bit to a 1. At the completion of a CSC operation, the CSC interrupts and provides status in the Color-Space Conversion Status Register.

15.3.33. Color-Space Conversion Status Register Description

The Color-Space Conversion Status Register contains CSC status information.

HW_DCP_CSCSTAT	0x80028310
HW_DCP_CSCSTAT_SET	0x80028314
HW_DCP_CSCSTAT_CLR	0x80028318
HW_DCP_CSCSTAT_TOG	0x8002831C

Table 713. HW_DCP_CSCSTAT

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

Table 714. HW_DCP_CSCSTAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x00	Reserved.
23:16	ERROR_CODE	RW	0x0	Indicates additional error codes for some error conditions. LUMA0_FETCH_ERROR_Y0 = 0x01 Error fetching from Luma Y0 buffer LUMA1_FETCH_ERROR_Y1 = 0x02 Error fetching from Luma Y1 buffer CHROMA_FETCH_ERROR_U = 0x03 Error fetching from Chroma (U) buffer CHROMA_FETCH_ERROR_V = 0x04 Error fetching from Chroma (V) buffer
15:6	RSVD	RO	0x00	Reserved.
5	ERROR_DST	RW	0x0	This bit indicates a bus error occurred when writing the output RGB buffer. When an error is detected, the CSC operation terminates.
4	ERROR_SRC	RW	0x0	This bit indicates a bus error occurred when reading from a source buffer. When an error is detected, the CSC operation terminates. See the Error Code field to identify the operation that failed.
3	RSVD	RO	0x0	Reserved.
2	ERROR_SETUP	RW	0x0	This bit indicates that the hardware has detected an invalid programming configuration. See the Error Code field to identify the cause of this error.
1	RSVD	RO	0x0	Reserved.
0	COMPLETE	RW	0x0	When set, this bit indicates that the CSC has completed its operation.

DESCRIPTION:

The Color-Space Conversion Status Register provides status for the color-space converter logic.

15.3.34. Color-Space Conversion Output Buffer Parameters Register Description

The Color-Space Conversion Output Buffer Parameters Register contains frame-buffer size information for the output RGB/YUV buffer.

HW_DCP_CSCOUTBUFPARAM 0x80028320

Table 715. HW_DCP_CSCOUTBUFPARAM

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

Table 716. HW_DCP_CSCOUTBUFPARAM Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x000000	Reserved.
23:12	FIELD_SIZE	RW	0x0	Indicates size of field as number of lines (vertical height of display) for unscaled images. When scaling is enabled, this value should be set to the height of the input buffer.
11:0	LINE_SIZE	RW	0x0	Indicates size of line in terms of horizontal pixels (horizontal width of display)

DESCRIPTION:

The Color-Space Conversion Output Buffer Parameters Register contains the framebuffer parameters for the output frame buffer. The values specified here should match the resolution of the LCD display in use. When scaling, the output site is taken from the TSCALE and YSCALE registers, so the LINE_SIZE bit field is unused. The FIELD_SIZE bit field, however, is still used to determine the site of the input buffer in the vertical direction.

Clipping may be achieved by setting the LINE_SIZE of the CSCOUTBUFPARAM register to a value less than the LINE_SIZE in the CSCINBUFPARAM register.

15.3.35. Color-Space Conversion Input Buffer Parameters Register Description

The Color-Space Conversion Input Buffer Parameters Register contains framebuffer size information for the input YUV/YCbCr buffer.

HW_DCP_CSCINBUFPARAM 0x80028330

Table 717. HW_DCP_CSCINBUFPARAM

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

Table 718. HW_DCP_CSCINBUFPARAM Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x000000	Reserved.
11:0	LINE_SIZE	RW	0x0	Indicates size of line in terms of horizontal pixels (horizontal width of display)

DESCRIPTION:

The Color-Space Conversion Input Buffer Parameters Register contains the frame-buffer parameters for the input YUV/YCbCr buffer. The field size is assumed to be the same as for the output buffer.

Vertical clipping can be achieved by setting the YCbCr pointers to a non-zero line in the framebuffer and reducing the FIELD_SIZE parameter in the CSCOUTBUFPARAM register.

Horizontal clipping can be done by offsetting the YCbCr pointers to a non-zero offset within the line of the input frame buffer and setting the LINE_SIZE field of the CSCOUTBUFPARAM register to a value smaller than the actual YUV source data line size.

15.3.36. Color-Space RGB Frame Buffer Pointer Description

The Color-Space RGB Frame Buffer Pointer points to the beginning of the RGB output frame buffer.

HW_DCP_CSCRGB 0x80028340

Table 719. HW_DCP_CSCRGB

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

Table 720. HW_DCP_CSCRGB Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x0	Current address pointer for the output frame buffer (this is a working register and will update as the CSC proceeds. The address MUST be word-aligned for proper CSC operation.

DESCRIPTION:

The Color-Space RGB Frame Buffer Pointer points to the current output location for the RGB frame buffer. This is a working register, so as the conversion proceeds, this

register will reflect the actual address within the frame buffer that the CSC is working on. Writes to this register are disabled once the ENABLE field of the Control 0 register is set.

15.3.37. Color-Space Luma (Y) Buffer Pointer Description

The Color-Space Luma (Y) Buffer Pointer points to the beginning of the Luminance input buffer.

HW_DCP_CSCLUMA 0x80028350

Table 721. HW_DCP_CSCLUMA

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

Table 722. HW_DCP_CSCLUMA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x0	Current address pointer for the input luminance (Y) buffer (this is a working register and will update as the CSC proceeds). The address MUST be word-aligned for proper CSC operation.

DESCRIPTION:

The Color-Space Luma (Y) Buffer Pointer points to the current input location for the luminance (Y) buffer. This is a working register, so as the conversion proceeds, this register will reflect the actual address within the frame buffer that the CSC is working on. Writes to this register are disabled once the ENABLE field of the Control 0 register is set.

15.3.38. Color-Space Chroma (U/Cb) Buffer Pointer Description

The Color-Space Chroma (U/Cb) Buffer Pointer points to the beginning of the Chrominance U/Cb input buffer.

HW_DCP_CSCCHROMAU 0x80028360

Table 723. HW_DCP_CSCCHROMAU

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

Table 724. HW_DCP_CSCCHROMAU Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x0	Current address pointer for the input chrominance (U/Cb) buffer (this is a working register and will update as the CSC proceeds). The address MUST be word-aligned for proper CSC operation.

DESCRIPTION:

Color-Space Chroma (U/Cb) Buffer Pointer points to the current input location for the chrominance U/Cb buffer. This is a working register, so as the conversion pro-

STMP3770

ceeds, this register will reflect the actual address within the frame buffer that the CSC is working on. Writes to this register are disabled once the ENABLE field of the Control 0 register is set.

15.3.39. Color-Space Chroma (V/Cr) Buffer Pointer Description

The Color-Space Chroma (V/Cr) Buffer Pointer points to the beginning of the Chrominance V/Cr/m input buffer.

HW_DCP_CSCCHROMAV 0x80028370

Table 725. HW_DCP_CSCCHROMAV

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

Table 726. HW_DCP_CSCCHROMAV Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	ADDR	RW	0x0	Current address pointer for the input chrominance (V/Cr) buffer (this is a working register and will update as the CSC proceeds). The address MUST be word-aligned for proper CSC operation.

DESCRIPTION:

The Color-Space Chroma (V/Cr) Buffer Pointer points to the current input location for the chrominance V/Cr buffer. This is a working register, so as the conversion proceeds, this register will reflect the actual address within the frame buffer that the CSC is working on. Writes to this register are disabled once the ENABLE field of the Control 0 register is set.

15.3.40. Color-Space Conversion Coefficient Register 0 Description

The Color-Space Conversion Coefficient Register 0 contains color-space conversion coefficients.

HW_DCP_CSCCOEFF0 0x80028380

Table 727. HW_DCP_CSCCOEFF0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD								C0								UV_OFFSET								Y_OFFSET							

Table 728. HW_DCP_CSCCOEFF0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x00	Reserved.
25:16	C0	RW	0x12A	Y multiplier coefficient

Table 728. HW_DCP_CSCCOEFF0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	UV_OFFSET	RW	0x80	Indicates the phase offset implicit for UV data (typically 128 or 0x80 to indicate a -0.5 to 0.5 range)
7:0	Y_OFFSET	RW	0x10	Indicates the amplitude offset implicit in the Y data. For YUV, this is typically 0 and for YCbCr, this is typically 16 (0x10)

DESCRIPTION:

The Color-Space Conversion Coefficient Register 0 contains coefficients used in the color-space conversion algorithm. The Y and UV offsets are subtracted from the source buffer to normalize them before the conversion. C0 is the coefficient that is used to multiply the luma component of the data for all three RGB components.

15.3.41. Color-Space Conversion Coefficient Register 1 Description

The Color-Space Conversion Coefficient Register 1 contains color-space conversion coefficients

HW_DCP_CSCCOEFF1 0x80028390

Table 729. HW_DCP_CSCCOEFF1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD				C1								RSVD				C4															

Table 730. HW_DCP_CSCCOEFF1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x00	Reserved.
25:16	C1	RW	0x198	Red Cr multiplier coefficient
15:10	RSVD	RO	0x00	Reserved.
9:0	C4	RW	0x204	Blue Cb multiplier coefficient

DESCRIPTION:

The Color-Space Conversion Coefficient Register 1 contains coefficients used in the color-space conversion algorithm.

C1 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the red component.

C4 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the blue component.

Both values should be coded as an unsigned fixed point number with 8 bits right of the decimal.

15.3.42. Color-Space Conversion Coefficient Register 2 Description

The Color-Space Conversion Coefficient Register 2 contains color-space conversion coefficients.

HW_DCP_CSCCOEFF2 0x800283A0

Table 731. HW_DCP_CSCCOEFF2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RSVD				C2								RSVD				C3															

Table 732. HW_DCP_CSCCOEFF2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x00	Reserved.
25:16	C2	RW	0x0D0	Green Cr multiplier coefficient
15:10	RSVD	RO	0x00	Reserved.
9:0	C3	RW	0x064	Green Cb multiplier coefficient

DESCRIPTION:

The Color-Space Conversion Coefficient Register 2 contains coefficients used in the color-space conversion algorithm.

C2 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the green component.

C3 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the green component.

Both values should be coded as an unsigned fixed point number with 8 bits right of the decimal.

15.3.43. Color-Space Conversion X-Scaling Register Description

The Color-Space Conversion X-Scaling Register contains controls for video scaling in the X-direction. Software should provide the scaled output width along with the required INT/FAC values.

HW_DCP_CSCXSCALE 0x800283E0

Table 733. HW_DCP_CSCXSCALE

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

Table 734. HW_DCP_CSCXSCALE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x00	Reserved.
25:24	INT	RW	0x0	Integer coefficient. This should be set to (source_width / target_width). For upscaling, this value should be 0. For downscaling, it should be 1 or 2. Downscaling greater than 2 not supported.
23:12	FRAC	RW	0x000	Fractional coefficient. This should be set to (source_width % target_width)
11:0	WIDTH	RW	0x000	Scaled video width

DESCRIPTION:

The Color-Space Conversion X-Scaling Register can be used to scale color-space converted video to a new resolution. This register is used in conjunction with the Y-Scaling register and is enabled by the SCALE bit in the CSC control register.

15.3.44. Color-Space Conversion Y-Scaling Register Description

The Color-Space Conversion Y-Scaling Register contains controls for video scaling in the Y-direction.

HW_DCP_CSCYSCALE 0x800283F0

Table 735. HW_DCP_CSCYSCALE

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

Table 736. HW_DCP_CSCYSCALE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x00	Reserved.
25:24	INT	RW	0x0	Integer coefficient. This should be set to (source_height / target_height). For upscaling, this value should be 0. For downscaling, it should be 1 or 2. Downscaling greater than 2 not supported.
23:12	FRAC	RW	0x000	Fractional coefficient. This should be set to (source_height % target_height)
11:0	HEIGHT	RW	0x000	Scaled video height

DESCRIPTION:

The Color-Space Conversion Y-Scaling Register can be used to scale color-space converted video to a new resolution. This register is used in conjunction with the X-Scaling register and is enabled by the SCALE bit in the CSC control register.

STMP3770

15.3.45. DCP Debug Select Register Description

The DCP Debug Select Register selects a debug register to view.

HW_DCP_DBGSELECT 0x80028400

Table 737. HW_DCP_DBGSELECT

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

Table 738. HW_DCP_DBGSELECT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:8	RSVD	RO	0x00	Reserved.
7:0	INDEX	RW	0x0	Selects a value to read via the debug data register. CONTROL = 0x01 OTPKEY0 = 0x10 OTPKEY1 = 0x11 OTPKEY2 = 0x12 OTPKEY3 = 0x13

DESCRIPTION:

The DCP Debug Select Register selects debug information to return in the debug data register.

15.3.46. DCP Debug Data Register Description

Reading the DCP Debug Data Register returns the debug data value from the selected index.

HW_DCP_DBGDATA 0x80028410

Table 739. HW_DCP_DBGDATA

31	30	29	28	27	26	25	24	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 740. HW_DCP_DBGDATA Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	DATA	RO	0x0	Debug Data

DESCRIPTION:

The DCP Debug Data Register returns the debug data from the selected debug index source.

15.3.47. DCP Version Register Description

The DCP Version Register is a read-only register indicating the implemented version of the DCP.

HW_DCP_VERSION

0x80028420

Table 741. HW_DCP_VERSION

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

Table 742. HW_DCP_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x1	Fixed read-only value reflecting the MAJOR version of the design implementation.
23:16	MINOR	RO	0x0	Fixed read-only value reflecting the MINOR version of the design implementation.
15:0	STEP	RO	0x0	Fixed read-only value reflecting the stepping version of the design implementation.

DCP Block v1.0

The license for the AEC code is documented here for compliance:

Copyright (C) 2000-2003, ASICS World Services, LTD., AUTHORS

All rights reserved. Redistribution and use in source, netlist, binary and silicon forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of ASICS World Services, the Authors and/or the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

STMP3770



16. SYNCHRONOUS SERIAL PORTS (SSP)

This chapter describes the two identical synchronous serial ports (SSP) included on the STMP3770. It includes sections on external pins, bit rate generation, frame formats, Motorola SPI mode, Texas Instruments Synchronous Serial Interface (SSI) mode, SD/SDIO/MMC mode, CE-ATA mode, and MS mode. Programmable registers are described in [Section 16.11](#).

16.1. 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; 1-bit, 4-bit, and 8-bit SD/SDIO/MMC; CE-ATA mode; and 1-bit and 4-bit MS modes. The SPI mode has enhancements to support 1-bit legacy MMC cards. The SSP also supports slave operation for the SPI and SSI modes. The SSP has a dedicated DMA channel in the bridge and can also be controlled directly by the CPU through PIO registers. [Figure 61](#) illustrates one of the two SSP ports included on the STMP3770. The only interaction between SSP1 and SSP2 is that they share the same input, SSPCLK.

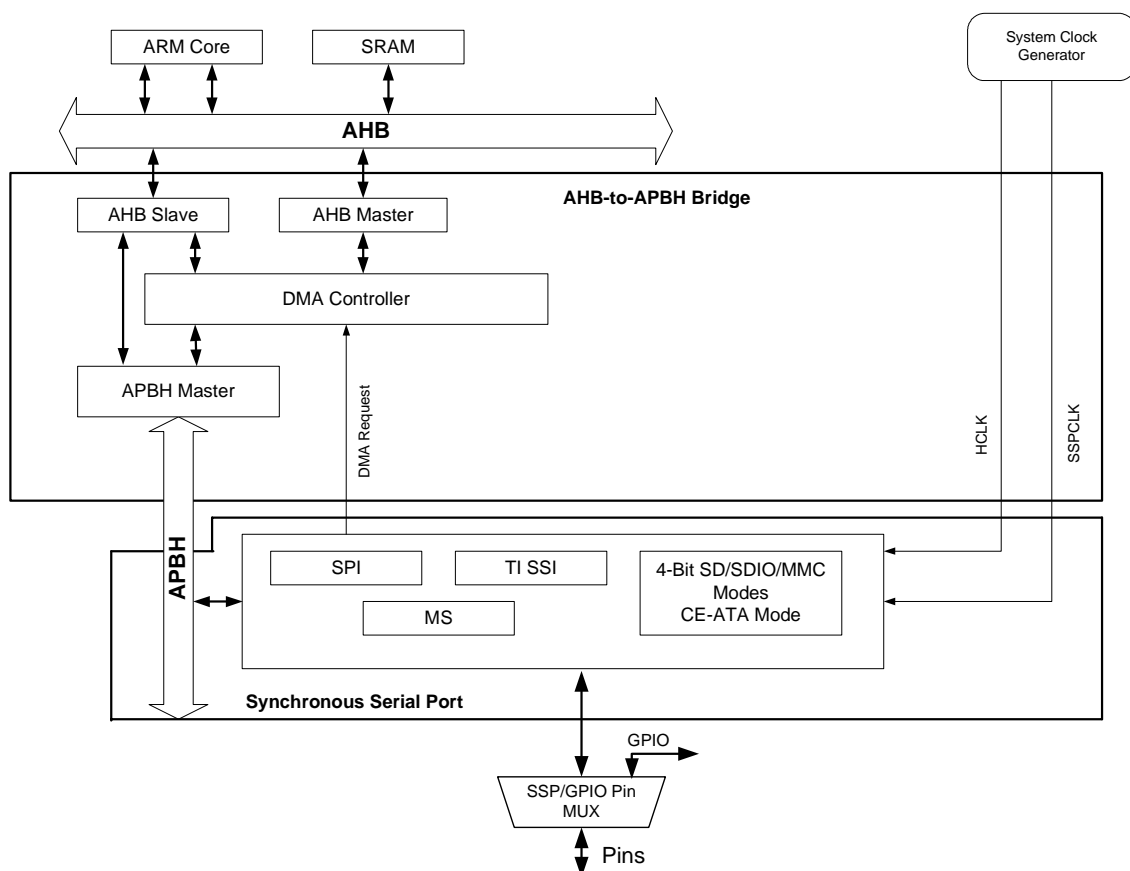


Figure 61. Synchronous Serial Port Block Diagram

16.2. External Pins

Table 743 lists the SSP pin placements for all supported modes.

Table 743. SSP Pin Matrix

PIN NAME	SPI MODE	TI SSI MODE	SD/SDIO/MMC/CE-ATA MODES	MS MODE
SSP_SCK	SCK	CLK	CLK	CLK
SSP_CMD	MOSI	MOSI	CMD	BS
SSP_DATA0	MISO	MISO	DATA0	DATA0
SSP_DATA1			DATA1/IRQ	DATA1
SSP_DATA2			DATA2	DATA2
SSP_DATA3	SSn	SSn	DATA3	DATA3
SSP_DETECT			CARD_DETECT	
SSP_D[7–4]			DATA7–DATA4	

The pin control interface on the STMP3770 provides all digital pins with selectable output drive strengths. In addition, all SSP data pins have selectable 47-K Ω pullup resistors, and SSP command pins have 10-K Ω pullups. Configuring the SSP_CMD pad to connect to the internal 10-K Ω pullup is recommended for SD/SDIO/MMC/CE-ATA modes during the Card_ID phase. After the Card_ID phase, the 10-K Ω pullup should be disabled, and the weaker external 47-K Ω pullup takes over. The SSP_DATA pads also can be configured to connect to an internal 47-K Ω pullup, which is required for SD/SDIO/MMC/CE-ATA modes. See Table 1205, “HW_PINCTRL_PULL0,” on page 987 and Table 1207, “HW_PINCTRL_PULL1,” on page 988 for configuration details.

16.3. 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, CLOCK_DIVIDE, from 2 to 254, which is programmed in HW_SSP_TIMING. The clock is further divided by a value from 1 to 256, which is 1 + CLOCK_RATE, where CLOCK_RATE is the value programmed in HW_SSP_TIMING.

The frequency of the output signal bit clock SSP_SCK is defined as follows:

$$SSP_SCK = \frac{SSPCLK}{CLOCK_DIVIDE * (1 + CLOCK_RATE)}$$

For example, if SSPCLK is 3.6864 MHz, and CLOCK_DIVIDE = 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 51, for more clock details.

16.4. Frame Format for SPI and SSI

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. Two basic frame types can be selected:

- Motorola SPI
- Texas Instruments Synchronous Serial Interface (SSI)

For both 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 time-out indication, which occurs when the receive FIFO still contains data after a time-out period.

For Motorola SPI frame format, 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.

16.5. Motorola SPI Mode

The SPI mode is used for general inter-component communication and legacy 1-bit MMC cards.

16.5.1. SPI DMA Mode

The SPI bus is inherently a full-duplex bidirectional interface. However, as most applications only require half-duplex data transmission, the STMP3770 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.

16.5.2. 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.

16.5.2.1. 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.

16.5.2.2. 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.

16.5.3. 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 62](#) and [Figure 63](#).

STMP3770

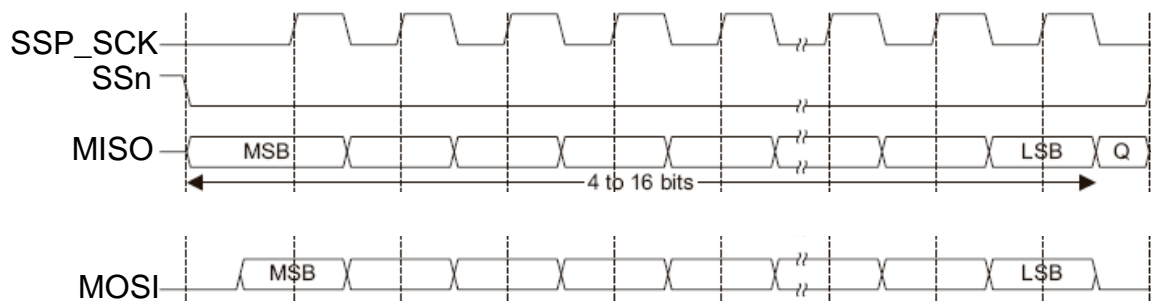


Figure 62. Motorola SPI Frame Format (Single Transfer) with POLARITY=0 and PHASE=0

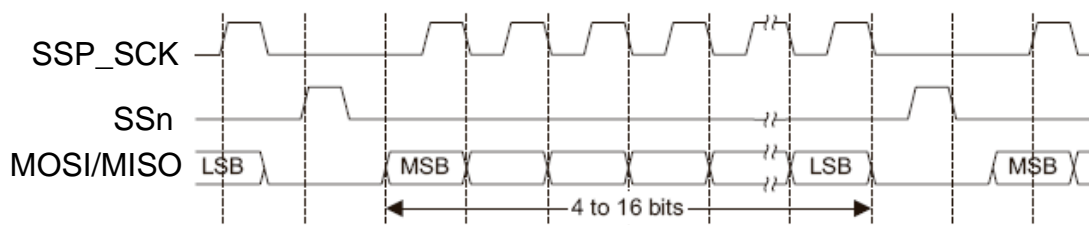


Figure 63. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=0

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 0. 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.

16.5.4. 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 64, which covers both single and continuous transfers.

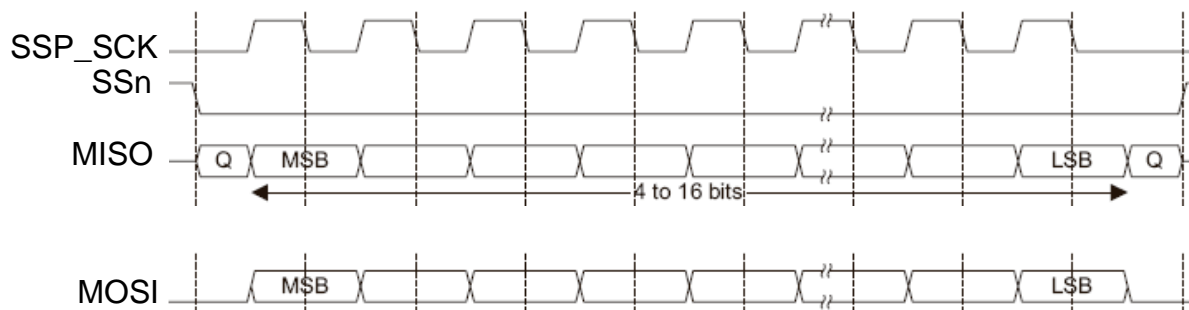


Figure 64. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=1

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, SSPFSOUT (the SSn pin in master mode) is held low between successive data words and termination is the same as that of a single word transfer.

16.5.5. 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 65 and Figure 66.

STMP3770

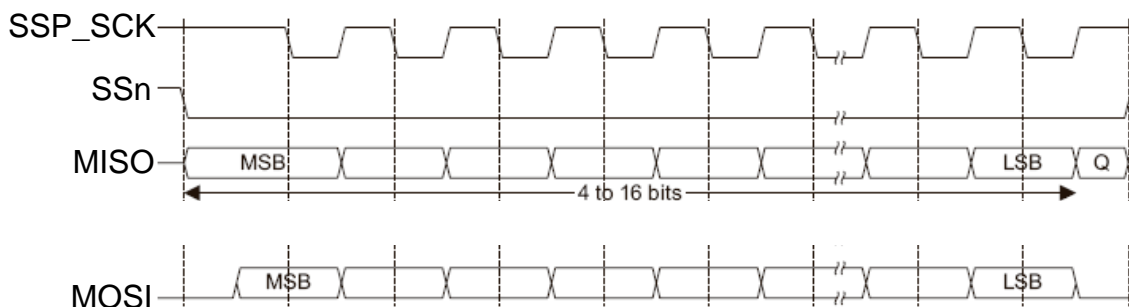


Figure 65. Motorola SPI Frame Format (Single Transfer) with POLARITY=1 and PHASE=0

Note: In Figure 65, Q is an undefined signal.

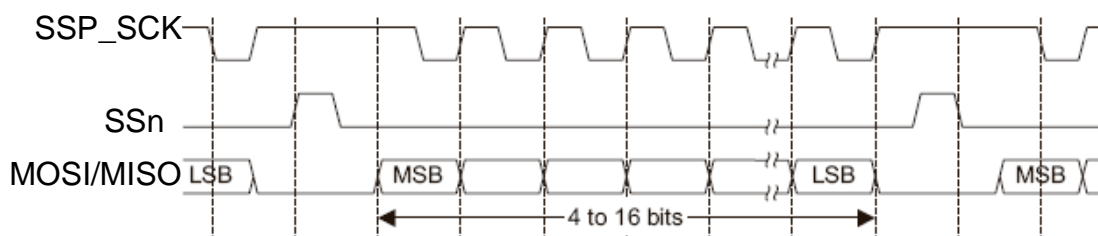


Figure 66. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=1 and PHASE=0

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 0. Therefore, the master device must raise SSPSFSSIN (the SSn pin in slave mode) 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.

16.5.6. 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 67, which covers both single and continuous transfers.

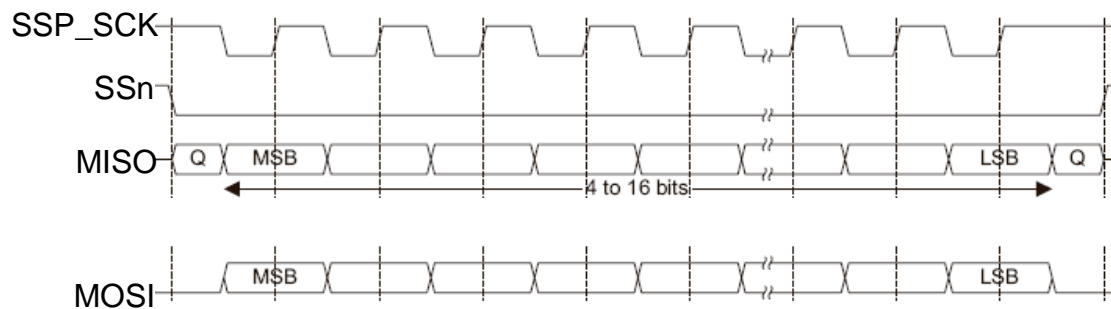


Figure 67. Motorola SPI Frame Format with POLARITY=1 and PHASE=1

Note: In Figure 67, 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.

16.6. Texas Instruments Synchronous Serial Interface (SSI) Mode

Figure 68 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

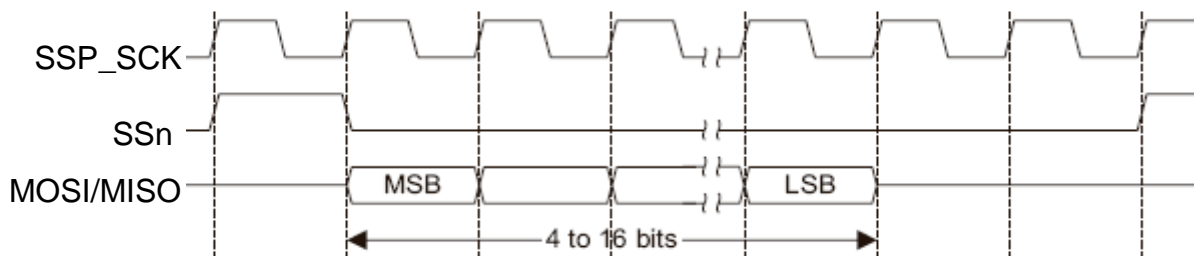


Figure 68. Texas Instruments Synchronous Serial Frame Format (Single Transfer)

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 69 shows the Texas Instrument synchronous serial frame format when back-to-back frames are transmitted.

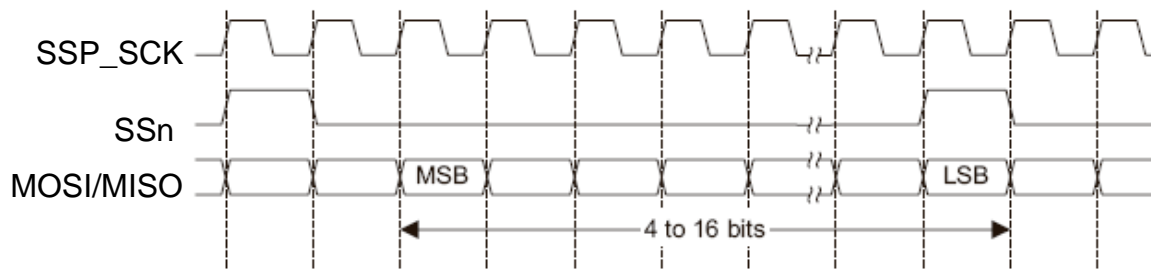


Figure 69. Texas Instruments Synchronous Serial Frame Format (Continuous Transfer)

16.7. SD/SDIO/MMC Mode

This mode is used to provide high performance with SD, SDIO, MMC, and high-speed (4-bit and 8-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 time-out)
- Transfer multiple blocks 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. A single DMA descriptor can also perform multiple card block transfers.

16.7.1. SD/MMC Command/Response Transfer

SD/MMC commands are written to the HW_SSP_CMDn 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 744](#).

Table 744. 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-impedance 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 time-out, 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 745](#) and [Table 746](#):

Table 745. SD/MMC Command Regular Response Token

Line	Start	Source	Data	CRC	End
CMD	0	0 (Card)	38-bit Response	CRC7	1

Table 746. SD/MMC Command Regular Long Response Token

Line	Start	Source	Data	CRC	End
CMD	0	0 (Card)	117-bit response	CRC16	1

16.7.2. 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 747](#). Block data transfers typically have 512 bytes of payload, 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 70](#) 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 747](#).

Table 747. 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 748](#).

Table 748. 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

In 8-bit bus mode, the block data is formatted as shown in [Table 749](#).

Table 749. SD/MMC Data Block Transfer 8-Bit Bus Mode

Line	Start	Data	Data	Data	CRC	End
DATA7	0	Data Bit 7 Byte 0	...	Data Bit 7 Byte 511	CRC16	1
DATA6	0	Data Bit 6 Byte 0	...	Data Bit 6 Byte 511	CRC16	1
DATA5	0	Data Bit 5 Byte 0	...	Data Bit 5 Byte 511	CRC16	1
DATA4	0	Data Bit 4 Byte 0	...	Data Bit 4 Byte 511	CRC16	1
DATA3	0	Data Bit 3 Byte 0	...	Data Bit 3 Byte 511	CRC16	1
DATA2	0	Data Bit 2 Byte 0	...	Data Bit 2 Byte 511	CRC16	1
DATA1	0	Data Bit 1 Byte 0	...	Data Bit 1 Byte 511	CRC16	1
DATA0	0	Data Bit 0 Byte 0	...	Data Bit 0 Byte 511	CRC16	1

16.7.2.1. 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. Multiple blocks can also be transferred with a single DMA descriptor.

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.

The SSP supports transferring multiple SD/MMC blocks per DMA descriptor. The SSP state machine needs to know the number of blocks and the size of a block. When `HW_SSP_CMD0_BLOCK_COUNT` is non-zero, the actual block size is:

0x1 shiftleft BLOCK_SIZE

For example, setting a value of 9 will result in a block size of 512 bytes. When `BLOCK_COUNT` is 0, `BLOCK_SIZE` is ignored and the bit field `HW_SSP_CTRL0_XFER_COUNT` represents the single block size or the number of byte to transfer. This must satisfy equation:

$HW_SSP_CTRL0_XFER_COUNT = (0x1 \text{ shiftleft } BLOCK_SIZE) \times (BLOCK_COUNT + 1)$

for `BLOCK_COUNT` greater than 0.

16.7.2.2. 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.

16.7.3. SDIO Interrupts

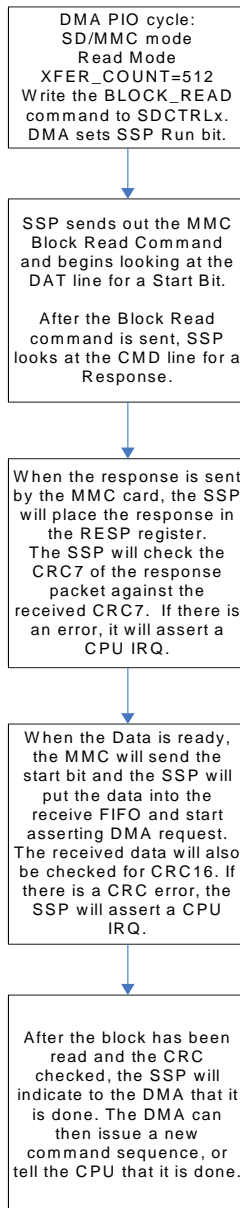
The SSP supports SDIO interrupts. When the SSP is in SD/MMC mode and the SDIO_IRQ bit in the HW_SSP_CTRL0 register is set, the SSP looks for interrupts on DATA1 during the valid IRQ periods. The valid IRQ periods are defined in the SDIO specification. If the card asserts an interrupt and SDIO_IRQ_EN 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.

16.7.4. 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_EN 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.

SD/MMC Block Read Example



SD/MMC Block Write Example

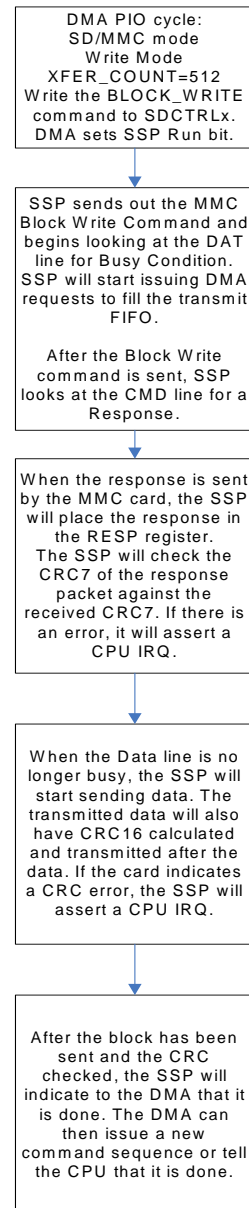


Figure 70. SD/MMC Block Transfer Flowchart

- Data Time-Out Error**—The SSP TIMEOUT counter is used to detect a time-out condition during data write or read operations. The time-out 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 time-out 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 time-out 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 FIFO_OVRFLW or FIFO_UNDRFLW status bit is set in the SSP Status Register and asserts a CPU IRQ.
- **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 Time-Out**—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_TIME-OUT status flag, and asserts the RESP_TIME-OUT_IRQ. The ISR should read the status register to find that a command response time-out has occurred. It can then decide to reset the DMA channel and SSP and re-try the operation.

16.7.5. 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.

16.8. CE-ATA Mode

CE-ATA devices can be accessed via SD/MMC interface. In addition to requiring the SD/MMC interface, Command Completion Signaling (CCS) and CCS Disable is needed. Refer to the CE-ATA specification for timing diagrams and more details on these signals. The MMC state machine includes extra states to handle this signaling. If the SDIO_IRQ_CHECK (WAIT_FOR_CCS) bit is set in HW_SSP_CTRL0 register, then the MMC state machine enters the CCS state after the SD/MMC response, or if there is a data transfer phase, then CCS state will be entered after the all the data is transferred.

If the LOCK_CS (DISABLE_CCS) bit is set in HW_SSP_CTRL0, then the CCSD state is entered after the SD/MMC response is received. A 4-bit CCSD counter is used to generate the timing for CCSD. The counter is initialized to 0 and increments

when in the CCSD state. When the counter reaches 8, '0' is driven on the SSPCMD output. When the counter reaches 12, '1' is driven out. A data transfer (HW_SSP_CTRL0_DATA_XFER=1) must accompany the DISABLE_CCS request to prevent the MMC state machine going idle before the signaling is complete. Also, the SSP actions are undefined when both DISABLE_CCS and WAIT_FOR_CCS are simultaneously set during a command.

16.9. MS Mode

The SSP MS mode supports 1-bit and 4-bit MS data transfers. The SSP MS mode is designed to work with the STMP3770 DMA controller. The DMA controller's linked descriptor architecture provides a high level of automatic operation without CPU intervention.

16.9.1. MS Mode I/O Pins

The MS standard defines three pins:

- **SDIO/DATA0**—A bidirectional data pin used for commands and data in serial and parallel interface.
- **DATA[3–1]**—A bidirectional parallel interface data bus 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.

16.9.2. 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 71](#). 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.

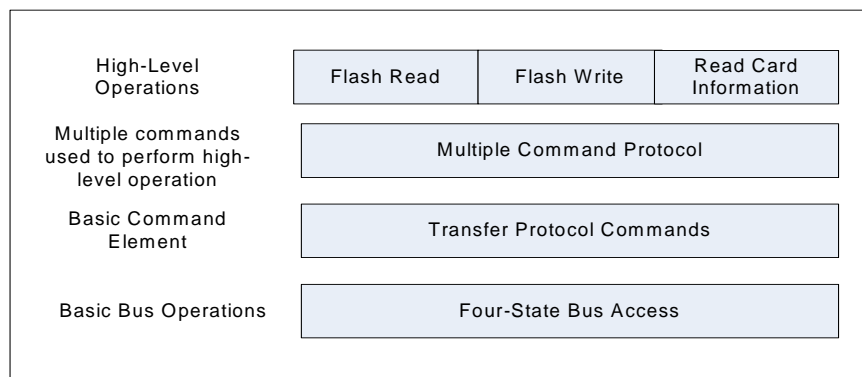


Figure 71. Basic MS Protocols

16.9.3. *MS Mode High-Level Operation*

The STMP3770 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 72](#).

16.9.4. *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 73](#).

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.

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/DATA0 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.

16.9.5. *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 16.9.6](#).

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/DATA line and count for time-out. If the time-out 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.

16.9.6. Checking Card Status

In MS mode, card status and flash page data are transferred over the SDIO/DATA3-7 lines. 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.

16.9.7. 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.

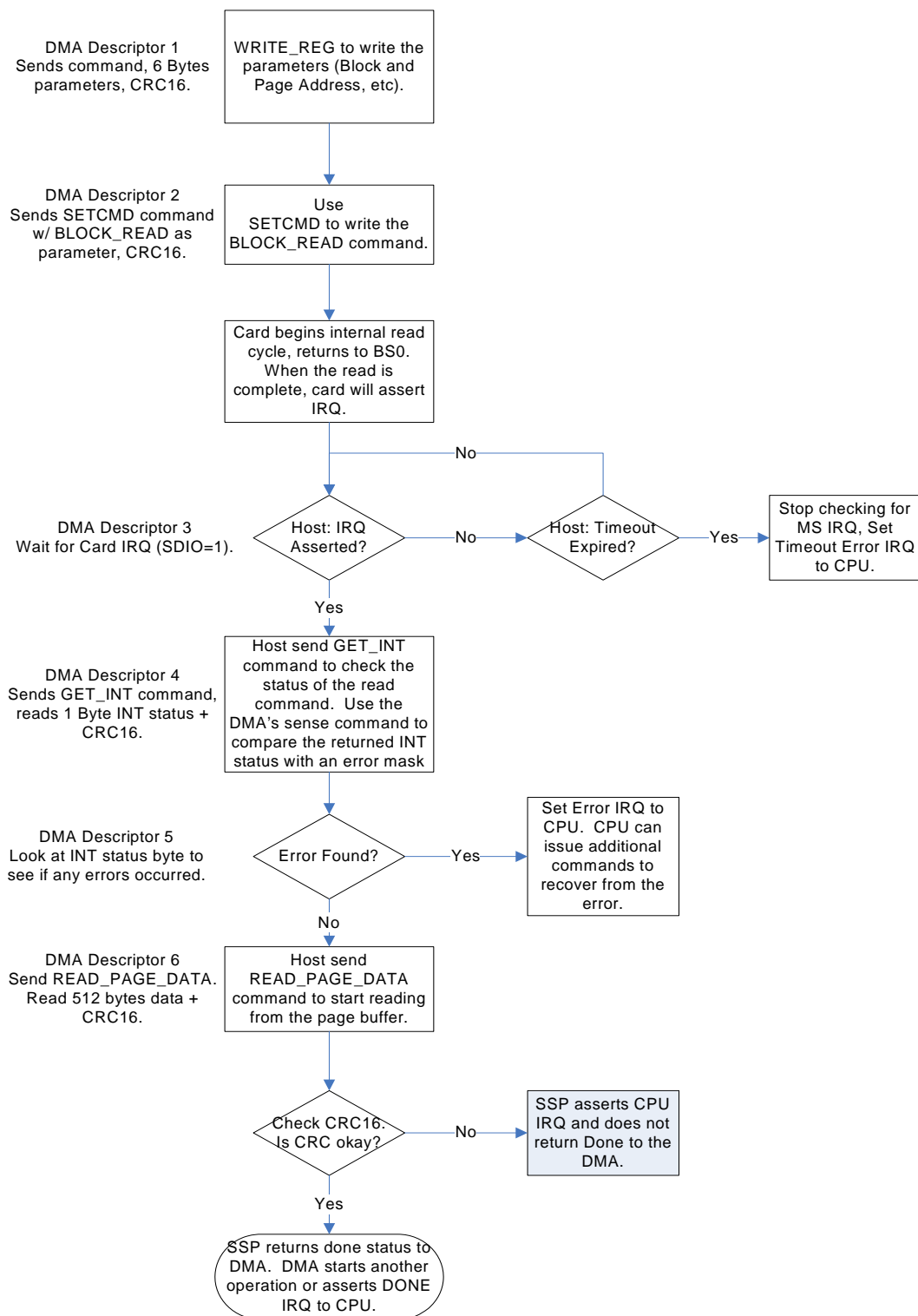


Figure 72. MS Operation Flowchart

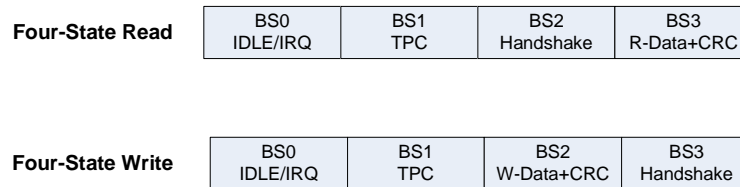


Figure 73. MS Four-State Read and Write

- **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 time-out in the SSP waiting for the handshake to complete during BS3.
- **Data Time-Out Error**—In MS mode, the time-out counter is used while waiting for an IRQ in BS0. If the time-out 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 time-out 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.

16.9.8. 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 more than 16 SCK cycles have passed before the ready handshake has been received, the SSP times out. This results in a RESP_TIMEOUT_IRQ to the CPU.

16.10. 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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

Table 751. HW_SSP_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
27	LOCK_CS	RW	0x0	<p>In SPI mode: This bit 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.</p> <p>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.</p> <p>In CE_ATA mode: 1= Disable/Cancel CCS (Command Completion Signal) from card. In CE_ATA mode, if this bit is set, then the DATA_XFER bit must also be set.</p>
26	IGNORE_CRC	RW	0x0	Ignore CRC. In SD/MMC and MS modes, ignores the Response CRC.
25	READ	RW	0x0	Read Mode. When this and DATA_XFER are set, the SSP reads data from the device. If this bit is not set, then the SSP writes data to the device.
24	DATA_XFER	RW	0x0	<p>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).</p> <p>In MS mode, this bit selects the destination of read transfers and source for write transfers: set to 1 if using Rx/TxFIFOs, and set to 0 if using Resp0/Cmd1 registers.</p> <p>When set to 0 in MS mode, the XFER_COUNT field must be 4 or less.</p>
23:22	BUS_WIDTH	RW	0x0	<p>Data Bus Width. SD/MMC and CE_ATA modes support all widths; MS mode supports 1-bit and 4-bit mode; and the other modes (SPI) supports 1-bit mode.</p> <p>ONE_BIT = 0x0 data bus is 1 bit wide. FOUR_BIT = 0x1 data bus is 4 bits wide. EIGHT_BIT = 0x2 data bus is 8 bits wide.</p>
21	WAIT_FOR_IRQ	RW	0x0	<p>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.</p> <p>In SD/MMC mode, this signal means wait for MMC ready before sending command. (MMC is busy when databit 0 is low.)</p>

STMP3770**Table 751. HW_SSP_CTRL0 Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
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 and 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 and MS modes) 0 = Do not wait for a response from the card. 1 = This command should receive a response from the card.
16	ENABLE	RW	0x0	Command Transmit Enable. (SD/MMC and 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.
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.

16.11.2. SD/MMC and MS Command Register 0 Description

HW_SSP_CMD0	0x80010010
HW_SSP_CMD0_SET	0x80010014
HW_SSP_CMD0_CLR	0x80010018
HW_SSP_CMD0_TOG	0x8001001C

STMP3770**Table 753. HW_SSP_CMD0 Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
15:8	BLOCK_COUNT	RW	0x0	SD/MMC block count. This value is one less than the number of blocks to transfer. For example, setting a value of 1 will transfer two blocks. This must satisfy the equation: $\text{HW_SSP_CTRL0_XFER_COUNT} = (1 \text{ shiftleft } \text{BLOCK_SIZE}) \times (\text{BLOCK_COUNT} + 1)$ for BLOCK_COUNT greater than 0. This is also SPI/SSI control word[15:8] for RX transfers.

Table 753. HW_SSP_CMD0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:0	CMD	RW	0x0	<p>SD/MMC Command Index (uses 5:0) or MS TPC [7:0] to be sent to card. This is also SPI/SSI control word[7:0] for RX transfers.</p> <p>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</p>

Table 762. HW_SSP_CTRL1

SDIO_IRQ	3	1
SDIO_IRQ_EN	3	0
RESP_ERR_IRQ	2	9
RESP_ERR_IRQ_EN	2	8
RESP_TIMEOUT_IRQ	2	7
RESP_TIMEOUT_IRQ_EN	2	6
DATA_TIMEOUT_IRQ	2	5
DATA_TIMEOUT_IRQ_EN	2	4
DATA_CRC_IRQ	2	3
DATA_CRC_IRQ_EN	2	2
FIFO_UNDERRUN_IRQ	2	1
FIFO_UNDERRUN_EN	2	0
CEATA_CCS_ERR_IRQ	1	9
CEATA_CCS_ERR_IRQ_EN	1	8
RCV_TIMEOUT_IRQ	1	7
RCV_TIMEOUT_IRQ_EN	1	6
FIFO_OVERRUN_IRQ	1	5
FIFO_OVERRUN_IRQ_EN	1	4
DMA_ENABLE	1	3
CEATA_CCS_ERR_EN	1	2
SLAVE_OUT_DISABLE	1	1
PHASE	1	0
POLARITY	0	9
SLAVE_MODE	0	8
WORD_LENGTH	0	7
	0	6
	0	5
	0	4
	0	3
SSP_MODE	0	2
	0	1
	0	0
	0	0

Table 763. HW_SSP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SDIO_IRQ	RW	0x0	If this bit is set, an SDIO card interrupt has occurred and an IRQ, if enabled, has been sent to the interrupt collector (ICOLL). Write a 1 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.
29	RESP_ERR_IRQ	RW	0x0	When the CHECK_RESP bit in CTRL0 is set, if an unexpected response (or response CRC) is received from the card, this bit will be set. Write a 1 to the SCT clear address to reset this interrupt request status bit.
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 bit is set, a command response time-out has occurred, and an IRQ, if enabled, has been sent to the interrupt collector. This is used for SD/MMC response time-out and MS handshake time-outs. Write a 1 to the SCT clear address to reset this interrupt request status bit.
26	RESP_TIMEOUT_IRQ_EN	RW	0x0	SD/MMC and MS Card Command Response Time-Out Error IRQ Enable. 0 = Response time-out IRQ is masked. 1 = Response time-out 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 16 cycles, then this CPU IRQ will be asserted.

Table 763. HW_SSP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25	DATA_TIMEOUT_IRQ	RW	0x0	Data Transmit/Receive Time-Out Error IRQ. If the time-out counter expires before the data bus is ready for write or sends read data, then a data time-out has occurred. Valid for SD/MMC and MS modes only. Write a 1 to the SCT clear address to reset this interrupt request status bit.
24	DATA_TIMEOUT_IRQ_EN	RW	0x0	Data Transmit/Receive Time-Out Error IRQ Enable. If the time-out counter expires before the data bus is ready for write or sends read data, then a data time-out has occurred. Valid for SD/MMC and MS modes only.
23	DATA_CRC_IRQ	RW	0x0	Data Transmit/Receive CRC Error IRQ. Valid for SD/MMC and MS modes only. Write a 1 to the SCT clear address to reset this interrupt request status bit.
22	DATA_CRC_IRQ_EN	RW	0x0	Data Transmit/Receive CRC Error IRQ Enable. Valid for SD/MMC and MS modes only.
21	FIFO_UNDERRUN_IRQ	RW	0x1	FIFO Underrun Interrupt. If the FIFO is read when it is empty, this bit will be set. Write a 1 to the SCT clear address to reset this interrupt request status bit.
20	FIFO_UNDERRUN_EN	RW	0x0	FIFO Underrun IRQ Enable. If this bit is set and the FIFO_UNDERRUN_IRQ bit is asserted, an IRQ will be generated.
19	CEATA_CCS_ERR_IRQ	RW	0x0	CE-ATA Unexpected CCS Error Interrupt. If this bit is enabled and an unexpected CCS interrupt occurs, an IRQ will be generated. Write a 1 to the SCT clear address to reset this interrupt request status bit.
18	CEATA_CCS_ERR_IRQ_EN	RW	0x0	CE-ATA Unexpected CCS Error Interrupt Enable. 0 = Disable. 1 = Enable.
17	RCV_TIMEOUT_IRQ	RW	0x0	Data Time-Out Interrupt. If this bit is enabled and the receive FIFO is not empty, an IRQ will be generated if 128 HCLK cycles pass before the Data register is read. This is supported for SPI modes only (modes 0,1,2). Write a 1 to the SCT clear address to reset this interrupt request status bit.
16	RCV_TIMEOUT_IRQ_EN	RW	0x0	Receive Time-Out. If this bit is 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	FIFO_OVERRUN_IRQ	RW	0x0	FIFO Overrun Interrupt. Indicates that the FIFO has been written to while full. Write a 1 to the SCT clear address to reset this interrupt request status bit.
14	FIFO_OVERRUN_IRQ_EN	RW	0x0	FIFO Overrun Interrupt Enable. If set, an IRQ will be generated if the 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	CEATA_CCS_ERR_EN	RW	0x0	CE-ATA Unexpected CCS Error Enable. This signal enables the logic to generate an error for unexpected CCS.

STMP3770



Table 763. HW_SSP_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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. In SD/MMC and MS modes: 0 = Command and TX data change after rising edge of SCK. 1 = Command and TX data change after falling edge of SCK. Polarity must be set to 1 for MS 1-bit data mode and 0 for MS 4-bit data mode. In SPI mode: 0 = Steady-state 0 on SCK when data is not being transferred. 1 = Steady-state 1 on SCK when data is not being transferred.
8	SLAVE_MODE	RW	0x0	Slave Mode. 0 = SSP is in master mode. 1 = SSP is in slave mode. Clear to 0 for SD/MMC and MS modes.
7:4	WORD_LENGTH	RW	0x8	Word Length in bits per word. The values 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/CE_ATA 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 synchronous serial mode 0x3 = SD/MMC Card 0x4 = MS 0x7 = CE_ATA. All other values are undefined. 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 SD_MMC = 0x3 SD/MMC mode MS = 0x4 MS mode CE_ATA = 0x7 CE_ATA mode

16.11.8. SSP Data Register Description

HW_SSP_DATA

0x80010070

Table 775. 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 bit. 0 = MS controller is not present in this product. 1 = MS controller is present.
29	SD_PRESENT	RO	0x1	SD/MMC Controller Present bit. 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.
27:22	RSVD	RO	0x0	Reserved.
21	DMASENSE	RO	0x0	Reflects the state of the ssp_dmasense output port. It indicates a DMA error (time-out or CRC) when asserted high at the end of a DMA command.
20	DMATERM	RO	0x0	Reflects the state of the ssp_dmaterm output port. This is a toggle signal.
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. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
15	RESP_ERR	RO	0x0	SD/MMC, MS Card Response to Command with an Error Condition. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
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. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
13	DATA_CRC_ERR	RO	0x0	Data CRC Error. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
12	TIMEOUT	RO	0x0	SD/MMC mode: Time-out counter expired before data bus was ready. MS mode: Time-out expired waiting for interrupt from card. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
11	RCV_TIMEOUT_STAT	RO	0x0	Raw Receive Time-Out Status. Indicates that no read has occurred to non-empty receive data FIFO for 128 cycles.
10	CEATA_CCS_ERR	RO	0x0	Unexpected CCS (CE-ATA Interrupt) occurred. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.

Table 777. HW_SSP_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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_END1 = 0xB MSTK_END2W = 0xC MSTK_END2R = 0xD MSTK_DONE = 0xE
19	CMD_OE	RO	0x0	Enable for SSP_CMD.
18:16	DMA_SM	RO	0x0	DMA State Machine: DMA_IDLE = 0x0 DMA_DMAREQ = 0x1 DMA_DMAACK = 0x2 DMA_STALL = 0x3 DMA_BUSY = 0x4 DMA_DONE = 0x5 DMA_COUNT = 0x6
15:12	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 MMC_CCS = 0x8 MMC_PUP = 0x9 MMC_WAIT = 0xA
11:10	CMD_SM	RO	0x0	MMC Command State Machine: CSM_IDLE = 0x0 CSM_INDEX = 0x1 CSM_ARG = 0x2 CSM_CRC = 0x3
9	SSP_CMD	RO	0x0	SSP_CMD.
8	SSP_RESP	RO	0x0	SSP_RESP.
7:0	SSP_RXD	RO	0x0	SSP_RXD.

16.11.15.SSP Version Register Description

This register reflects the version number for the SSP.

HW_SSP_VERSION 0x80010110

Table 778. HW_SSP_VERSION

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
MAJOR								MINOR								STEP															

STMP3770**Table 779. HW_SSP_VERSION Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x02	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x00	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

SSP Block v2.0

17. LCD INTERFACE (LCDIF)

This chapter describes the LCD interface included on the STMP3770 and includes operation examples. Programmable registers are described in [Section 17.4](#).

17.1. Overview

Many products based on the STMP3770 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, 16/18/24 bpp 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 System (or MCU) interface that is used for setup and to write to the frame buffer or read from it. Also, several displays support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) or the VSYNC mode for high-speed data transfers. In addition to these displays, digital video encoder ICs accept the ITU-R BT.656 format 4:2:2 YCbCr digital component video and convert it to analog TV signals. The LCDIF block on STMP3770 supports all of these different interfaces by sharing register space, FIFOs, and ALU resources.

The high-level block diagram of the LCD interface provided on the STMP3770 is shown in [Figure 74](#).

The block has several major features:

- DMA data transfers for both LCD write and read operations allow minimal CPU overhead.
- Most modes are supported for an 8-bit data bus width in the 100-pin package. (See [Table 780](#).)
- Programmable timing supports a wide range of controllers.
- Programmable parameters for VSYNC interface and DOTCLK interface (VSYNC/HSYNC/DOTCLK with/without ENABLE) to support moving picture mode on a variety of displays.
- ITU-R BT.656 mode (called Digital Video Interface or DVI mode here) to support PAL and NTSC formats.

17.2. Operation

The general description provided in [Section 17.2.1](#) and [Section 17.2.2](#) is applicable to the System write interface, VSYNC, DOTCLK, and DVI interfaces, because they all share the same pipeline until the TXFIFO. Differences for each mode are then described in separate sections, as follows:

- [Section 17.2.3](#), “System Interface”
- [Section 17.2.4](#), “VSYNC Interface”
- [Section 17.2.5](#), “DOTCLK Interface”
- [Section 17.2.6](#), “Controlling VSYNC, HSYNC, and DOTCLK Signal Generation”
- [Section 17.2.7](#), “ITU-R BT.656 Digital Video Interface (DVI)”

LCDIF pin usage by interface mode is described in [Section 17.2.8](#).

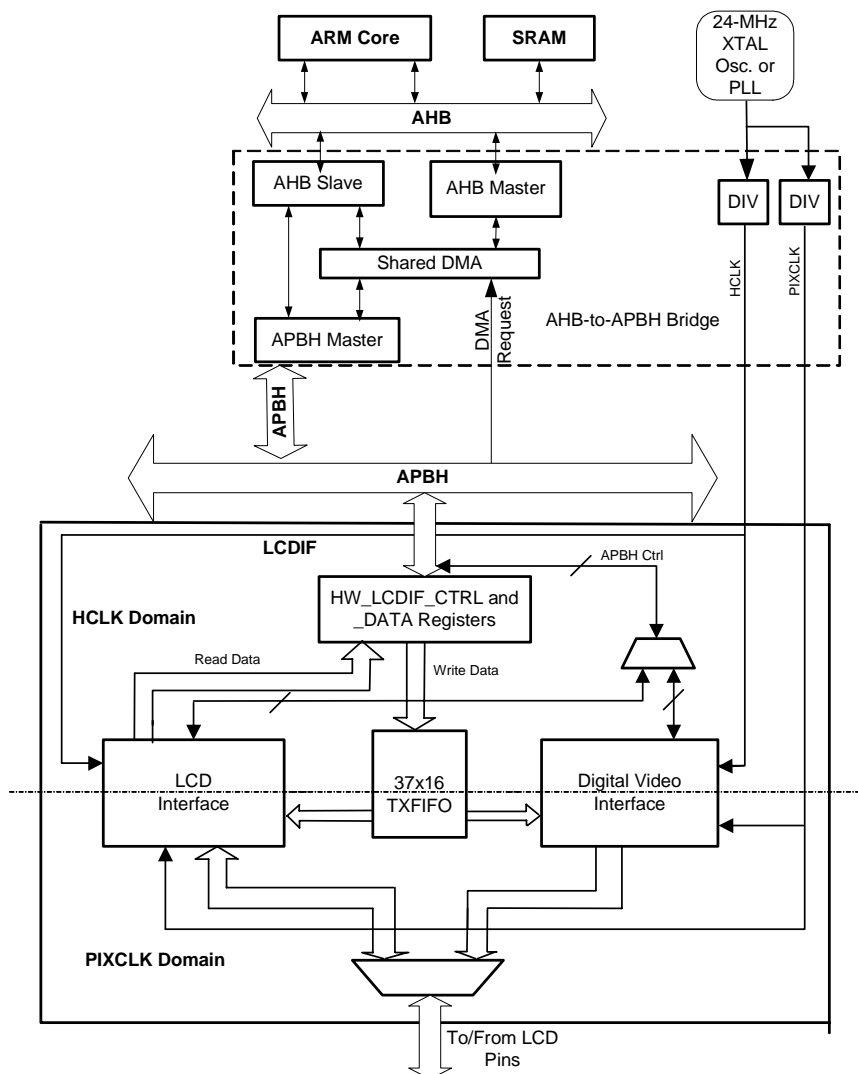


Figure 74. Top-Level LCDIF Block Diagram

17.2.1. DMA and FIFO Control

The LCDIF provides an efficient DMA mechanism to control an external smart LCD with an integrated controller or a digital video encoder supporting the ITU-R BT.656 standard. The block resides on the APBH bus. It has memory-mapped control and data registers. The AHB-APBH bridge DMA can be used to move data from a memory-mapped frame buffer to the LCD's internal frame buffer (or to a digital video encoder) or vice versa. Because the LCDIF uses the AHB-APBH bridge DMA, the software designer can take advantage of the DMA's linked descriptors. This enables substantial flexibility for setting up frame buffers. The CPU can also directly send commands or data by setting up a non-DMA transfer and writing directly to the data register. The APBH interface portion of the LCDIF block works off of HCLK, while the external interface to displays/encoders works off of PIXCLK.

The LCDIF provides a request signal to the central DMA; see [Section 12.2. “APBX DMA” on page 336](#) for more information. The request signal is asserted any time the LCDIF is enabled and its write FIFO has space for more data or its read FIFO has at least one data to be read back. The DMA request signal is also visible in the LCDIF control and status register. Software writing/reading directly to (or from) the LCDIF data register (as opposed to using DMA) should monitor the FIFO status bits and only write (read) new data when the FIFO is not full (not empty). In the DOTCLK and DVI modes, the DMA must be able to keep up with the LCDIF data requests, because these modes require a continuous supply of data and FIFO underflow will send invalid data to the display/video encoder. For that, the HCLK frequency must be much higher than the PIXCLK frequency.

The FIFO status bits in the HW_LCDIF_STAT register indicate the full and empty states of both the write and read FIFOs. When the write FIFO is not full or the read FIFO is not empty, the data register can be safely written/read with a word, half-word, or byte as required; doing otherwise will result in incorrect operation.

- In the System write and read modes, the RUN bit is cleared automatically once the LCDIF has received/transmitted all the data as per the COUNT field and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.
- This is slightly different in the streaming modes, such as VSYNC, DOTCLK, and DVI. If the BYPASS_COUNT bit is set, then to end the current transfer, the software should make the corresponding mode bit the value 0 (VSYNC_MODE, DOTCLK_MODE, or DVI_MODE in CTRL register), so that all data that is currently in the LCDIF TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and toggle the dma_end_cmd signal.

In either of the above cases, the LCDIF transmits out all data it has received to that point—that is, it will flush the FIFO. The transfer will then finish normally, and the LCDIF will return to the idle state and be ready to be programmed for the next transfer.

The SYNC_SIGNALS_ON bit is used in the VSYNC and DOTCLK modes. If the LCD controller requires the VSYNC/HSYNC/DOTCLK signals to be on before starting a data transfer in those modes, the SYNC_SIGNALS_ON bit should be set. This bit should also be set in case it is required to transmit data via the System interface while the VSYNC/HSYNC/DOTCLK signals are still being generated.

The LCDIF can be configured for 8- or 16-bit transfers in the System and VSYNC modes. In 8-bit mode, pins corresponding to the upper bits will send/receive 0, unless they have been configured to connect to another peripheral (such as GPIO). The DOTCLK and DVI modes support 8-bit transfers only.

17.2.2. Write and Read Data Paths

[Figure 75](#) and [Figure 76](#) show the general operations that occur in the write and read data paths.

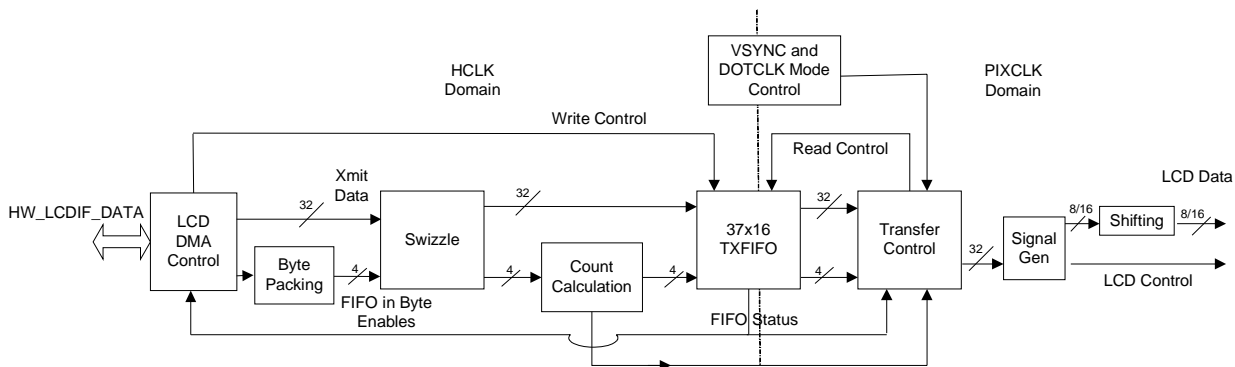


Figure 75. LCDIF Write Path

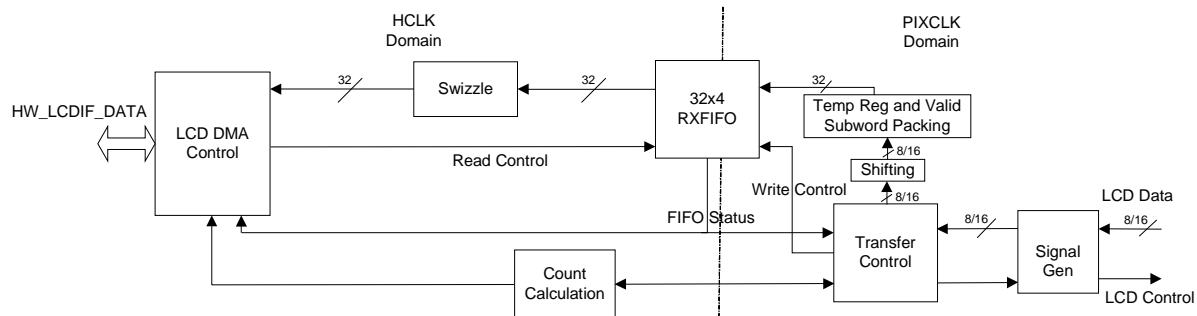


Figure 76. LCDIF Read Path

The LCDIF receives little-endian input in the HW_LCDIF_DATA register, but can transform the output with the DATA_SWIZZLE field in the HW_LCDIF_CTRL register. (The swizzle operation can also be performed on data that is read back from the display via the System interface.) The data swizzle manipulates the outgoing data based on the following values:

- 00 (0): No swizzle (little-endian)
- 01 (1): Swap bytes 0 and 3, swap bytes 1 and 2 (big-endian)
- 10 (2): Swap half-words
- 11 (3): Swap bytes within each half-word

A special bit field in the CTRL1 register, called the BYTE_PACKING_FORMAT, can be used to specify which bytes within the 32-bit word are going to be valid. This bit field is useful, for example, if the RGB data to be transferred is in the 24-bit unpacked format, as shown in the example in [Figure 78](#). The data register (HW_LCDIF_DATA) uses the APBH bus byte enables in combination with the BYTE_PACKING_FORMAT, to determine how many valid bytes are in each written word. For example, if the entire 32-bit word is valid, the LCDIF performs two (16-bit mode) or four (8-bit mode) LCD data operations. In the read-back mode, the number of valid bytes to be stored in a 32-bit word is programmed using the READ_MODE_NUM_PACKED_SUBWORDS field in HW_LCDIF_CTRL1.

The examples in [Figure 77–Figure 80](#) illustrate some different combinations of register programming for write mode. Assume that the data written into the

HW_LCDIF_DATA register is of the format {A7–A0, B7–B0, C7–C0, D7–D0} in 8-bit mode and {A15–A0, B15–B0} in 16-bit mode.

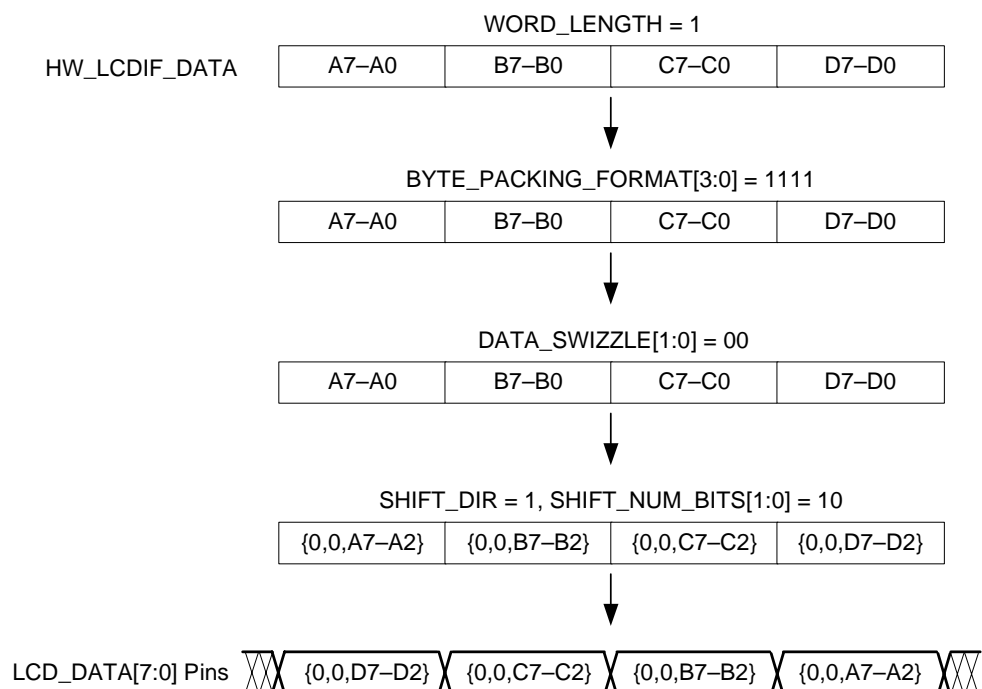


Figure 77. 8-Bit LCDIF Register Programming—Example A

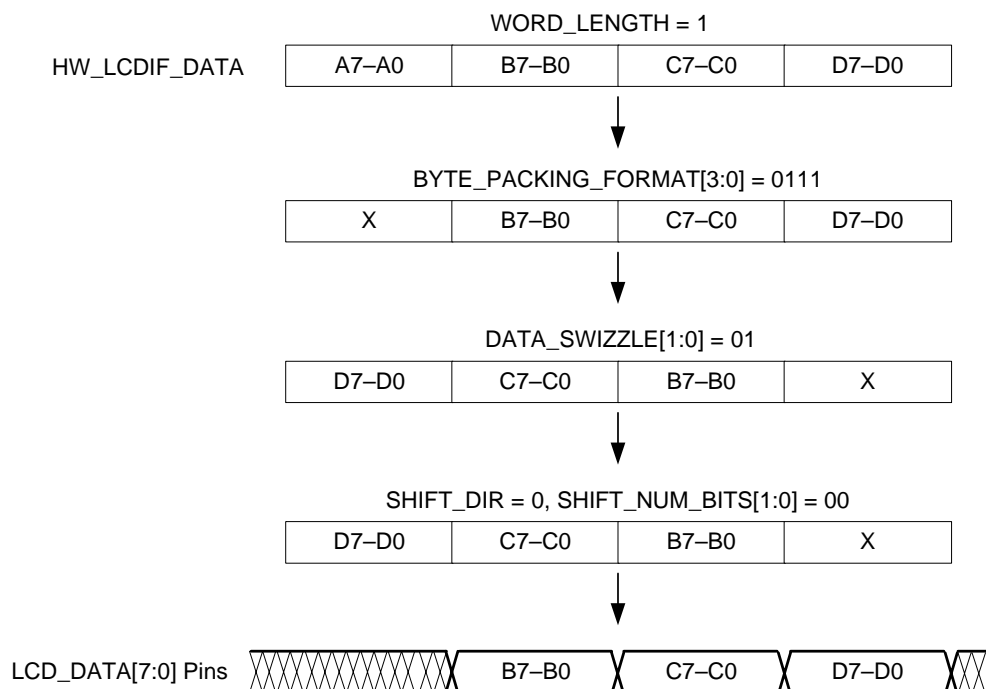


Figure 78. 8-Bit LCDIF Register Programming—Example B

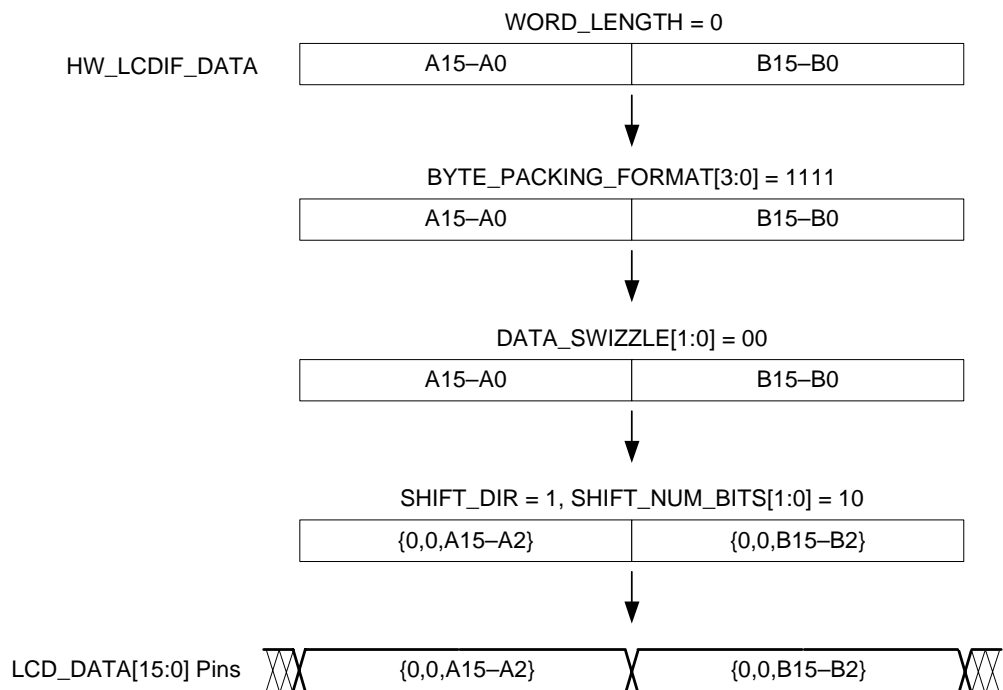


Figure 79. 16-Bit LCDIF Register Programming—Example A

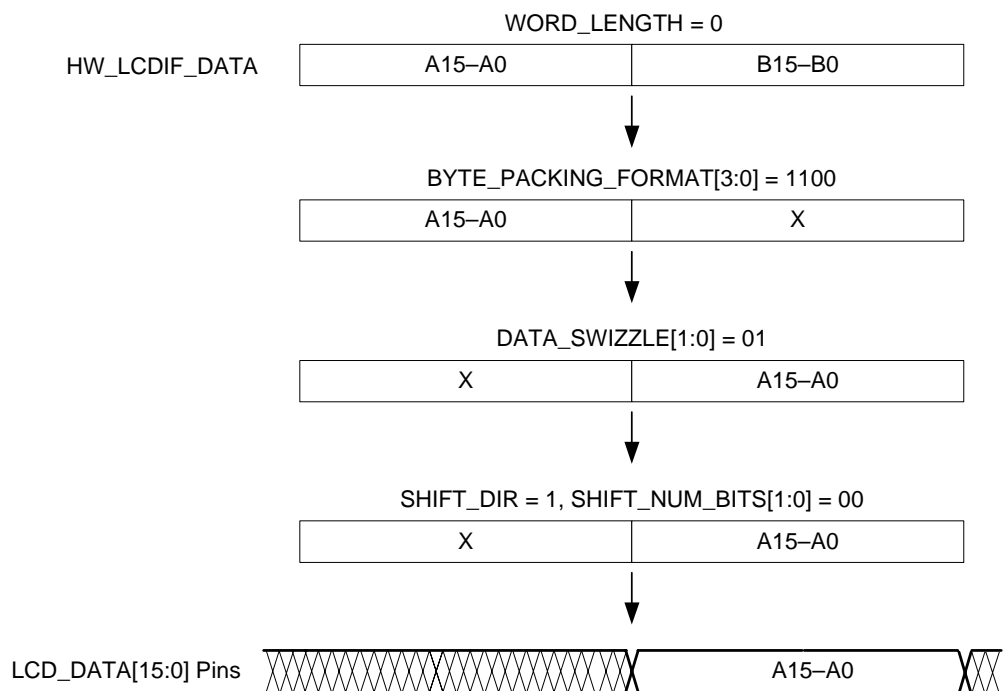


Figure 80. 16-Bit LCDIF Register Programming—Example B

17.2.3. System Interface

The System interface is used to transfer data and commands between the internal buffer of LCD controller/display and the MCU at relatively lower speeds. 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 PIXCLK cycles. 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.

Figure 81 and Figure 82 show the timing-related information in both the write and read modes following 6800/8080 protocol.

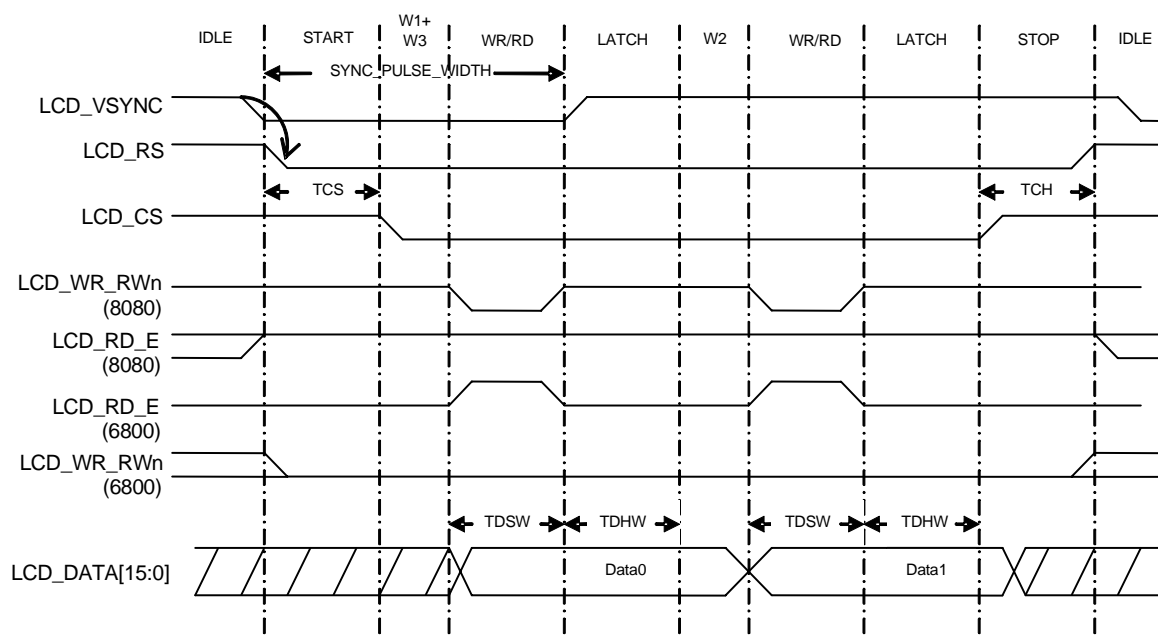


Figure 81. LCD Interface Signals in Write Mode

The LCDIF can support the 6800 MCU protocol, as well as the 8080 MCU protocol.

- In the 6800 mode, the LCD_RD_E pin acts as the active high data strobe (ENABLE) and the LCD_WR_RWn polarity indicates write or read operation.
- In the 8080 mode, the LCD_RD_E and LCD_WR_RWn pins act as active low data strobes in the read and write operations respectively.

The LCDIF has flexible pin and strobe timings, shown in Figure 81 and Figure 82, which enable it to optimally support a wide range of LCDs.

- In both the modes, the data strobe is asserted active for TDS PIXCLK clock cycles and new data is written on leading edge of the data strobe and latched on its trailing edge.
- The data strobe then returns to its original state and remains there for TDH PIXCLK cycles.
- The LCD_CS signal also remains asserted (low) for at least TDH PIXCLK cycles.
- The Data/Command register select signal is asserted TAS PIXCLK cycles before LCD_CS and remains asserted for TCH cycles after LCD_CS is no longer active.

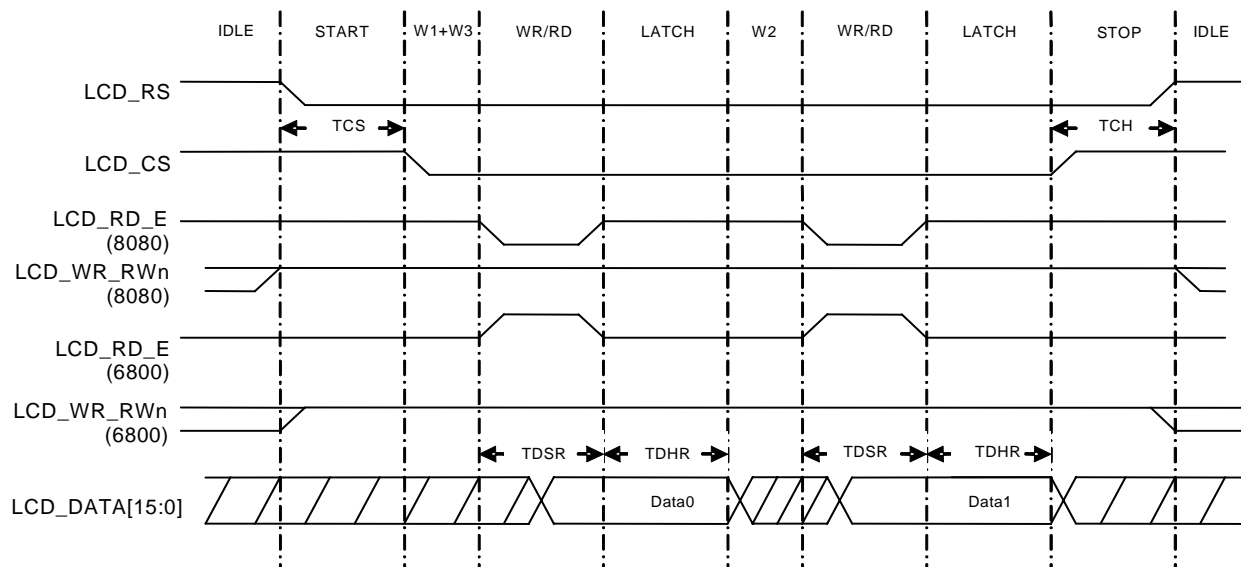


Figure 82. LCD Interface Signals in Read Mode

The minimum cycle time is two PIXCLK cycles ($TDS=TDH=1$). This results in a maximum LCD data rate of 12MB/s when PIXCLK is 24 MHz. TDS and TDH are 8-bit values, so the minimum LCDIF rate is 510 PIXCLK cycles (47 kHz with a 24-MHz PIXCLK). The timings are not automatically adjusted if the PIXCLK frequency changes, so it may be necessary to adjust the timings if PIXCLK changes.

17.2.3.1. Initializing the LCDIF

The following steps show how to initialize the LCDIF block in System mode. These same steps are also common to all other LCDIF modes of operation.

1. To select the pins and their directions for talking to the LCD panel, set the appropriate bits in the HW_PINCTRL_MUXSELx registers in the PINCTRL block.
2. Start the PIXCLK and set the appropriate frequency by programming the registers in CLKCTRL.
3. Bring the LCDIF out of soft reset and clock gate.
4. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller.
5. Set the BUSY_ENABLE bit in the HW_LCDIF_CTRL1 register if connected to an LCD controller that implements a busy line. Otherwise, the busy line is ignored in the System mode of operation.
6. Set the READ_WRITEB bit in the HW_LCDIF_CTRL register, indicating whether it is a read or a write operation.
7. Set the DATA_SWIZZLE according to the endianness of the LCD controller. Also set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to output the data after it is shifted left or right.
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 LCD_RS

signal is high, regardless of the programming of the DATA_SELECT register. This is applicable only in the System and VSYNC modes of interface.

9. Set the WORD_LENGTH field appropriately—0 = 16-bit bus, 1 = 8-bit bus.
10. Enable the necessary IRQs.
11. Set the timings in the HW_LCDIF_TIMING register appropriately.
 - CMD_HOLD is the hold time in cycles for the LCD_RS signal.
 - CMD_SETUP is the setup time in cycles for the LCD_RS 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 for the System and VSYNC modes.
12. Set the MODE86 register based on whether the LCD controller uses 6800 or 8080 mode of transfer.
13. Set LCD_CS_CTRL in HW_LCDIF_CTRL1 depending on what behavior of CS is required by the LCD panel for the particular mode of operation. See the description of this bit field in CTRL1 register for more details.

17.2.3.2. Runtime Steps for System Interface Write Operations

The following runtime steps show how to perform write operations using the System interface:

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 halfwords. Make sure that the BYASS_COUNT bit is 0.
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 bits to ensure that it does not overflow the LCDIF data buffers.

17.2.4. VSYNC Interface

The VSYNC interface uses the same protocol as the System interface, with an additional signal VSYNC at the frame rate of the display. It is used in the moving picture display mode where data has to be written to the internal LCD buffer at a speed higher than the display rate and displayed in synchronization with the VSYNC signal. The VSYNC signal is programmable for period, polarity and direction. Many other programmable parameters are shared with the System interface.

The SYNC_SIGNALS_ON bit can be set if the target requires the VSYNC signal to be generated at least one frame before the actual data transfer starts. The VERITCAL_WAIT_CNT indicates the number of PIXCLKs to wait from the leading VSYNC edge before starting data transfer on the interface.

The WAIT_FOR_VSYNC_EDGE bit indicates whether the hardware should wait until it sees the leading VSYNC edge before starting the data transfer. This bit can be helpful when there is more than one DMA buffer for one frame worth of data and DMA chaining is required. In that case, the last DMA descriptor in the current frame can make WAIT_FOR_VSYNC_EDGE to be 0, and the first descriptor in the next frame can make WAIT_FOR_VSYNC_EDGE 1. This transition from 0 to 1 helps differentiate between data belonging to the current frame, and that belonging to the next frame. In this case, the WAIT4ENDCMD bit of the APBH Channel 0 Register

should not be set since the block will not send any dma_end_cmd signal to the DMA until the transfer is completely stopped.

17.2.4.1. Code Examples

The code below shows an example of two DMA descriptors chained together with BYPASS_COUNT = 1.

```
static reg32_t LCDIF_DMA_CMD1[4] =
{
// WD0: Link to next DMA descriptor //
  (reg32_t) LCDIF_DMA_CMD2,
// WD1: DMA Channel Command //
  (BF_APBH_CHn_CMD_XFER_COUNT (5000) | // First part of a frame
   BF_APBH_CHn_CMD_CMDWORDS (1) | // Number of PIO words to write
   BF_APBH_CHn_CMD_WAIT4ENDCMD (0) |
   BF_APBH_CHn_CMD_SEMAPHORE (1) |
   BF_APBH_CHn_CMD_IRQONCMPLT (1) |
   BF_APBH_CHn_CMD_CHAIN (1) | // Read from memory
   BF_APBH_CHn_CMD_COMMAND (2)),
// WD2: Buffer address //
  (reg32_t) FrameBuffer1, // Data to DMA for first part of a frame
//WD3: PIO word //
  BF_LCDIF_CTRL_BYPASS_COUNT (1) |
  BF_LCDIF_CTRL_WAIT_FOR_VSYNC_EDGE (1)
};

static reg32_t LCDIF_DMA_CMD2[4] =
{
// WD0: Link to next DMA descriptor //
  (reg32_t) LCDIF_DMA_CMD1,
// WD1: DMA Channel Command //
  (BF_APBH_CHn_CMD_XFER_COUNT(4000) | // Second part of a frame
   BF_APBH_CHn_CMD_CMDWORDS(1) | // Number of PIO words to write
   BF_APBH_CHn_CMD_WAIT4ENDCMD(0) |
   BF_APBH_CHn_CMD_SEMAPHORE(1) |
   BF_APBH_CHn_CMD_IRQONCMPLT(1) |
   BF_APBH_CHn_CMD_CHAIN(1) | // Read from memory,
   BF_APBH_CHn_CMD_COMMAND(2)),
// WD2: Buffer address //
  (reg32_t) FrameBuffer2, // Data to DMA for first part of a frame
//WD3: PIO word //
  BF_LCDIF_CTRL_BYPASS_COUNT(1) |
  BF_LCDIF_CTRL_WAIT_FOR_VSYNC_EDGE(0)
};
```

In the case where each frame consists of just one DMA buffer, or to use the COUNT field to transfer data per frame, the BYPASS_COUNT field can be 0, so that the value in the COUNT field can be used to indicate how many bytes to transfer. The following code shows this situation.

Here, WAIT4ENDCMD bit of APBH DMA channel 0 register can be set to 1 since dma_end_cmd will be issued every time the block has transferred COUNT number of data. SYNC_SIGNALS_ON bit also should be made 1 in this special case if VSYNC signal is an output from the block.

```
static reg32_t LCDIF_DMA_CMD1[4] =
{
// WD0: Link to next DMA descriptor //
  (reg32_t) LCDIF_DMA_CMD2,
// WD1: DMA Channel Command //
  (BF_APBH_CHn_CMD_XFER_COUNT (5000) | //First part of frame
   BF_APBH_CHn_CMD_CMDWORDS (1) | // Number PIO words to write
   BF_APBH_CHn_CMD_WAIT4ENDCMD (1) |
   BF_APBH_CHn_CMD_SEMAPHORE (1) |
   BF_APBH_CHn_CMD_IRQONCMPLT (1) |
```



```

    BF_APBH_CHn_CMD_CHAIN (1)
    BF_APBH_CHn_CMD_COMMAND (2)), // Read from memory

// WD2: Buffer address //
    (reg32_t) FrameBuffer1, // Data to DMA for first part of a frame

//WD3: PIO word //
    BF_LCDIF_CTRL_BYPASS_COUNT (0)
    BF_LCDIF_CTRL_WAIT_FOR_VSYNC_EDGE (1)
    BF_LCDIF_CTRL_COUNT (5000)
};

static reg32_t LCDIF_DMA_CMD2[4] =
{
// WD0: Link to next DMA descriptor //
    (reg32_t) LCDIF_DMA_CMD1,
// WD1: DMA Channel Command //
    (BF_APBH_CHn_CMD_XFER_COUNT (4000) //Second part of frame
    BF_APBH_CHn_CMD_CMDWORDS (1) // Number PIO words to write
    BF_APBH_CHn_CMD_WAIT4ENDCMD (0)
    BF_APBH_CHn_CMD_SEMAPHORE (1)
    BF_APBH_CHn_CMD_IRQONCMPLT (1)
    BF_APBH_CHn_CMD_CHAIN (1)
    BF_APBH_CHn_CMD_COMMAND (2)), // Read from memory

// WD2: Buffer address //
    (reg32_t) FrameBuffer2, // Data to DMA for second part of a frame

//WD3: PIO word //
    BF_LCDIF_CTRL_BYPASS_COUNT (0)
    BF_LCDIF_CTRL_WAIT_FOR_VSYNC_EDGE (0)
    BF_LCDIF_CTRL_COUNT (4000)
};

```

The VSYNC interface allows for 8-bit or 16-bit data transfers. This signal is mapped onto two pins: LCD_BUSY and LCD_D15. The desired pin can be selected in the PINCTRL/PINMUX registers.

The following code illustrates programming the LCDIF with VSYNC signal as an output.

```

// Note: Common initialization steps in Section 17.2.3.1 must also be
// executed along with the following code.

BF_CS2 (LCDIF_CTRL, VSYNC_MODE, 1, WAIT_FOR_VSYNC_EDGE. 1);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 1);
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Making VSYNC signal an output
BF_CS1 (LCDIF_VDCTRL0, ENABLE_PRESENT, 0); //Disabling the generation of ENABLE signal
BF_CS1 (VDCTRL0, VSYNC_POL, 0);
//Setting the polarity of VSYNC signal to be low during VSYNC_PULSE_WIDTH time
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 0, VSYNC_PULSE_WIDTH_UNIT, 0);
BF_CS2 (LCDIF_VDCTRL1, VSYNC_PERIOD, 6000, VSYNC_PULSE_WIDTH, 100);
//Frame display rate in terms of number of PIXCLKs.
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 0, HSYNC_PERIOD, 0);
BF_CS1 (LCDIF_VDCTRL3, VERTICAL_WAIT_CNT, 50);
BF_CS1 (LCDIF_VDCTRL3, SYNC_SIGNALS_ON, 1);
// If it is required to generate VSYNC signal before the actual data transfer begins
BF_CS1 (LCDIF_CTRL, RUN, 1);

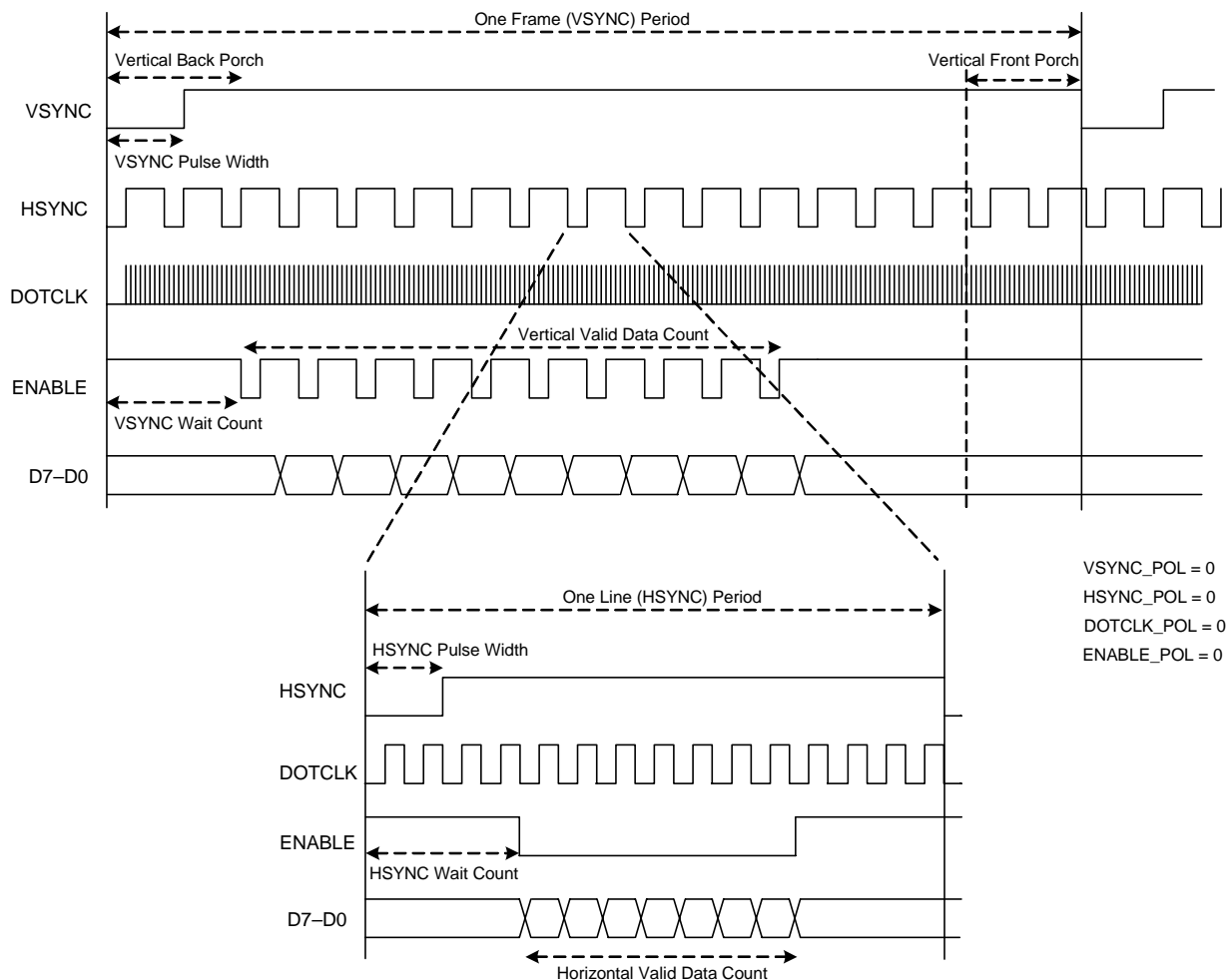
```

To stop the transfer completely, the ideal way is to make VSYNC_MODE = 0. The block will transmit everything in its FIFO, turn off the RUN bit and toggle the dma_end_cmd signal indicating to the DMA that it is done with the transfer.

17.2.5. DOTCLK Interface

The DOTCLK interface is another mode used in moving picture displays. It includes the VSYNC, HSYNC, DOTCLK and (optional) ENABLE signals. The interface is popularly called the RGB interface if the ENABLE signal is present.

[Figure 83](#) shows the DOTCLK protocol with its programmable parameters.


Figure 83. LCD Interface Signals in DOTCLK Mode

The DOTCLK mode writes data at high speed to the LCD, and the display operation is synchronized with the VSYNC, HSYNC and DOTCLK signals. The hardware can support only 8-bit transfers in this mode. The HSYNC, DOTCLK and ENABLE signals are multiplexed with the upper bits of the 16-bit LCDIF data bus. The VSYNC signal can be multiplexed onto two different pins of the chip, as mentioned in [Section 17.2.3.1](#). The polarities, periods and pulse-widths of the control signals are programmable using the HW_LCDIF_VDCTRL0-3 registers. The units for the VSYNC signal can be number of PIXCLKs or number of horizontal lines and can be selected using the VSYNC_PULSE_WIDTH_UNIT and VSYNC_PERIOD_UNIT bit fields. The VERTICAL_WAIT_CNT is by default given the same unit as the VSYNC_PERIOD. The PIXCLK is controlled using the HW_CLKCTRL_PIX, HW_CLKCTRL_FRAC, and HW_CLKCTRL_CLKSEQ registers in the CLKCTRL block.

17.2.5.1. Code Example

The following code shows an example for programming a 320x240 display with a display area of 160x120 at the center of the display. Note that setting up the display for this display window must be done through the System mode.

// Note: Common initialization steps in [Section 17.2.3.1](#) must also be
 // executed along with the following code.

```
BF_CS1 (LCDIF_CTRL, DOTCLK_MODE, 1);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 1); //Always for DOTCLK mode
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Vsync is always an output in the DOTCLK mode
BF_CS4 (LCDIF_VDCTRL0, VSYNC_POL, 0, HSYNC_POL, 0, DOTCLK_POL, 0, ENABLE_POL, 0);
BF_CS1 (LCDIF_VDCTRL0, ENABLE_PRESENT, 1);
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 1, VSYNC_PULSE_WIDTH_UNIT, 1);
BF_CS1 (LCDIF_VDCTRL0, DOTCLK_V_VALID_DATA_CNT, 120);
BF_CS2 (LCDIF_VDCTRL1, VSYNC_PULSE_WIDTH, 2, VSYNC_PERIOD, 240);
BF_CS3 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 10, HSYNC_PERIOD, 320, DOTCLK_H_VALID_DATA_CNT, 160);
BF_CS3 (LCDIF_VDCTRL3, SYNC_SIGNALS_ON, 1, HORIZONTAL_WAIT_CNT, 80, VERTICAL_WAIT_CNT, 60);
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

To stop the transfer completely, the ideal way is to make DOTCLK_MODE = 0. In that case, the block will transmit whatever it had in its FIFO, turn off the RUN bit and toggle the dma_end_cmd signal indicating to the DMA that it is done with the transfer.

17.2.6. Controlling VSYNC, HSYNC, and DOTCLK Signal Generation

The SYNC_SIGNALS_ON bit field is used to control the generation of the VSYNC, HSYNC, and DOTCLK signals.

- If SYNC_SIGNALS_ON = 1, DOTCLK_MODE = 0, VSYNC_MODE = 0, and DVI_MODE = 0, then the VSYNC, HSYNC, and DOTCLK signals will start being generated based on parameters defined in HW_LCDIF_VDCTRL0–3. If only the VSYNC signal needs to be generated, make sure to have HSYNC_PERIOD = 0.
- To switch from the System interface to the DOTCLK interface, set SYNC_SIGNALS_ON = 1 if the LCD controller needs to wait for one frame or more before accepting data. When a sufficient number of blank VSYNC frames have been transmitted, set RUN = 1 so that real data can be transmitted.
- To waste no frames at all (or for controllers that do not support switching between System and DOTCLK modes), program SYNC_SIGNALS_ON = 0, DOTCLK_MODE = 1, and RUN = 1. The LCDIF will start generating VSYNC/HSYNC/DOTCLK signals and start transmitting data from the first frame itself.

17.2.7. ITU-R BT.656 Digital Video Interface (DVI)

ITU-R BT.656 Digital Video Interface shown in [Figure 84](#) transmits 4:2:2 YCbCr digital component video to a digital video encoder that can translate it into analog NTSC or PAL TV signal. Unique timing codes (timing reference signals) are embedded within the video stream to indicate the different timing events that would have been otherwise indicated by VSYNC, HSYNC and BLANK signals.

The hardware supports 8-bit data transfers; the pins are shared with the lower 8 bits of LCD data bus. The LCD_RS pin is shared with the clock signal of the interface (called CCIRCLK here for uniqueness). The mode also shares the write FIFO with the LCD interface and the associated pipeline. The programmable parameters in registers HW_LCDIF_DVICTRL0–3 allow setting the total number of horizontal lines per frame, vertical and horizontal blanking interval, odd and even field start and end positions, etc. In short, these parameters are provided to ensure that the hardware has enough flexibility to generate the right NTSC or PAL data streams.

Most of the initialization steps of [Section 17.2.3.1](#), such as data shifting, swizzle, etc., are applicable to DVI mode also. BYPASS_COUNT should always be 1 when

operating in the DVI mode. The register descriptions in [Section 17.4](#) include example code for programming the DVICTRL0–3 registers.

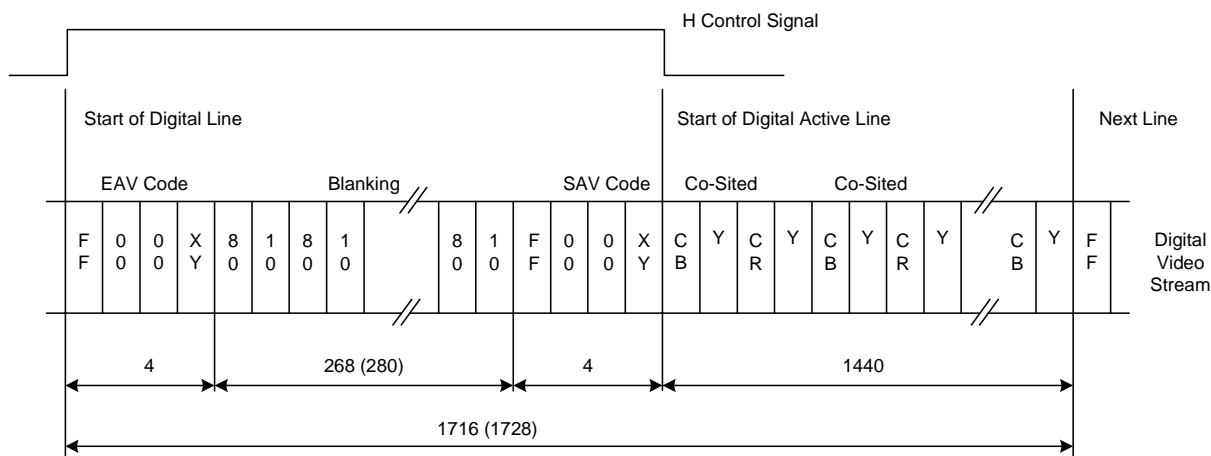


Figure 84. LCDIF Interface Signals in ITU-R BT.656 Digital Video Interface Mode

17.2.8. LCDIF Pin Usage by Interface Mode

Table 780 shows the pin usage for all of the supported modes when the HW_PINCTRL_MUXSEL2 and HW_PINCTRL_MUXSEL3 registers are programmed for LCD functionality. See [Chapter 33, “Pin Control and GPIO” on page 929](#), for a more complete description of pin multiplexing options and how to program each pin individually.

Note that the VSYNC signal has been mapped onto two pins, LCD_BUSY and LCD_D15. The pin multiplexing can be programmed to select either of those pins to function as VSYNC, with the following restrictions:

- In 16-bit VSYNC mode, pin LCD_BUSY must be programmed as LCD_VSYNC.
- In 8-bit VSYNC or DOTCLK mode, either of the pins can be programmed to be LCD_VSYNC.
- In DOTCLK mode, the HSYNC, DOTCLK and ENABLE signals are internally mapped on LCD_14, LCD_13 and LCD_12 pins respectively by the LCDIF block, so that no PINMUX programming is required.

Table 780. External Pins Supported in LCDIF Interface Modes

PIN NAME	8-BIT SYSTEM LCD MODE	16-BIT SYSTEM LCD MODE	8-BIT VSYNC LCD MODE	16-BIT VSYNC LCD MODE	8-BIT DOTCLK LCD MODE	8-BIT DVI MODE
LCD_RS	LCD_RS	LCD_RS	LCD_RS	LCD_RS	LCD_RS	LCD_DVICLK
LCD_CS	LCD_CS	LCD_CS	LCD_CS	LCD_CS	LCD_CS	X
LCD_WR_RWn	LCD_WR_RWn	LCD_WR_RWn	LCD_WR_RWn	LCD_WR_RWn	LCD_WR_RWn	X
LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	X
LCD_D15/ LCD_VSYNC*	X	LCD_D15	LCD_VSYNC (optional)	LCD_D15	LCD_VSYNC (optional)	X

Table 780. External Pins Supported in LCDIF Interface Modes (Continued)

PIN NAME	8-BIT SYSTEM LCD MODE	16-BIT SYSTEM LCD MODE	8-BIT VSYNC LCD MODE	16-BIT VSYNC LCD MODE	8-BIT DOTCLK LCD MODE	8-BIT DVI MODE
LCD_D14/ LCD_HSYNC	X	LCD_D14	X	LCD_D14	LCD_HSYNC	X
LCD_D13/ LCD_DOTCLK	X	LCD_D13	X	LCD_D13	LCD_DOTCLK	X
LCD_D12/ LCD_ENABLE	X	LCD_D12	X	LCD_D12	LCD_ENABLE	X
LCD_D11	X	LCD_D11	X	LCD_D11	X	X
LCD_D10	X	LCD_D10	X	LCD_D10	X	X
LCD_D9	X	LCD_D9	X	LCD_D9	X	X
LCD_D8	X	LCD_D8	X	LCD_D8	X	X
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	X
LCD_BUSY/ LCD_VSYNC	LCD_BUSY	LCD_BUSY	LCD_BUSY (OR optional LCD_VSYNC)	LCD_VSYNC	LCD_BUSY (OR optional LCD_VSYNC)	X

17.3. 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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

Table 782. HW_LCDIF_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
24	DVI_MODE	RW	0x0	Set to 1 to get into the ITU-R BT.656 digital video interface mode. Toggle this bit from 1 to 0 to make the hardware go out of DVI mode after completing all data transfers, deasserting the RUN bit and toggling the dma_end_cmd signal.
23	BYPASS_COUNT	RW	0x0	Set to 1 to disable the transfer count decrement during data transfer. The transfer count should be usually used only in the System interface mode (both read and write). In the VSYNC mode, this bit field should be cleared to 0 only if each display frame consists of just one DMA buffer.
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 HW_LCDIF_DATA below 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	VSYNC_MODE	RW	0x0	Set this bit to 1 to make the LCDIF hardware go into VSYNC mode and begin generating the VSYNC signal off the PIXCLK if it has been programmed to be output. Toggle this bit from 1 to 0 to make the hardware go out of VSYNC mode after completing all data transfers, deasserting the RUN bit and toggling the dma_end_cmd signal.
19	DOTCLK_MODE	RW	0x0	Set this bit to 1 to make the hardware go into the DOTCLK mode, i.e., VSYNC/HSYNC/DOTCLK/ENABLE interface mode. ENABLE is optional, selected by the ENABLE_PRESENT bit. Toggle this bit from 1 to 0 to make the hardware go out of DOTCLK mode after completing all data transfers, deasserting the RUN bit and toggling the dma_end_cmd signal.
18	DATA_SELECT	RW	0x0	Command Mode polarity bit. This bit should be changed only when RUN is 0. CMD_MODE = 0x0 Command Mode. LCD_RS signal is Low. DATA_MODE = 0x1 Data Mode. LCD_RS 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.

Table 782. HW_LCDIF_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
16	RUN	RW	0x0	When this bit is set by software, the LCDIF sends 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 LCDIF Control Register provides a variety of control functions to the programmer. These functions allow the interface to work with a variety of LCD controllers and to minimize overhead and increase performance of LCD programming. The register has been organized such that switching between the different LCD modes can be done with minimum PIO writes.

17.4.2. LCDIF General Control 1 Register Description

The LCDIF Control Register provides additional control of the LCDIF block.

HW_LCDIF_CTRL1	0x80030010
HW_LCDIF_CTRL1_SET	0x80030014
HW_LCDIF_CTRL1_CLR	0x80030018
HW_LCDIF_CTRL1_TOG	0x8003001C

Table 783. HW_LCDIF_CTRL1

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3
RSVD											BYTE_PACKING_FORMAT					OVERFLOW_IRQ_EN	UNDERFLOW_IRQ_EN	CUR_FRAME_DONE_IRQ_EN	VSNC_EDGE_IRQ_EN	OVERFLOW_IRQ	UNDERFLOW_IRQ	CUR_FRAME_DONE_IRQ	VSNC_EDGE_IRQ	READ_MODE_NUM_PACKED_SUBWORDS				FIRST_READ_DUMMY
																												LCD_CS_CTRL
																												BUSY_ENABLE
																												MODE86
																												RESET

Table 784. HW_LCDIF_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	RSVD	RO	0x0	Reserved.
19:16	BYTE_PACKING_FORMAT	RW	0xf	This bit field is used to show which data bytes in the HW_LCDIF_DATA register are valid and should be transmitted. Default value 0xF indicates that all bytes are valid. For 8-bit transfers, any combination in this bit field mean that valid data is present in the corresponding bytes. In the 16-bit mode, a 16-bit half-word is valid only if adjacent bits [1:0] or [3:2] or both are 1. A value of 0x0 means that none of the bytes is valid and should not be used. For example, set the bit field value to 0x7 if the display data is arranged in the 24-bit unpacked format (X-R-G-B where X value is invalid and should not be transmitted).
15	OVERFLOW_IRQ_EN	RW	0x0	This bit is set to enable an overflow interrupt in the TXFIFO in write mode or in the RXFIFO in read mode.
14	UNDERFLOW_IRQ_EN	RW	0x0	This bit is set to enable an underflow interrupt in TXFIFO in the write mode or in the RXFIFO in read mode.
13	CUR_FRAME_DONE_IRQ_EN	RW	0x0	This bit is set to 1 enable an interrupt every time the hardware enters in the vertical blanking state.
12	VSYNC_EDGE_IRQ_EN	RW	0x0	This bit is set to enable an interrupt every time the hardware encounters the leading VSYNC edge in VSYNC and DOTCLK modes.
11	OVERFLOW_IRQ	RW	0x0	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software writing a 1 to its SCT clear address. A TXFIFO overflow in the write mode (System/VSYNC/DOTCLK/DVI mode) or RXFIFO overflow in System read mode was detected, data samples have been lost. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
10	UNDERFLOW_IRQ	RW	0x0	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software writing a 1 to its SCT clear address. A TXFIFO underflow in the write mode (System/VSYNC/DOTCLK/DVI mode) or RXFIFO underflow in the System read mode was detected. Could produce an error in the DOTCLK/DVI modes. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.

Table 784. HW_LCDIF_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9	CUR_FRAME_DONE_IRQ	RW	0x0	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a 1 to its SCT clear address. It indicates that the hardware has completed transmitting the current frame and is in the vertical blanking period in the DOTCLK/DVI modes. In the VSYNC mode, it indicates that the hardware has completed transmitting all data samples until it detected the WAIT_FOR_VSYNC_EDGE = 1. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
8	VSYNC_EDGE_IRQ	RW	0x0	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software writing a 1 to its SCT clear address. It is set whenever the leading VSYNC edge is detected. Useful in counting the number of frames to wait before starting the DMA transfer in the VSYNC/DOTCLK modes. NO_REQUEST = 0x0 No Interrupt Request Pending. REQUEST = 0x1 Interrupt Request Pending.
7:5	READ_MODE_NUM_PACKED_SUBWORDS	RW	0x0	Indicates the number of valid 8-bit or 16-bit subwords that will be packed in the 32-bit HW_LCDIF_DATA register in the read mode. The subword size (8 bits or 16 bits) is determined by the WORD_LENGTH bit. The swizzle operation is performed after READ_MODE_NUM_PACKED_SUBWORDS number of data has been received from the interface and stored in the little-endian format. For example, if the data to be read back should be stored in memory in 24-bit unpacked RGB format, set this value to 0x3 so that each 32-bit word will contain only 3 valid bytes (RGB).
4	FIRST_READ_DUMMY	RW	0x0	Set this bit to 1 if the first data read back from the LCD panel/controller is dummy data. It will then not be stored in the read FIFO.
3	LCD_CS_CTRL	RW	0x0	If this bit is cleared to 0 in VSYNC mode, the LCD_CS pin toggles according to the System interface protocol. If it is cleared to 0 in the DOTCLK mode, the LCD_CS pin is driven high throughout the operation. In both VSYNC and DOTCLK modes, if this bit is set to 1, the LCD_CS pin is driven low throughout the operation. In System interface mode, this bit must be cleared to the default value of 0.
2	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.

Table 784. HW_LCDIF_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	MODE86	RW	0x0	<p>This bit is used to select between the 8080 and 6800 series of microprocessor modes. This bit should be changed only when RUN is 0.</p> <p>8080_MODE = 0x0 Pins LCD_WR_RWn and LCD_RD_E function as active low WR and active low RD signals respectively.</p> <p>6800_MODE = 0x1 Pins LCD_WR_RWn and LCD_RD_E function as Read/Writeb and active high Enable signals respectively.</p>
0	RESET	RW	0x0	<p>Reset bit for the external LCD controller. This bit can be changed at any time. It CANNOT be reset by SFTRST.</p> <p>LCDRESET_LOW = 0x0 LCD_RESET output signal is low.</p> <p>LCDRESET_HIGH = 0x1 LCD_RESET output signal is high.</p>

DESCRIPTION:

The LCDIF General Control 1 Register provides additional programming to the LCDIF. It implements some bits that are unlikely to change often in a particular application. It also carries interrupt-related bits that are common across more than one mode of operation.

17.4.3. LCDIF Timing Register Description

The LCDIF Timing Register controls the various setup and hold times enforced by the LCD interface.

HW LCDIF TIMING

0x80030020

Table 785. HW LCDIF TIMING

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
CMD_HOLD								CMD_SETUP								DATA_HOLD								DATA_SETUP							

Table 786. HW_LCDIF_TIMING Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	CMD_HOLD	RW	0x00	Number of PIXCLK cycles that the LCD_RS signal is active after LCD_CS is deasserted.
23:16	CMD_SETUP	RW	0x00	Number of PIXCLK cycles that the LCD_RS signal is active before LCD_CS is asserted.
15:8	DATA_HOLD	RW	0x00	Data bus hold time in PIXCLK cycles. Also the time that the data strobe is deasserted in a cycle
7:0	DATA_SETUP	RW	0x00	Data bus setup time in PIXCLK cycles. Also the time that the data strobe is asserted in a cycle.

Table 788. HW_LCDIF_VDCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
27	VSYNC_POL	RW	0x0	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.
26	HSYNC_POL	RW	0x0	Default 0 active low during HSYNC_PULSE_WIDTH time and will be high during the rest of the HSYNC period. Set it to 1 to invert the polarity.
25	DOTCLK_POL	RW	0x0	Default is data launched at negative edge of DOTCLK and captured at positive edge. Set it to 1 to invert the polarity. Set it to 0 in DVI mode.
24	ENABLE_POL	RW	0x0	Default 0 active low during valid data transfer on each horizontal line.
23:22	RSVD	RO	0x0	Reserved.
21	VSYNC_PERIOD_UNIT	RW	0x0	Default 0 for counting VSYNC_PERIOD in terms of PIXCLKs. Set it to 1 to count in terms of complete horizontal lines. PIXCLKs will probably be used in the VSYNC mode, while horizontal line will be used in the DOTCLK mode.
20	VSYNC_PULSE_WIDTH_UNIT	RW	0x0	Default 0 for counting VSYNC_PULSE_WIDTH in terms of PIXCLKs. Set it to 1 to count in terms of complete horizontal lines.
19	INTERLACE	RW	0x0	Set this bit to 1 to make the total VSYNC period equal to the VSYNC_PERIOD field plus half the HORIZONTAL_PERIOD field (i.e., VSYNC_PERIOD field plus half horizontal line); otherwise, it is just VSYNC_PERIOD. Should be used only in DOTCLK mode, not in VSYNC interface mode.
18:10	RSVD	RO	0x000	Reserved.
9:0	DOTCLK_V_VALID_DATA_CNT	RW	0x000	Number of horizontal lines that contain valid data in the DOTCLK mode.

DESCRIPTION:

This register gives general programmability to the VSYNC signal, including polarity, direction, wait period, etc.

DESCRIPTION:

The values used in this register provide values for the different parameters for the ITU-R BT.656 interface.

EXAMPLE:

```
//525/60 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0); //1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x106); //262
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x20D); //525
//625/50 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0); //1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x118); //280
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x271); //625
```

17.4.9. Digital Video Interface Control 1 Register Description

The Digital Video Interface Control 1 Register provides the overall control of the digital video interface.

```
HW LCDIF DVICTRL1          0x80030080
```

Table 797. HW LCDIF DVICTRL1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD		F1_START_LINE										F1_END_LINE										F2_START_LINE									

Table 798. HW_LCDIF_DVCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29:20	F1_START_LINE	RW	0x000	Vertical line number from which Field 1 begins.
19:10	F1_END_LINE	RW	0x000	Vertical line number at which Field1 ends.
9:0	F2_START_LINE	RW	0x000	Vertical line number from which Field 2 begins.

DESCRIPTION:

The values used in this register provide values for the different parameters for the ITU-R BT.656 interface.

EXAMPLE:

```
//525/60 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x4);//4
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x109);//265
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x10A);//266
//625/50 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x138);//312
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x139);//313
```

```
//525/60 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x3); //3
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x108); //264
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x11A); //282
//625/50 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x271); //625
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x137); //311
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x14F); //335
```


17.4.13. LCDIF Status Register Description

The LCDIF Status Register can be used to check the current status of the LCDIF block.

HW LCDIF STAT

0x800300c0

Table 805. HW_LCDIF_STAT

[illegible]

Table 806. HW_LCDIF_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	PRESENT	RO	0x1	0 = LCDIF is not present in this product 1 = LCDIF is present.
30	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.
29	RXFIFO_FULL	RO	0x0	Read-only view of the signal that indicates that LCD read datapath FIFO is full, will be generally used in the write mode of the LCD interface.
28	RXFIFO_EMPTY	RO	0x1	Read-only view of the signal that indicates that LCD read datapath FIFO is empty, will be generally used in the read mode of the LCD interface.
27	TXFIFO_FULL	RO	0x0	Read-only view of the signal that indicates that LCD write datapath FIFO is full, will be generally used in the write mode of the LCD interface.
26	TXFIFO_EMPTY	RO	0x0	Read-only view of the signal that indicates that LCD write datapath FIFO is empty, will be generally used in the read mode of the LCD interface.
25	BUSY	RO	0x0	Read-only view of the input busy signal from the external LCD controller.
24	DVI_CURRENT_FIELD	RO	0x0	Read-only view of the current field being transmitted. DVI_CURRENT_FIELD = 0 means field 1. DVI_CURRENT_FIELD = 1 means field 2.
23:0	RSVD	RO	0x0	Reserved.

DESCRIPTION:

The LCD interface status register that contains read-only views of some parameters or current state of the block.

17.4.14. LCDIF Version Register Description

The LCDIF Version Register can be used to read the version of the LCDIF block being used in this SOC.

HW LCDIF VERSION

0x800300d0

Table 807. HW_LCDIF_VERSION

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
MAJOR								MINOR								STEP															

Table 808. HW_LCDIF_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x2	Fixed read-only value reflecting the MAJOR field of RTL version.
23:16	MINOR	RO	0x0	Fixed read-only value reflecting the MINOR field of RTL version.
15:0	STEP	RO	0x0	Fixed read-only value reflecting the stepping of RTL version.

DESCRIPTION:

The LCDIF Debug 0 Register is for diagnostic use only.

17.4.15. LCDIF Debug 0 Register Description

The LCDIF Debug 0 Register provides a diagnostic view of the state machine and other useful internal signals.

HW LCDIF DEBUG0

0x800300e0

Table 809. HW LCDIF DEBUG0

STREAMING_END_DETECTED	3 1
WAIT_FOR_VSYNC_EDGE_OUT	3 0
SYNC_SIGNALS_ON_REG	2 9
DMACMDKICK	2 8
ENABLE	2 7
HSYNC	2 6
VSYNC	2 5
CUR_FRAME_TX	2 4
EMPTY_WORD	2 3
CUR_STATE	2 2
	2 1
	2 0
	1 9
	1 8
	1 7
	1 6
DATA_COUNT	1 5
	1 4
	1 3
	1 2
	1 1
	1 0
	0 9
	0 8
	0 7
	0 6
	0 5
	0 4
	0 3
	0 2
	0 1
0 0	

STMP3770**Table 810. HW_LCDIF_DEBUG0 Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
31	STREAMING_END_DETECT	RO	0x0	Read-only view of any of the _MODE bits going from 1 to 0.
30	WAIT_FOR_VSYNC_EDGE_OUT	RO	0x0	Read-only view of WAIT_FOR_VSYNC_EDGE bit in the VSYNC mode after it comes out of the FIFO.
29	SYNC_SIGNALS_ON_REG	RO	0x0	Read-only view of internal sync_signals_on_reg signal.
28	DMACMDKICK	RO	0x0	Read-only view of the DMA command kick signal.
27	ENABLE	RO	0x1	Read-only view of ENABLE signal.
26	HSYNC	RO	0x1	Read-only view of HSYNC signal.
25	VSYNC	RO	0x1	Read-only view of VSYNC signal.
24	CUR_FRAME_TX	RO	0x0	This bit is 1 for the time the current frame is being transmitted. When COUNT is bypassed, this signal becomes 0 when WAIT_FOR_VSYNC_EDGE bit is detected to go from 0 to 1. Useful for VSYNC mode debug.
23	EMPTY_WORD	RO	0x1	Indicates that the current word is empty.
22:16	CUR_STATE	RO	0x1	Read-only view of the current state machine state in System interface mode (TX or RX).
15:0	DATA_COUNT	RO	0x0	Read-only view of the current state of the data counter in TX or RX mode.

DESCRIPTION:

This register is for diagnostic use only.

LCDIF Block v2.0

18. TIMERS AND ROTARY DECODER

This chapter describes the timers and rotary decoder included on the STMP3770. Programmable registers are described in [Section 18.4](#).

18.1. Overview

The STMP3770 implements four timers and a rotary decoder, as shown in [Figure 85](#). 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.

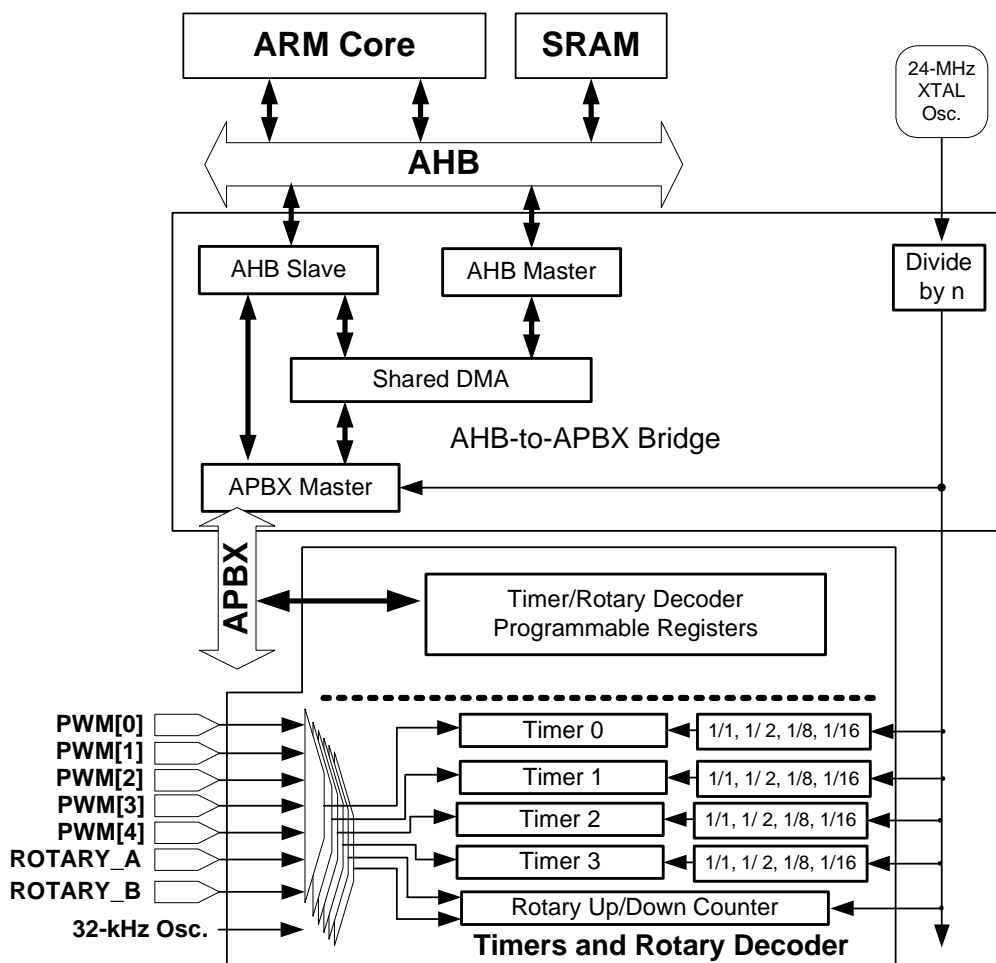


Figure 85. Timers and Rotary Decoder Block Diagram

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 0. When it decrements to 0, it sets an interrupt status bit associated with the counter.

- If the RELOAD bit is set to 1, 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 0.

Figure 86 shows a detailed view of either Timer 0, Timer 1, or Timer 2. Timer 3 has additional functionality, which is shown in Figure 87.

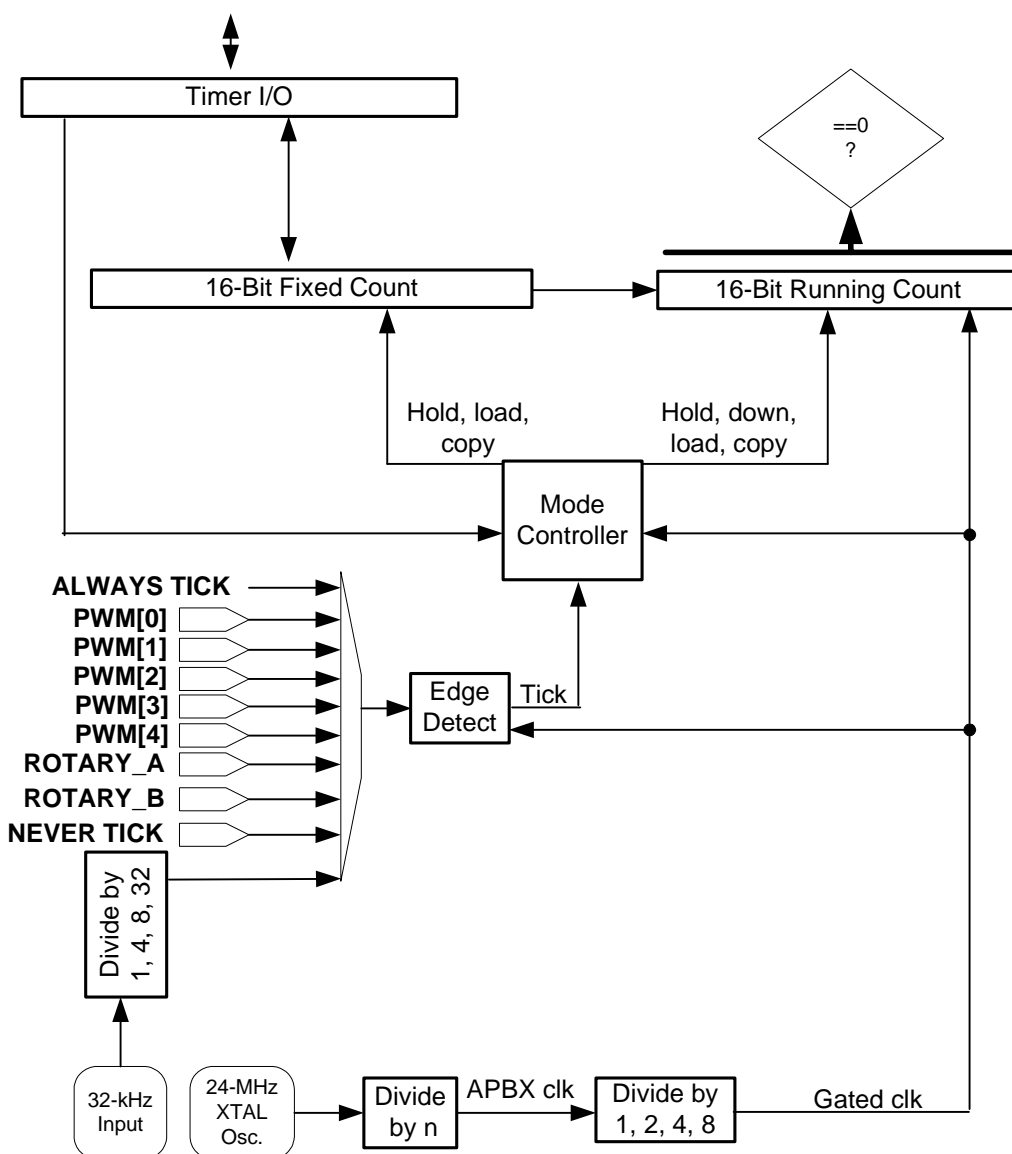


Figure 86. Timer 0, Timer 1, or Timer 2 Detail

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 811 lists the timer state machine transitions.

Table 811. 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 0. When the fixed count has been written with a value of 0 and the running count reaches 0, 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 0, 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 0, 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. Any timer interrupt can be programmed as an FIQ or as a regular IRQ. See [Chapter 5, “Interrupt Collector” on page 75](#) for programming details.

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 33, “Pin Control and GPIO”](#))

on page 929 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 87](#) shows a detailed view of Timer 3.

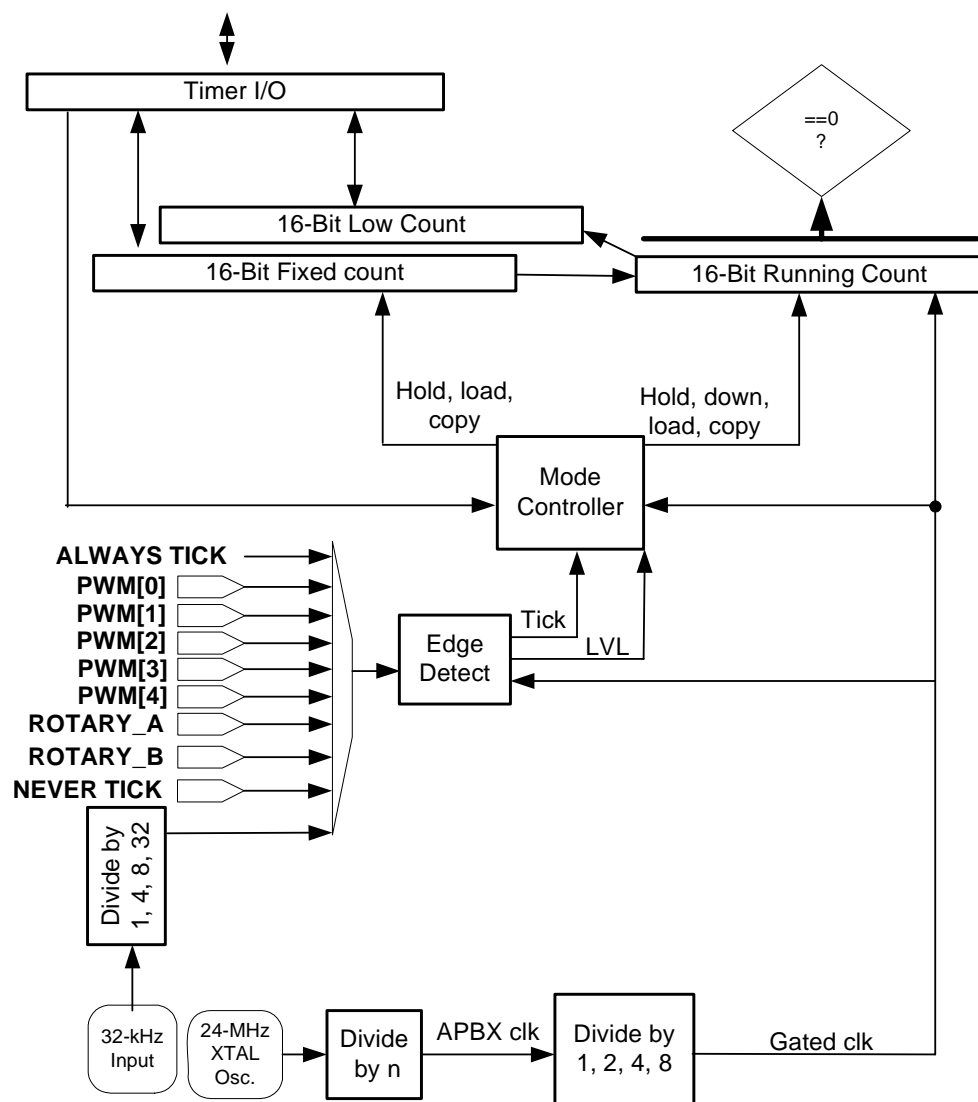


Figure 87. Timer 3 Detail

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 88](#)).

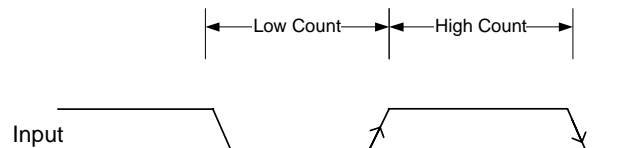


Figure 88. Pulse-Width Measurement Mode

- 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 Timer 3, 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 89](#). It includes a debounce circuit for each input, as shown in [Figure 90](#). This figure shows the debounce circuit for input A, though the circuit is identical for input B.

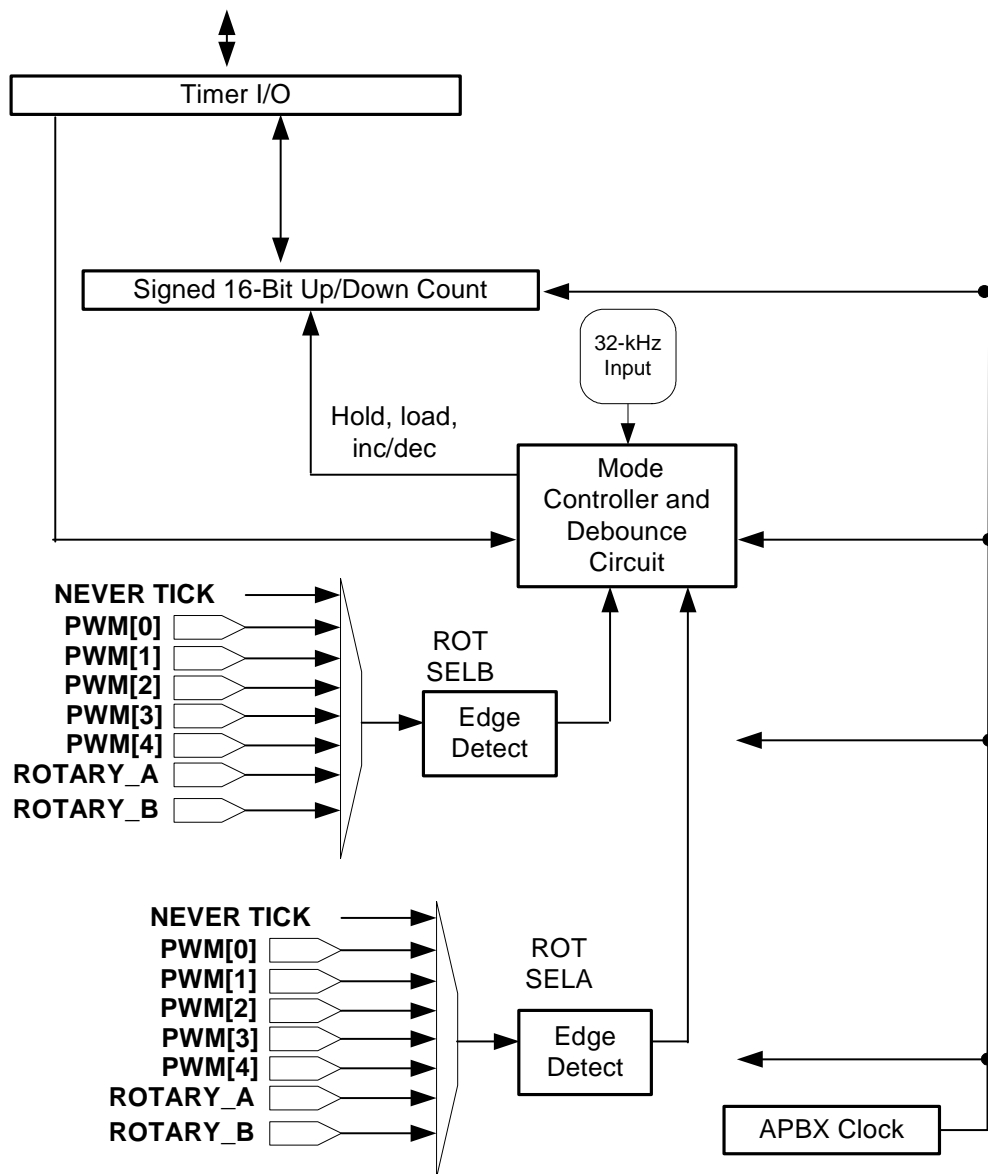


Figure 89. Detail of Rotary Decoder

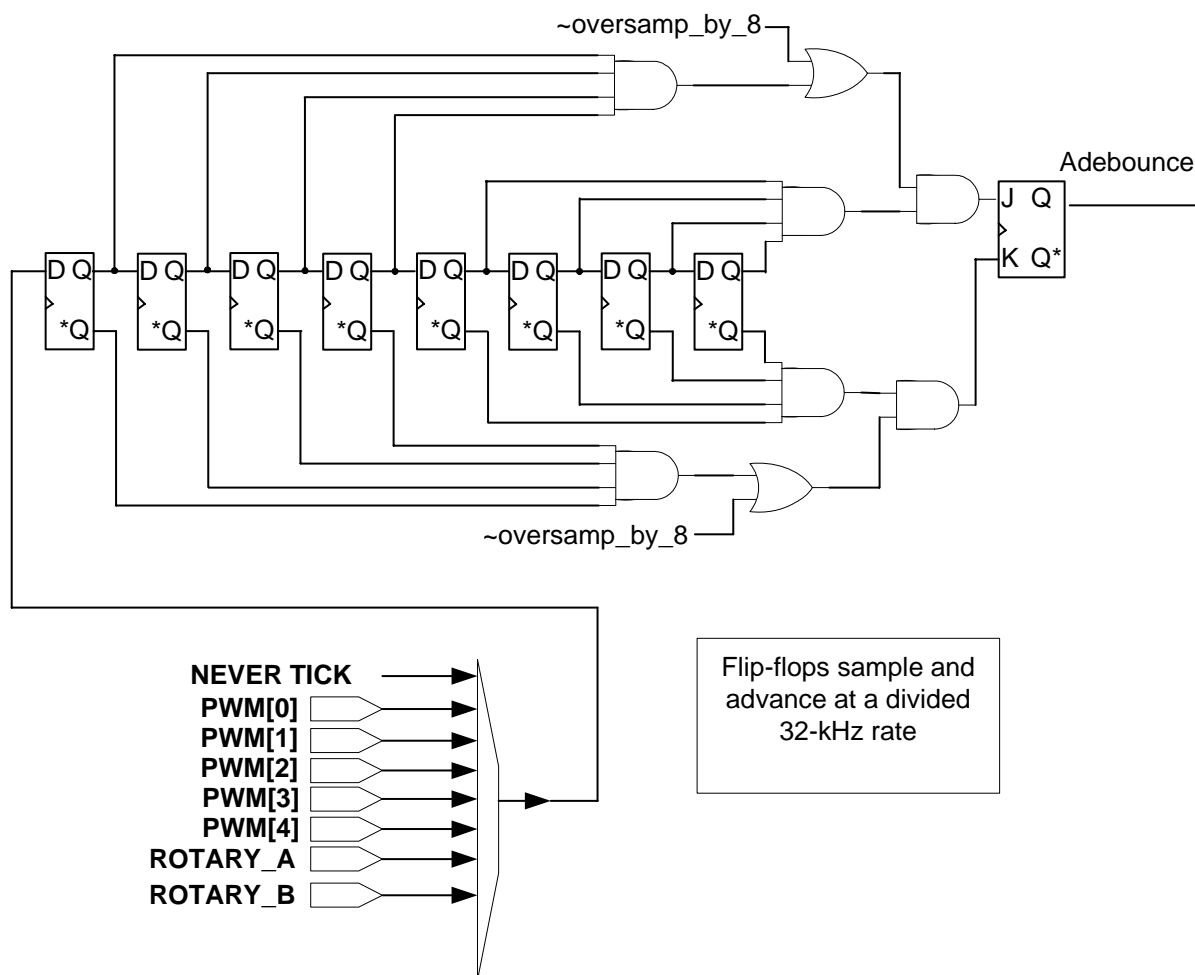


Figure 90. Rotary Decoding Mode—Debouncing Rotary A and B Inputs

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_ROT_COUNT. The updown counter can be treated as either a relative count or an absolute count, depending on the state of the HW_TIMROT_ROT_CTRL_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 91](#)).

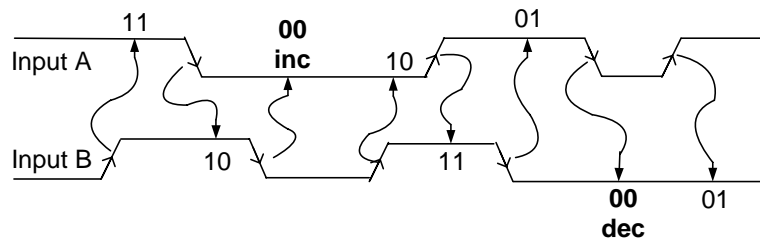


Figure 91. Rotary Decoding Mode—Input Transitions

Figure 91 shows that each detected edge causes a transition in the decoder state machine. Not all transitions are legal (see Table 812). 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 812. 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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013 for additional information on using the SFTRST and CLKGATE bit fields.

Table 814. 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 over-sampling (debouncing) the rotary A and B inputs. Note that the divider value is actually the (value of this field + 1).
15:13	RSVD	RO	0x0	Reserved.
12	RELATIVE	RW	0x0	Set this bit to 1 to cause the rotary decoders up/down counter to be reset to 0 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 1 to invert the input to the edge detector.
8	POLARITY_A	RW	0x0	Set this bit to 1 to invert the input to the edge detector.
7	RSVD	RO	0x0	Reserved.
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	RSVD	RO	0x0	Reserved.
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.

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_ROT_COUNT 0x80068010

Table 818. HW_TIMROT_TIMCTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
14	IRQ_EN	RW	0x0	Set this bit to 1 to enable the generation of a CPU interrupt when the count reaches 0 in normal counter mode.
13:9	RSVD	RO	0x0	Reserved.
8	POLARITY	RW	0x0	Set this bit to 1 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 1 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 1 to cause the timer to reload its current count from its fixed count value whenever the current count decrements to 0. When cleared to 0, the timer enters a mode that freezes at a count of 0. When the fixed count is 0, setting this bit to 1 causes a continuous reload of the fixed count register so that writing 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.

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

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RUNNING_COUNT																FIXED_COUNT															

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 down count. If the RELOAD bit is set to 1, then the new value will be loaded into the running count the next time it reaches 0. If the UPDATE bit is set to 1, then the new value is also copied into the running count immediately. If both the RELOAD and UPDATE bits are cleared to 0, then the new value is never picked up by the running count.

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																IRQ	IRQ_EN	RSVD				POLARITY	UPDATE	RELOAD	PRESCALE	SELECT					

STMP3770



Table 822. HW_TIMROT_TIMCTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	RSVD	RO	0x0	Reserved.
15	IRQ	RW	0x0	This bit is set to 1 when Timer 1 decrements to 0. Write a 0 or use clear SCT mode to clear it.
14	IRQ_EN	RW	0x0	Set this bit to 1 to enable the generation of a CPU interrupt when the count reaches 0 in normal counter mode.
13:9	RSVD	RO	0x0	Reserved.
8	POLARITY	RW	0x0	Set this bit to 1 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 1 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 1 to cause the timer to reload its current count from its fixed count value whenever the current count decrements to 0. When cleared to 0, the timer enters a mode that freezes at a count of 0. When the fixed count is 0, setting this bit to 1 causes a continuous reload of the fixed count register so that writing 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 1.

STMP3770

Table 828. 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 down count. If the RELOAD bit is set to 1, then the new value will be loaded into the running count the next time it reaches 0. If the UPDATE bit is set to 1, then the new value is also copied into the running count, immediately. If both the RELOAD and UPDATE bits are cleared to 0, then the new value is never picked up by the running count.

DESCRIPTION:

This register contains the programmable and read-back counter values for Timer 2.

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 829. HW_TIMROT_TIMCTRL3

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD												TEST_SIGNAL		IRQ	IRQ_EN	RSVD		DUTY_VALID	DUTY_CYCLE	POLARITY	UPDATE	RELOAD	PRESCALE	SELECT							

Table 830. HW_TIMROT_TIMCTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	RSVD	RO	0x0	Reserved.
19:16	TEST_SIGNAL	RW	0x0	<p>Selects the source of the signal to be measured in duty cycle mode.</p> <p>NEVER_TICK = 0x0 Never tick. Freeze the count.</p> <p>PWM0 = 0x1 Input from PWM0.</p> <p>PWM1 = 0x2 Input from PWM1.</p> <p>PWM2 = 0x3 Input from PWM2.</p> <p>PWM3 = 0x4 Input from PWM3.</p> <p>PWM4 = 0x5 Input from PWM4.</p> <p>ROTARYA = 0x6 Input from Rotary A.</p> <p>ROTARYB = 0x7 Input from Rotary B.</p> <p>32KHZ_XTAL = 0x8 Input from 32-kHz crystal.</p> <p>8KHZ_XTAL = 0x9 Input from 8 kHz (divided from 32-kHz crystal).</p> <p>4KHZ_XTAL = 0xA Input from 4 kHz (divided from 32-kHz crystal).</p> <p>1KHZ_XTAL = 0xB Input from 1 kHz (divided from 32-kHz crystal).</p> <p>TICK_ALWAYS = 0xC Always tick.</p>
15	IRQ	RW	0x0	This bit is set to 1 when Timer 3 decrements to 0. Write a 0 or use clear SCT mode to clear it.
14	IRQ_EN	RW	0x0	Set this bit to 1 to enable the generation of a CPU interrupt when the count reaches 0 in normal counter mode.
13:11	RSVD	RO	0x0	Reserved.
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 1 to cause the timer to operate in duty cycle measuring mode.
8	POLARITY	RW	0x0	<p>Set this bit to 1 to invert the input to the edge detector.</p> <p>0 = Positive edge detection.</p> <p>1 = Invert to negative edge detection.</p>
7	UPDATE	RW	0x0	Set this bit to 1 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	<p>Set this bit to 1 to cause the timer to reload its current count from its fixed count value whenever the current count decrements to 0.</p> <p>When cleared to 0, the timer enters a mode that freezes at a count of 0. When the fixed count is 0, setting this bit to 1 causes a continuous reload of the fixed count register so that writing a non-zero value will start the timer.</p>

Table 832. 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	<p>Software loads the fixed count bit field with the value to down count.</p> <p>If the RELOAD bit is set to 1, then the new value will be loaded into the running count the next time it reaches 0.</p> <p>If the UPDATE bit is set to 1, then the new value is also copied into the running count, immediately.</p> <p>If both the RELOAD and UPDATE bits are cleared to 0, then the new value is never picked up by the running count.</p> <p>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.</p>

DESCRIPTION:

This register contains the programmable and read-back counter values for Timer 3. The definitions of the fields change depending whether the timer is in normal or duty cycle mode.

18.4.11. TIMROT Version Register Description

This register indicates the version of the block for debug purposes.

```
HW TIMROT VERSION          0x800680a0
```

Table 833. HW TIMROT VERSION

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
MAJOR								MINOR								STEP															

Table 834. HW_TIMROT_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x01	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

TIMROT Block v1.1

STMP3770



19. REAL-TIME CLOCK, ALARM, WATCHDOG, PERSISTENT BITS

This chapter describes the real-time clock, alarm clock, watchdog reset, persistent bits, and millisecond counter included on the STMP3770. Programmable registers are described in [Section 19.8](#).

19.1. Overview

The real-time clock (RTC) and alarm share a one-second pulse time domain. The watchdog reset and millisecond counter run on a one-millisecond time domain. The RTC, alarm, and persistent bits use persistent storage and reside in a special power domain (crystal domain) that remains powered up even when the rest of the chip is in its powered-down state. [Figure 92](#) 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. The activation of these crystals is under software control through the *RTC persistent bits*, as described later in this chapter.

- The XTAL32KHZ_PWRUP bit in the Persistent Register 0 controls the activity of the 32-kHz crystal at all times (chip power on or off).
- The XTAL24MHZ_PWRUP bit in the Persistent Register 0 controls the behavior of the 24-MHz crystal during the power-off state. (The 24-MHz crystal is always on when the chip is powered up.)

The one-second time base is derived either from the 24.0-MHz crystal oscillator or a 32-kHz crystal oscillator (which can be either exactly 32.0 kHz or 32.768 kHz), as controlled by bits 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 and will continue to count seconds during that time.

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, but 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 (always on) crystal power domain. Shadow versions of these values are maintained in the CPU's power and APBX clock domain when the chip is in the power-up state. When the chip transitions from power-off to power-on, the master values are copied to shadow registers 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 and bits in the HW_RTC_PERSISTENT0 register. The bits in registers HW_RTC_PERSISTENT1 through HW_RTC_PERSISTENT5 are available to store application state information over power-downs and are completely software-defined.

Immediately after reset, it will take approximately three milliseconds 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 0 by the copy controller before reading the initial state of these values (see [Figure 93](#)). The order in which registers are updated is Persistent 0, 1, 2, 3, 4, 5, Alarm, Seconds. (This list is in bitfield order, from LSB to MSB, as they would appear in the STALE_REGS and NEW_REGS bitfields of the HW_RTC_STAT register. For example, the Seconds register corresponds to STALE_REGS or NEW_REGS containing 0x80.)

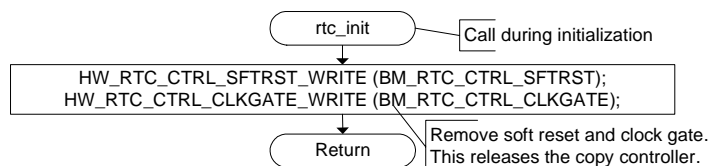


Figure 93. RTC Initialization Sequence

19.2. Programming and Enabling the RTC Clock

The RTC functions implemented in the crystal power domain are referred to as RTC analog functions. The clock frequency and clock source for the RTC analog functions are programmable. There are three possible clock options.

- If the `HW_RTC_PERSISTENT0_CLOCKSOURCE` bit is set to 0, these functions operate on a clock domain derived from the 24.0-MHz crystal oscillator divided by 768 to yield 31.250 kHz.
- If the `HW_RTC_PERSISTENT0_CLOCKSOURCE` bit is 1 and the `HW_RTC_PERSISTENT0_XTAL32_FREQ` bit is 0, then the optional external driving crystal clock will be used and its frequency should be 32.768 kHz.
- However, if the `HW_RTC_PERSISTENT0_CLOCKSOURCE` is 1 and the `HW_RTC_PERSISTENT0_XTAL32_FREQ` bit is also 1, then the external crystal generated clock should be 32.000 kHz for correct operation.

Thus, the `HW_RTC_PERSISTENT0_XTAL32_FREQ` bit gives the systems designer some flexibility as to which external crystal to use on the board.

Switching between these two clock domains is handled by a glitch-free clock mux and can be done on the fly. The 1-Hz time base is derived by dividing either 32.768 kHz by 32768 or by dividing 31.250 kHz by 31,250, or by dividing 32.000 kHz by 32000.

By reading and examining the `HW_RTC_STAT_XTAL32000_PRESENT` and `HW_RTC_STAT_XTAL32768_PRESENT` bits, software can discover if there is an optional crystal clock present and the frequency at which it runs (32.768 kHz or 32.000 kHz). Only one of these fuse bits will be asserted if there is such a crystal attached. If there is no crystal present, both bits will be deasserted.

19.3. RTC Persistent Register Copy Control

The copying of a persistent shadow register (digital) to persistent master storage (analog) occurs automatically. This 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 eight shadow registers can take several milliseconds to complete. Do not attempt to write to more than one shadow register immediately before power down. Whenever possible, software should ensure that the `HW_RTC_STAT_NEW_REGS` field is 0 before powering down the chip or setting the `HW_RTC_CTRL_SFTRST` bit. Otherwise, some of the write data could be lost because the shadow registers could be powered down/reset before the new values can be copied to persistent master storage.

Registers are copied between the digital and analog sides one by one and in 32-bit words. There are no hardwired uses for any of the bits of Persistent registers 1

through 5. And thus, the bits in these registers can be defined and set by software. Persistent Register 0 is reserved for hardware programming and configuration.

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_NEW_REGS is a 0. 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.

NOTE: The HW_RTC_CTRL_SUPPRESS_COPY2ANALOG diagnostic bit is never set while any copy or update operation is underway. Doing so will result in undefined operation of the copy controller.

Figure 94 shows the copy and test procedure for a single register. However, software can write to any register whose HW_RTC_STAT_NEW_REGS bit is 0 even if the copy controller is currently busy copying a different register. For example, if the copy controller is busy copying Persistent Register 1 from a previous write, software can simultaneously write to Persistent Register 0 during this copy. After the copy controller has finished copying register 1, it will then, in turn, begin the copy process for the new value of Register 0. Thus, registers can thus be written in any order. Again, the main important rule that must be followed is that the HW_RTC_STAT_NEW_REGS bit for a particular register must be 0 before it can be written and is independent of the state of the HW_RTC_STAT_NEW_REGS bits for the other registers.

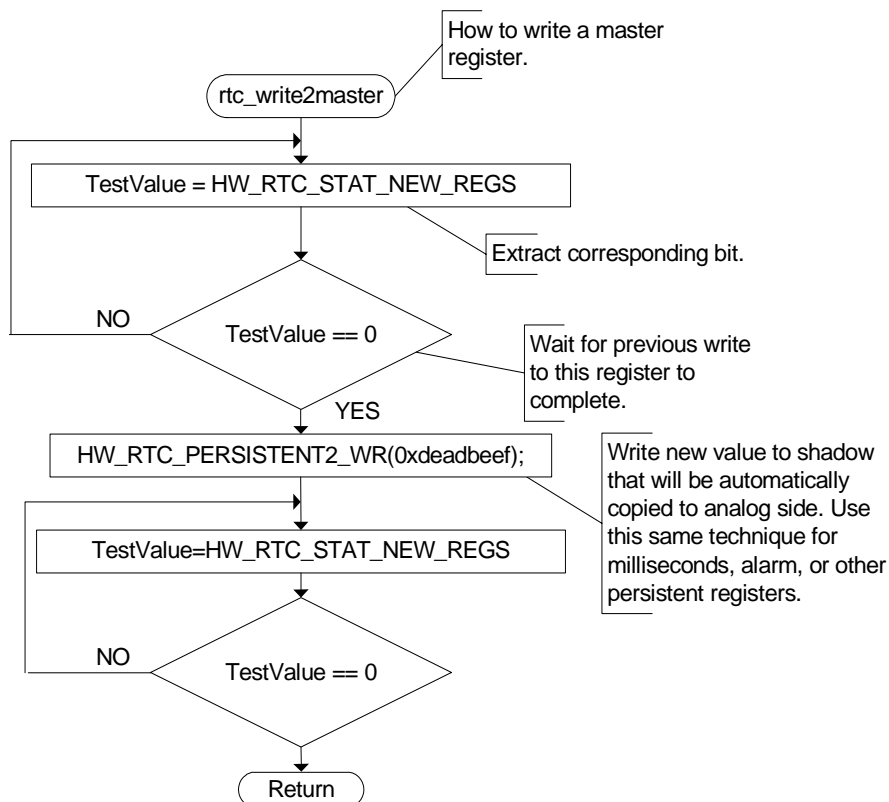


Figure 94. RTC Writing to a Master Register from CPU

The digital shadow registers will be updated (copied from analog to digital) with values from their analog persistent counterparts under two conditions: first, whenever the chip is powered on, and secondly, whenever the HW_RTC_CTRL_FORCE_UPDATE bit is set by software. Then, an update occurs, and all persistent registers are updated. Persistent registers cannot be updated singly.

19.4. Real-Time Clock Function

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 clock or the 32-kHz clock, as determined by writable bit values in the RTC Persistent Register 0.

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 when a voltage is first applied to the BATT pin (battery insertion).

For consistency across applications, it is recommended that the seconds timer should be referenced to January 1, 1980 at a 32-bit value of 0 (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.

19.4.1. 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 34.4.10, "Correct Way to Soft Reset a Block" on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

19.5. Millisecond Resolution Timing Function

A millisecond counter facility is provided based on a 1-kHz signal derived from 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. The programmer can change the resolution of the millisecond counter to be 1, 2, 4, 8, or 16 milliseconds. This is programmed through bits in Persistent Register 0.

WARNING: When the 32.768-kHz or 32.000-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.

19.6. Alarm Clock Function

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. 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. Although these two functions can be enabled at the same time, one should remember that the CPU will only be interrupted if the chip is powered up at the time of the alarm event.

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 cause of the wake-up 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 in the Persistent register 0 is visible in the digital shadow register. If an alarm wake event occurs while the chip is powered up, the ALARM_WAKE bit will not be set in the persistent register because the chip was not woken up.

The alarm must be “present” on an actual chip to perform this function (see the HW_RTC_STAT_ALARM_PRESENT bit description).

19.7. Watchdog Reset Function

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 watchdog generates this reset if software does not rewrite this register before this time elapses.

The watchdog timer decrements the register value once for every tick of the 1-kHz clock supplied from the RTC analog section (see [Figure 92](#)). 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 (analog persistent storage).

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, and it must be reprogrammed after cycling power or resetting the block.

The watchdog timer must be “present” on an actual chip to perform this function (see the HW_RTC_STAT_WATCHDOG_PRESENT bit description).

19.8. 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.

19.8.1. Real-Time Clock Control Register Description

HW_RTC_CTRL is the control register for the RTC.

HW_RTC_CTRL	0x8005C000
HW_RTC_CTRL_SET	0x8005C004
HW_RTC_CTRL_CLR	0x8005C008
HW_RTC_CTRL_TOG	0x8005C00C

Table 835. HW_RTC_CTRL

SFTRST	3 1																									SUPPRESS_COPY2ANALOG					
CLKGATE	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
																										FORCE_UPDATE					
																										WATCHDOGEN					
																										ONEMSEC_IRQ					
																										ALARM_IRQ					
																										ONEMSEC_IRQ_EN					
																										ALARM_IRQ_EN					

Table 836. 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.
30	CLKGATE	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block. This bit has no effect on RTC analog.
29:7	RSVD	RO	0x0	Reserved.
6	SUPPRESS_COPY2ANALOG	RW	0x0	This bit is used for diagnostic purposes. 0 = Normal operation. Use SCT writes to set, clear, or toggle. 1 = Suppress the automatic copy that normally occurs to the analog side, whenever a shadow register is written.
5	FORCE_UPDATE	RW	0x1	This bit is how the software requests the update of the shadow registers from values in RTC analog. When software sets this bit, all eight of the shadow registers are updated from the corresponding values in the persistent registers in RTC analog. The state of this update operation is reflected on the STALEREFS bits on the STAT register, which are set to all ones upon an update request and are cleared by hardware as the update proceeds. Hardware clears this bit immediately after detecting it has been set. Software does not need to clear. Software must NOT look to the state of this bit to determine the status of the update operation. Instead, it must look to the STALEREG bits in the STAT register to determine when any given register has been updated and/or when the update operation is complete. Notice that the default value of this bit is 1, so that a reset (either chip-wide or soft) always results in an update.
4	WATCHDOGEN	RW	0x0	1 = Enable Watchdog Timer to force chip wide resets. Use SCT writes to set, clear, or toggle.

Table 840. HW_RTC_MILLISECONDS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	COUNT	RW	0x00000000	32-bit milliseconds counter.

DESCRIPTION:

HW_RTC_MILLISECONDS provides access to the 32-bit millisecond 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 based on a pulse from RTC analog that is 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.
```

19.8.4. 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 841. HW_RTC_SECONDS

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
COUNT																															

Table 842. 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 0x0 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 0 only 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.

Note that if a low frequency (32.000-kHz or 32.768-kHz) crystal is available in the system, then the seconds count and the milliseconds count will be derived from different clocks. Namely, the seconds count is derived from the low-frequency crystal

Table 848. HW_RTC_PERSISTENT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:18	SPARE_ANALOG	RW	0x0	Reserved for special uses.
17	AUTO_RESTART	RW	0x0	Set to 1 to enable the chip to automatically power up approximately 180 ms after powering down.
16	DISABLE_PSWITCH	RW	0x0	Disables the PSWITCH pin startup functionality unless the voltage on the PSWITCH pin goes above the VDDXTAL pin voltage by a threshold voltage. Typically, this voltage is created by pulling PSWITCH up with a current limiting resistor to a higher voltage, such as the Li-Ion battery voltage.
15:14	LOWERBIAS	RW	0x0	Reduce bias current of 24-MHz crystal. 00 = Nominal 01 = -25% 10 = -25% 11 = -50%
13	DISABLE_XTALOK	RW	0x0	Set to 1 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 stops oscillating for any reason.
12:8	MSEC_RES	RW	0x1	This bit field encodes the value of the millisecond count resolution in a one-hot format. Resolutions supported are: 1, 2, 4, 8, and 16 milliseconds. The bit field directly gives the resolution without need for decode.
7	ALARM_WAKE	RW	0x0	Set when the chip is powered up by an alarm event from rtc_ana. Can then be cleared by software as desired.
6	XTAL32_FREQ	RW	0x0	If CLOCKSOURCE (bit 0 of this register) is 1, then this bit gives the exact frequency of the 32-kHz crystal. If this bit is 1 = frequency is 32.768 kHz. If it is 1, the frequency is 32.000 kHz. If CLOCKSOURCE is 0, the value of this bit is immaterial.
5	XTAL32KHZ_PWRUP	RW	0x0	Set to 1 to power up the 32-kHz crystal oscillator. Set to 0 to disable the oscillator. This bit controls the oscillator at all times (chip powered on or not).
4	XTAL24MHZ_PWRUP	RW	0x0	Set to 1 to keep the 24.0-MHz crystal oscillator powered up while the chip is powered down. Set to 0 to disable during chip power down. Note: The oscillator is always on while the chip is powered on.
3	LCK_SECS	RW	0x0	Set to 1 to lock down the seconds count. Once this bit is written with a 1, the user will not be able either to write to the seconds register or to change this bit back to a 0—except by removing the battery.

Table 848. HW_RTC_PERSISTENT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	ALARM_EN	RW	0x0	Set this bit to 1 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 bit[2] to be written and persistent bit[2] will always read back 0, regardless of the values in the shadow register.
1	ALARM_WAKE_EN	RW	0x0	This bit is set to 1 to upon the arrival of an alarm event that powers up the chip. ALARM_EN must be set to 1 to enable the detection of an alarm event. This bit is reset by writing a 0 directly to the shadow register, which causes the copy controller to move it across to the analog domain.
0	CLOCKSOURCE	RW	0x0	Set to 1 to select the 32-kHz crystal oscillator as the source for the 32-kHz clock domain used by the RTC analog domain circuits. Clear to 0 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 general persistent bits are available for software use. The register initializes 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
```

19.8.8. Persistent State Register 1 Description

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states.

```
HW_RTC_PERSISTENT1      0x8005C070
HW_RTC_PERSISTENT1_SET  0x8005C074
HW_RTC_PERSISTENT1_CLR  0x8005C078
HW_RTC_PERSISTENT1_TOG  0x8005C07C
```

Table 849. HW_RTC_PERSISTENT1

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3
GENERAL																												

Table 850. HW_RTC_PERSISTENT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	GENERAL	RW	0x0000	Firmware use, defined as follows: ENUMERATE_500MA_TWICE = 0x1000 Enumerate at 500 mA twice before dropping back to 100 mA. USB_BOOT_PLAYER_MODE = 0x0800 Boot to player when connected to USB. SKIP_CHECKDISK = 0x0400 Run Checkdisk flag. USB_LOW_POWER_MODE = 0x0200 USB Hi/Lo Current select. OTG_HNP_BIT = 0x0100 HNP has been required if set to 1. OTG_ATL_ROLE_BIT = 0x0080 USB role.

DESCRIPTION:

The register initializes 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
```

19.8.9. Persistent State Register 2 Description

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states.

```
HW_RTC_PERSISTENT2      0x8005C080
HW_RTC_PERSISTENT2_SET  0x8005C084
HW_RTC_PERSISTENT2_CLR  0x8005C088
HW_RTC_PERSISTENT2_TOG  0x8005C08C
```

Table 851. HW_RTC_PERSISTENT2

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
GENERAL																															

Table 852. HW_RTC_PERSISTENT2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	GENERAL	RW	0x00000000	FIRMWARE/SOFTWARE

DESCRIPTION:

The register initializes 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
```

19.8.10. Persistent State Register 3 Description

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states.

```
HW_RTC_PERSISTENT3      0x8005C090
HW_RTC_PERSISTENT3_SET  0x8005C094
HW_RTC_PERSISTENT3_CLR  0x8005C098
```


20. PULSE-WIDTH MODULATOR (PWM) CONTROLLER

This chapter describes the pulse-width modulator (PWM) controller included on the STMP3770 and how to use it. Programmable registers are described in [Section 20.4](#).

20.1. Overview

The STMP3770 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 0, 1, or high-impedance 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 95](#) 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_ACTIVEN and HW_PWM_PERIODn.

When programming a channel, it is important to remember that there is an order dependence for register writes.

- The HW_PWM_ACTIVEN register must be written first, followed by HW_PWM_PERIODn.
- If the order is reversed, the parameters written to the HW_PWM_ACTIVEN register will not take effect in the hardware.

The hardware waits for a HW_PWM_PERIODn 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.

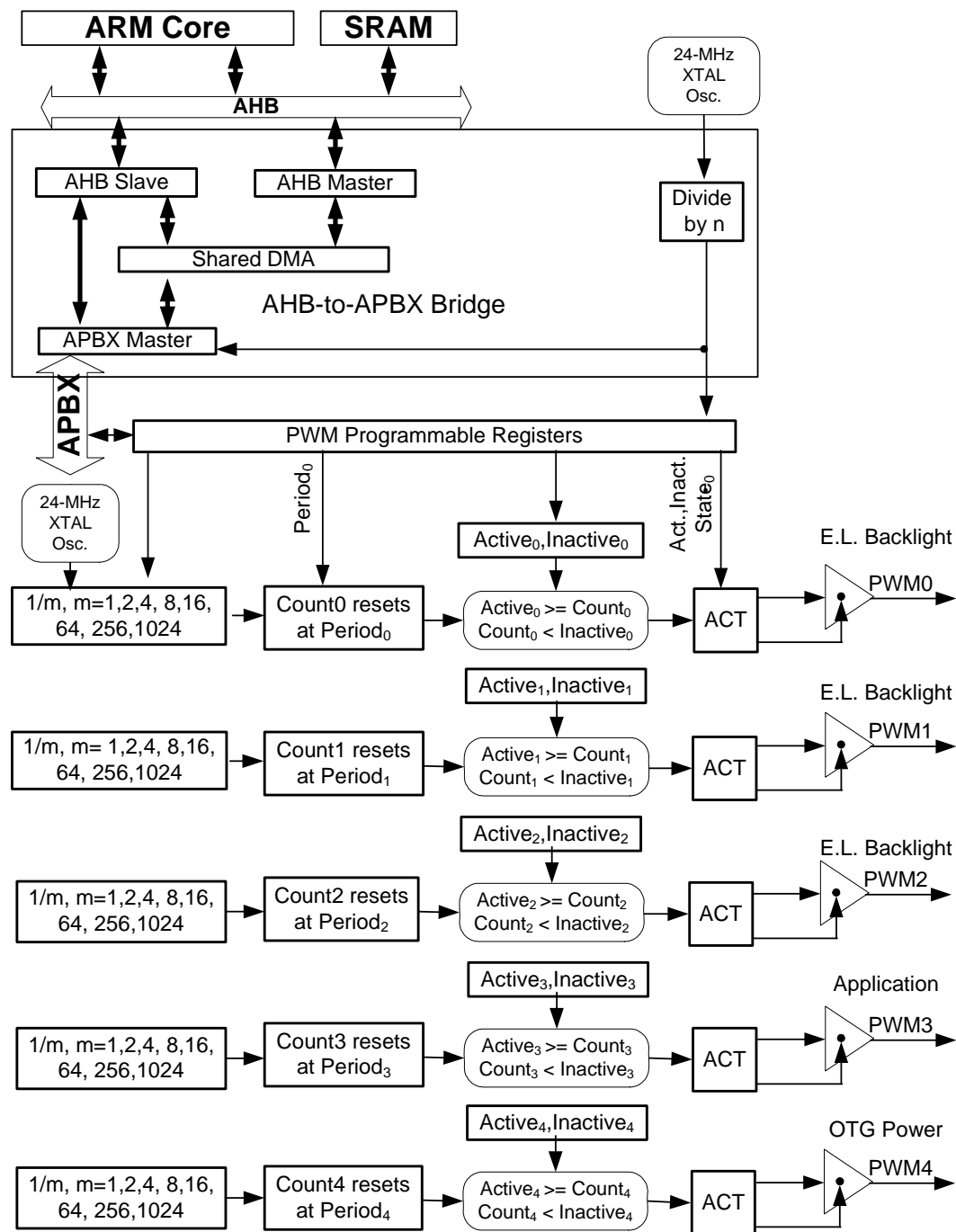


Figure 95. Pulse-Width Modulation Controller (PWM) Block Diagram

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 1 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 96, 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.

It should be noted that if the clock gate is enabled during normal operation, the PWM channel outputs will remain in the state a(ctive or inactive), that they were in at the time of the clock gate assertion. If the channel is in the active state at the time, it will not return to the resting or inactive state. Later, when the clock gate is de-asserted, operation will resume where it left off. This is important to remember because some external devices may not be able to tolerate a “high” output level for extended periods of time.

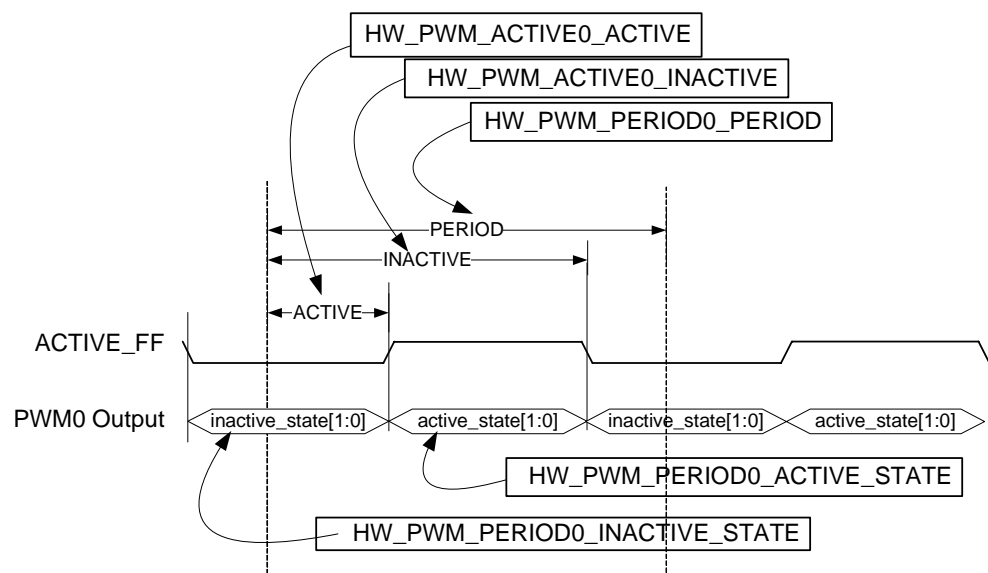


Figure 96. PWM Output Example

The actual values obtainable on the output are shown in Figure 97. Notice that one possible state is to turn off the output driver to provide a high-impedance 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 impedance. 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 97](#), a differential pair is established using Channel 0 and Channel 1. 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-impedance state.

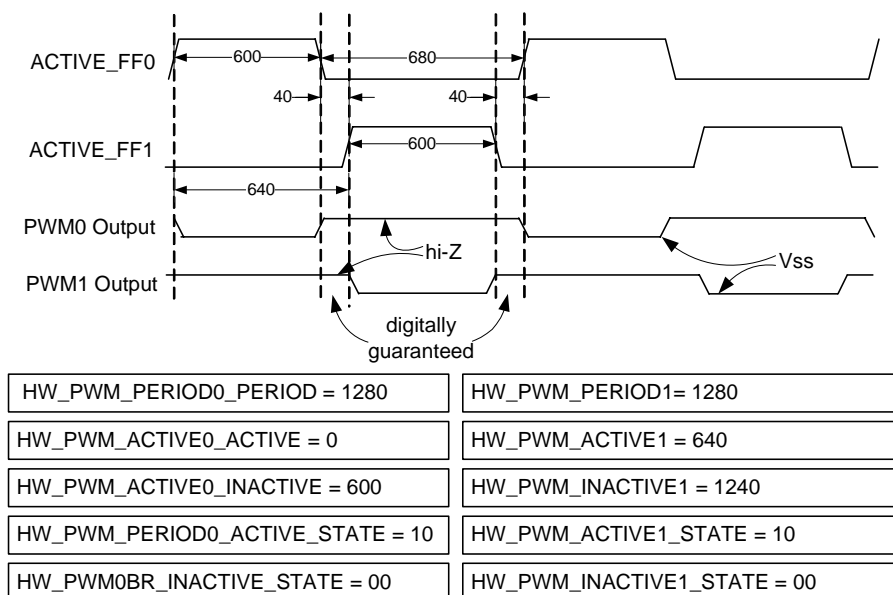


Figure 97. PWM Differential Output Pair Example

[Figure 98](#) 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 1. The output pin can be set to a 0, a 1, 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.

20.2.1. Multi-Chip Attachment Mode

The multi-chip attachment mode (MATT) allows a 24-MHz crystal clock that is an input to the STMP3770 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 98](#).

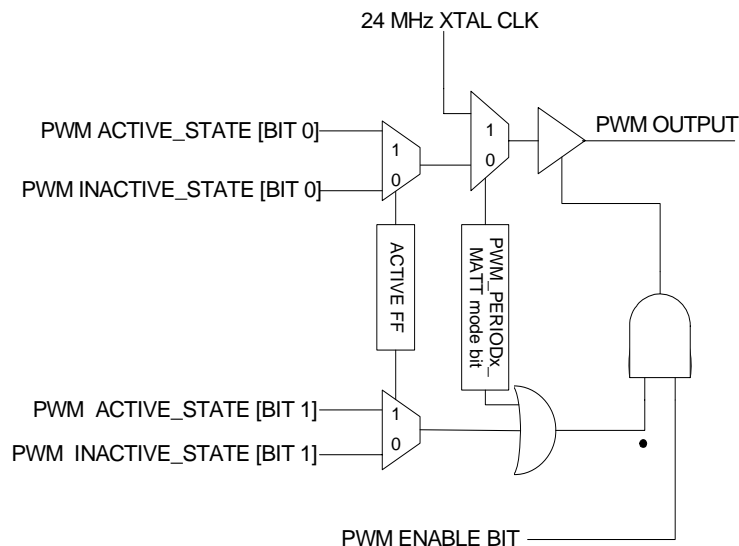


Figure 98. PWM Output Driver

20.2.2. Channel 2 Analog Enable Function

The output generation for channel 2 is slightly different than shown in Figure 98. In this channel, there is an additional enable that is controlled from the analog LRADC block. This signal is synchronized in the XTAL domain and ANDed with the PWM ENABLE bit. So, in this case, either enable source can disable the output. Also, this analog enable control signal can be disabled through the PWM2_ANA_CTRL_ENABLE bit in the HW_PWM_CTRL register. When disabled, Channel 2 behaves identically to the other channels.

20.2.3. Analog Feedback for Backlight Control Using PWM Channel 2

Figure 99 shows a path for feedback control of PWM Channel 2 to allow for fine backlight current control.

Both HW_LRADC_CTRL2_BL_ENABLE and HW_PWM_CTRL_PWM2_ANA_CTRL_ENABLE must be set to activate this path. HW_LRADC_CTRL2_BL_MUX_SELECT chooses whether pin LRADC1 or LRADC4 is used for the analog feedback input. The LRADC itself is not used for this function; only the input pin is used.

There is a gain-of-4 stage that can be optionally bypassed with HW_LRADC_CTRL2_BL_AMP_BYPASS. To give fine control of the backlight brightness, the trip-point of the comparator can be adjusted across a 40-dB range using HW_LRADC_CTRL2_BL_BRIGHTNESS. In this mode, the PWM Channel 2 will be driving a boost converter connected to the battery. The STMP3770 includes special protection circuitry to detect if the clocks stop to ensure that the boost converter does not get stuck in a high current state—similar to the on-chip DC-DC converter protections.

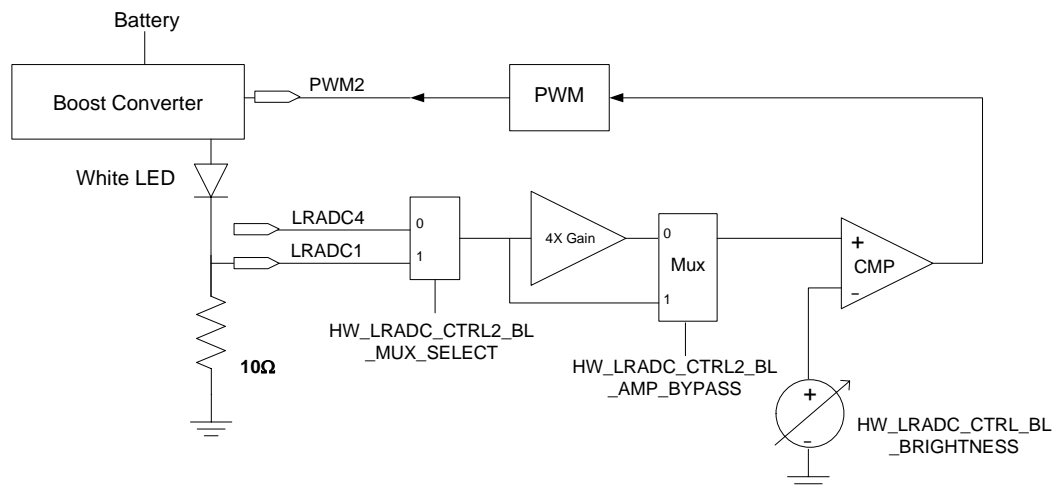


Figure 99. Backlight Current Control

20.3. 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 34.4.10, "Correct Way to Soft Reset a Block" on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

20.4. Programmable Registers

The following registers are available for CPU programmer access and control of the PWM controller.

20.4.1. PWM Control and Status Register Description

The PWM Control and Status Register 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 863. HW_PWM_CTRL

SFTRST	3	1																													PWM2_ANA_CTRL_ENABLE	0	5			0	4			0	3			0	2			0	1			0	0
CLKGATE	3	0																													PWM4_ENABLE	0	4			0	3			0	2			0	1			0	0				
PWM4_PRESENT	2	9																													PWM3_ENABLE	0	3			0	2			0	1			0	0								
PWM3_PRESENT	2	8																													PWM2_ENABLE	0	7			0	6			0	5			0	4								
PWM2_PRESENT	2	7																													PWM1_ENABLE	0	6			0	5			0	4			0	3								
PWM1_PRESENT	2	6																													PWM0_ENABLE	0	5			0	4			0	3			0	2								
PWM0_PRESENT	2	5																																																			
RSVD	2	4	2	3	2	2	1	0	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0						
	2	3	2	2	1	0	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0								
	2	2	2	1	0	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0									
	2	1	2	1	0	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0									
	2	0	2	1	0	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0									
	1	9	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0									
	1	8	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0											
	1	7	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0											
	1	6	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0											
	1	5	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
	1	4	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
	1	3	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
	1	2	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
	1	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
	1	0	0	9	0	8	0	7	0	6	0	5	0	4	0	3	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

Table 864. HW_PWM_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it forces a block-wide reset.
30	CLKGATE	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, 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.
24:6	RSVD	RO	0x0	Reserved.
5	PWM2_ANA_CTRL_ENABLE	RW	0x0	Setting this bit to 1 enables PWM Channel 2 output to be enabled by analog circuitry outside this block much the way the PWM2_ENABLE bit controls it. Note that the PWM2_ENABLE bit must still be set and channel must still be programmed normally for the PWM output to cycle. The PWM2_ENABLE bit acts like a master switch in this context.
4	PWM4_ENABLE	RW	0x0	Enables PWM channel 4 to begin cycling when set to 1. To enable PWM4 onto the output pin, the pin control registers must programmed accordingly.
3	PWM3_ENABLE	RW	0x0	Enables PWM channel 3 to begin cycling when set to 1. To enable PWM3 onto the output pin, the pin control registers must programmed accordingly.
2	PWM2_ENABLE	RW	0x0	Enables PWM channel 2 to begin cycling when set to 1. To enable PWM2 onto the output pin, the pin control registers must programmed accordingly.

20.4.2. PWM Channel 0 Active Register Description

HW_PWM_ACTIVE0	0x80064010
HW_PWM_ACTIVE0_SET	0x80064014
HW_PWM_ACTIVE0_CLR	0x80064018
HW_PWM_ACTIVE0_TOG	0x8006401C

Table 865. HW PWM ACTIVE0

Table 866. HW_PWM_ACTIVE0 Bit Field Descriptions

5-37xx-DS2-1.04-031408

Table 872. HW_PWM_PERIOD1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
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-impedance. HI_Z = 0x0 Inactive state sets PWM output to high-impedance. 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-impedance. HI_Z = 0x0 Active state sets PWM output to high-impedance. 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 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

Table 876. HW_PWM_PERIOD2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23	MATT	RW	0x0	Multichip Attachment <u>mode</u> . 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-impedance. HI_Z = 0x0 Inactive state sets PWM output to high-impedance. 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-impedance. HI_Z = 0x0 Active state sets PWM output to high-impedance. 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 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

20.4.8. 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 877. HW_PWM_ACTIVE3

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
INACTIVE																ACTIVE															

STMP3770



21. I²C INTERFACE

This chapter describes the I²C interface implemented on the STMP3770. It includes sections on the external pins, interrupt sources, I²C bus protocol, and programming examples. Programmable registers are included in [Section 21.4](#).

21.1. Overview

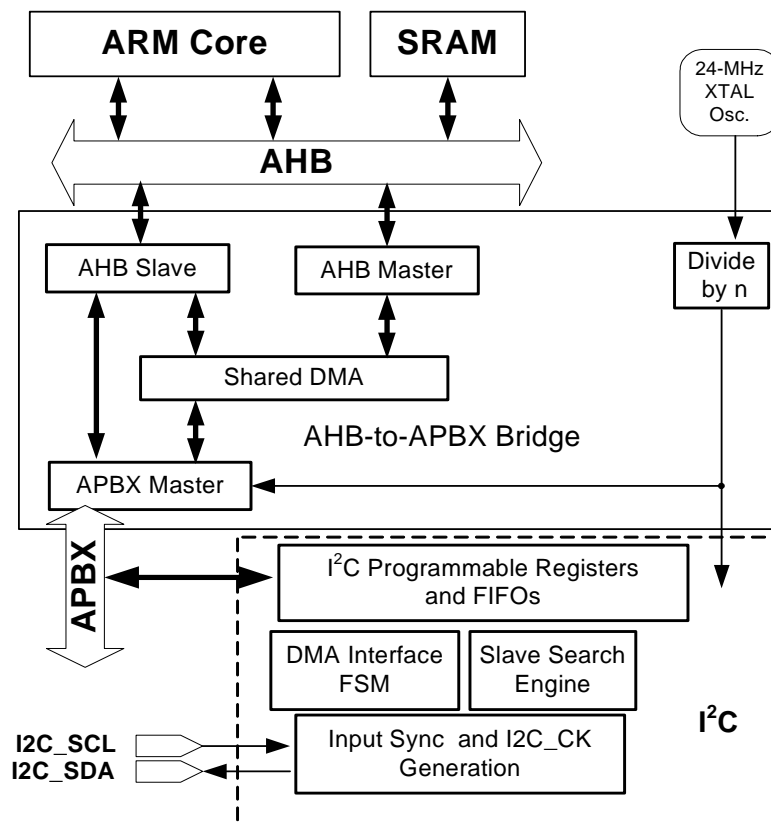
The I²C is a standard two-wire serial interface used to connect the chip with peripherals or host controllers. This interface provides a standard speed (up to 100 kbps), and a fast speed (up to 400 kbps) I²C connection to multiple devices with the chip acting in either I²C master or I²C slave mode. Typical 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 STMP3770, 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 100](#) shows a block diagram of the I²C interface implemented on the STMP3770.

Figure 100. I²C Interface Block Diagram

21.2. Operation

The I²C Interface on the STMP3770 includes the following external pins:

- **I2C_SDA: 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 2k Ω resistors are used to pull the signals up to VddIO for normal and fast speeds.

21.2.1. 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 887](#).

Table 887. I²C Slave and Master Interrupt Condition in HW_I2C_CTRL1

SOURCE	BIT NAME	DESCRIPTION
Slave Address	SLAVE_IRQ	This interrupt is generated when an address match occurs. It indicates that the CPU should read the captured RW bit from the I ² C address byte to determine the type of DMA to use for the data transfer phase.
Slave Stop	SLAVE_STOP_IRQ	This interrupt is generated when a stop condition is detected after a slave address has been matched.
Oversize Xfer	OVERSIZE_XFER_TERM_IRQ	The DMA and I ² C 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	EARLY_TERM_IRQ	The DMA and I ² C 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	MASTER_LOSS_IRQ	A master begins transmission on an idle I ² C bus and monitors the data line. If it ever attempts to send a 1 on the line and notes that a 0 has been sent instead, then it notes that it has lost mastership of the I ² C 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	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 I ² C protocol is processed, and the I ² C bus returns to the idle state.
Data Engine Complete	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	BUS_FREE_IRQ	When bus mastership is lost during the I ² C 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.

21.2.2. I²C Bus Protocol

The I²C interface operates as shown in [Figure 101](#) and [Figure 102](#).

- 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 0x86, and its default read address is 0x87. 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.

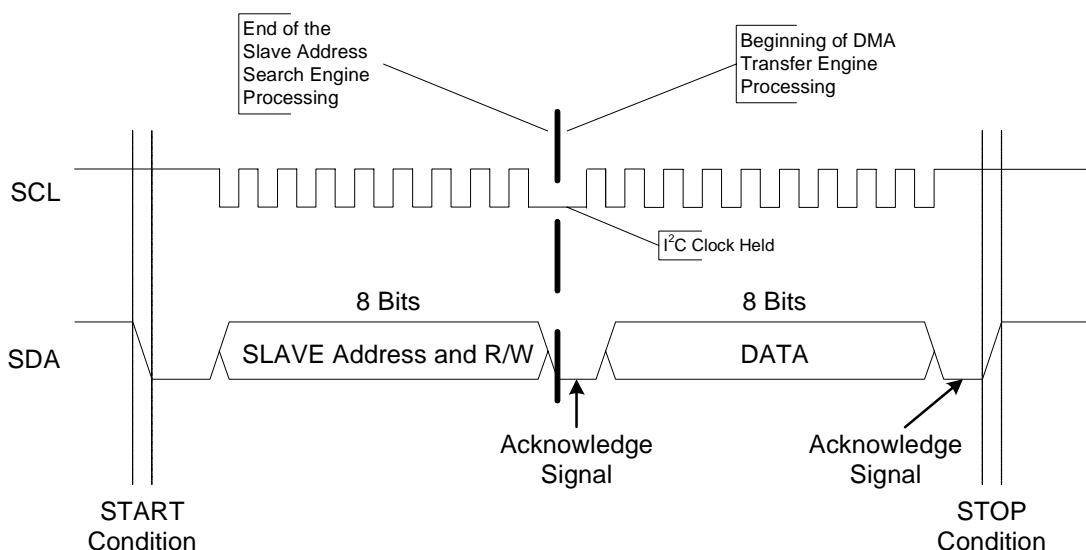
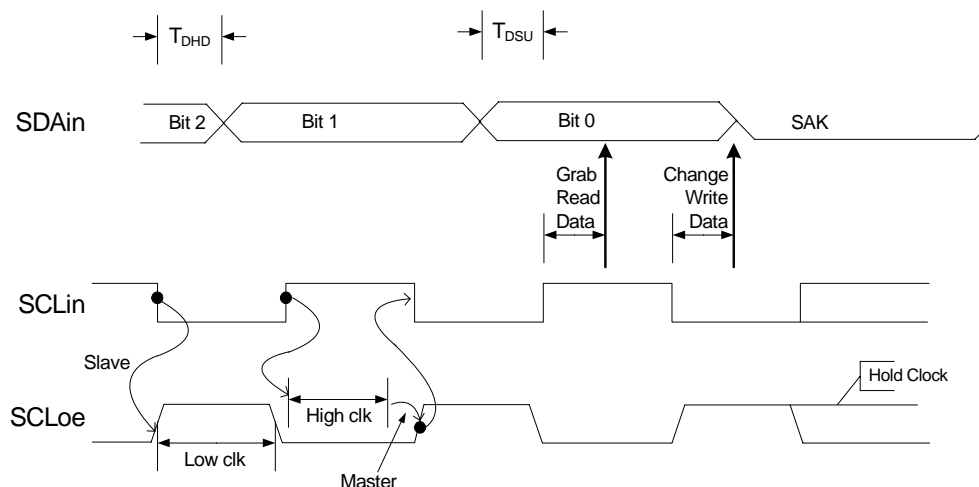


Figure 101. I²C Data and Clock Timing

The clock is generated by the master, according to parameters set in the HW_I2C_TIMINGn register. This register also provides programmable timing for capturing received data, as well as for changing the transmitted data bit.


Figure 102. I²C Data and Clock Timing Generation

21.2.2.1. 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 888. I²C Transfer When the Interface is Transmitting as a Master

ST	SAD+W	SAK	DATA	SAK	SP
----	-------	-----	------	-----	----

[Table 889](#) 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 889. 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 890. 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 891. I²C Transfer “FM Tuner” Read of Three Bytes

ST	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP
----	-------	-----	------	-----	------	-----	------	------	----

21.2.2.2. Typical EEPROM Transactions

I²C EEPROMs typically have a specific transaction sequence for reading and writing data bytes to and from the EEPROM array. [Table 892](#) through [Table 895](#) 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 889](#) 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 892. 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 893. 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 894. 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 895. 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
----	-------	-----	-----	-----	-----	-----	----	-------	-----	------	-----	------	-----	------	------	----

21.2.2.3. Master Mode Protocol

In master mode, the I²C interface generates the clock and initiates all transfers.

21.2.2.4. 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.

21.2.2.5. Master Mode Operation

The finite state machine for master mode operation is shown in [Figure 103](#) through [Figure 106](#). [Figure 103](#) shows the generation of the optional start condition. [Figure 104](#) shows the receive states, [Figure 105](#) shows the transmit states. [Figure 106](#) shows the generation of the optional stop state.

[Table 896](#) through [Table 899](#) show examples of Master Mode I²C transactions. [Table 889](#) defines each sub-address shown. The following read-after-write transactions are performed using the restart technique.

Table 896. 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 897. I²C Transfer When the Interface as Master is Receiving >1 Byte of Data from Slave

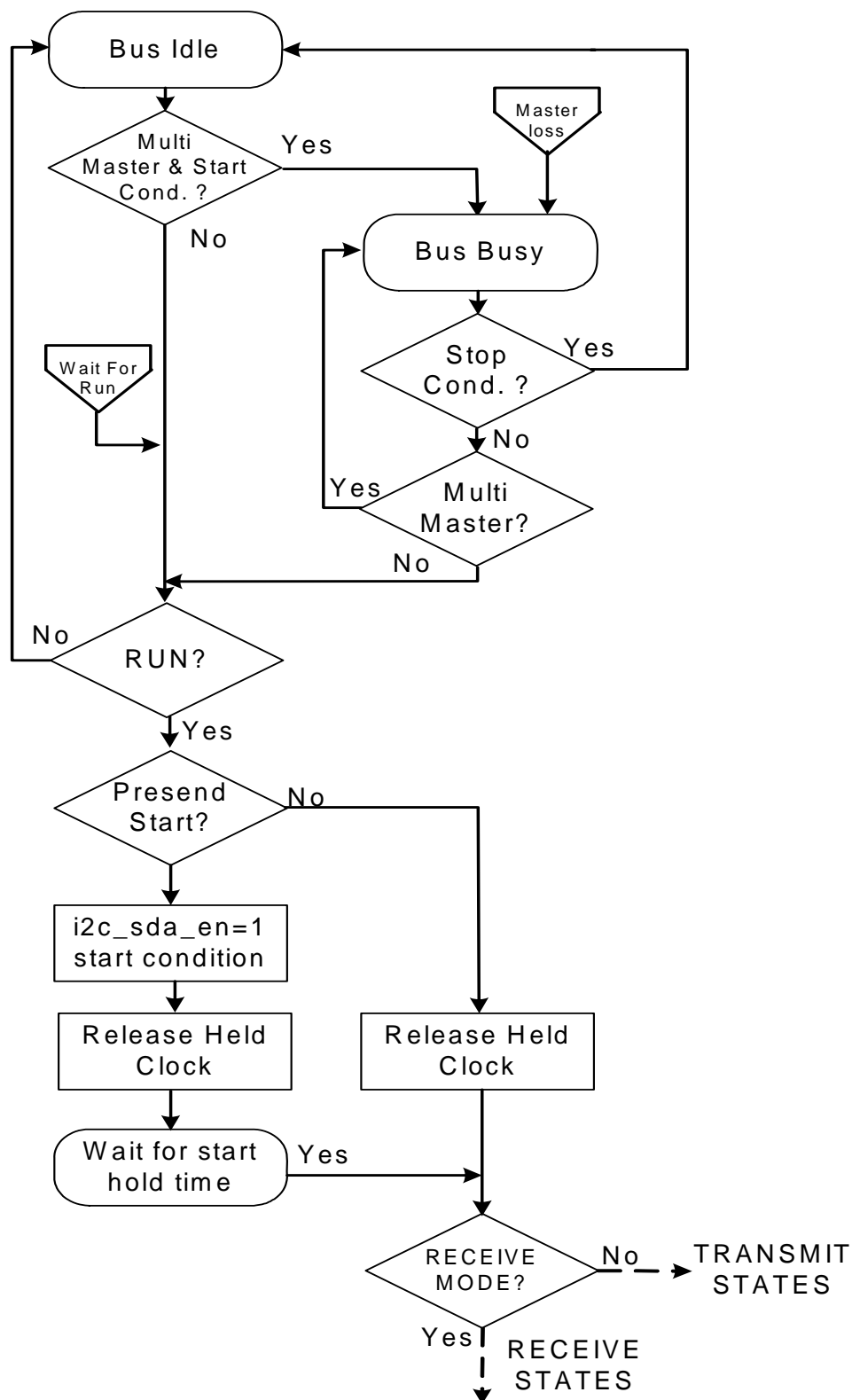
ST	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP
----	-------	-----	------	-----	------	-----	------	------	----

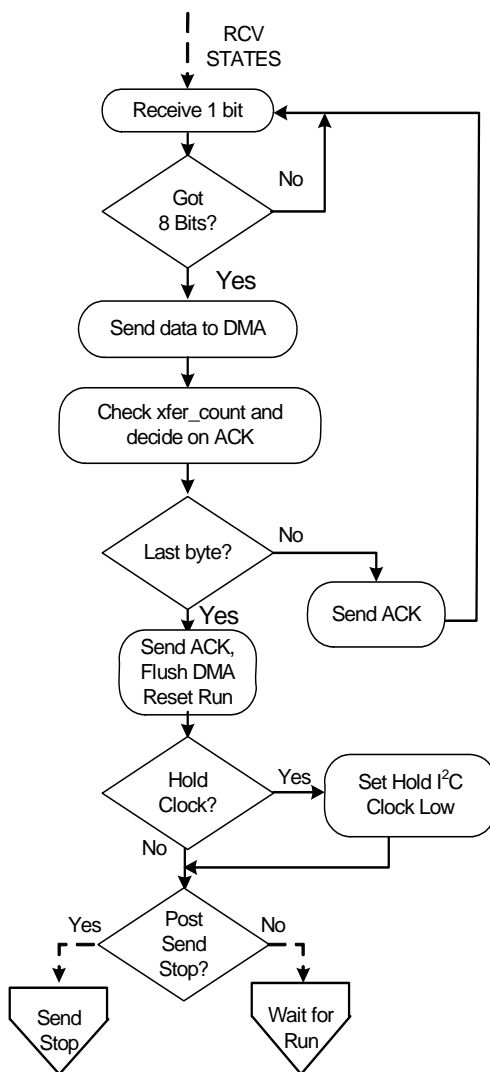
Table 898. I²C Transfer when Master is Receiving 1 Byte of Data from Slave Internal Subaddress

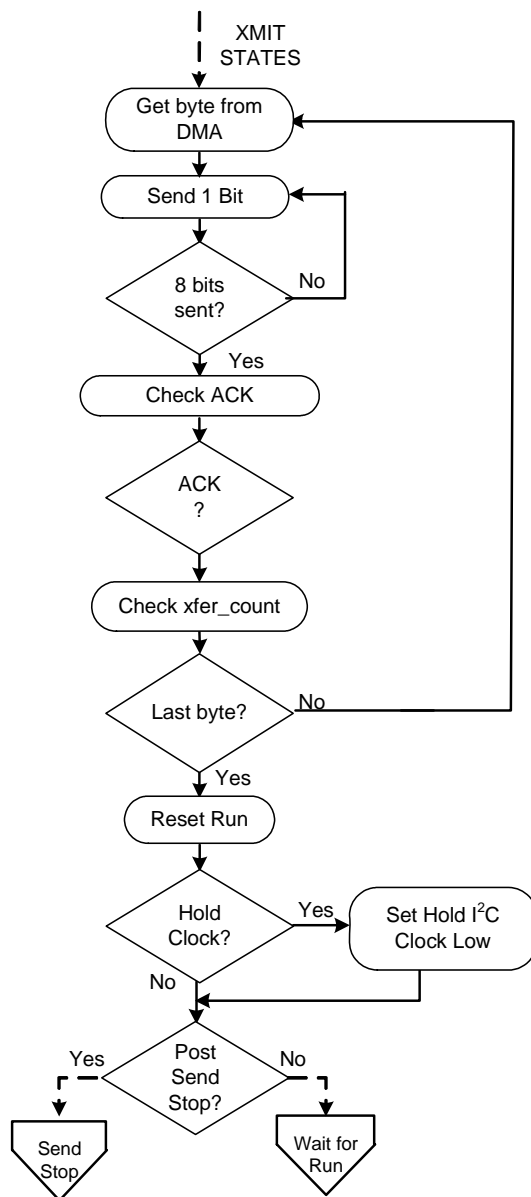
ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	NMAK	SP
----	-------	-----	-----	-----	-----	-----	----	-------	-----	------	------	----

Table 899. 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 103. I²C Master Mode Flow Chart—Initial States

**Figure 104. I²C Master Mode Flow Chart—Receive States**

Figure 105. I²C Master Mode Flow Chart—Transmit States

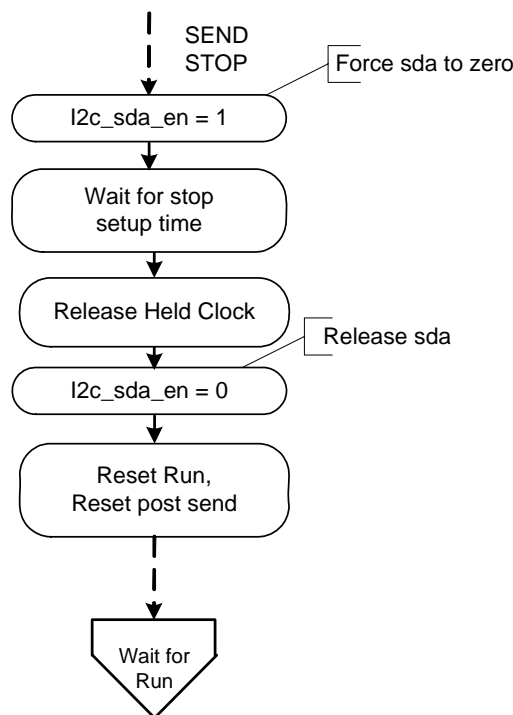


Figure 106. I²C Master Mode Flow Chart—Send Stop States

21.2.2.6. 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 107](#).

- 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 STMP3770 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 0.
- 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 STMP3770 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 0 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 STMP3770 slave DMA transfer engine reports this to software as an Early Termination interrupt request.

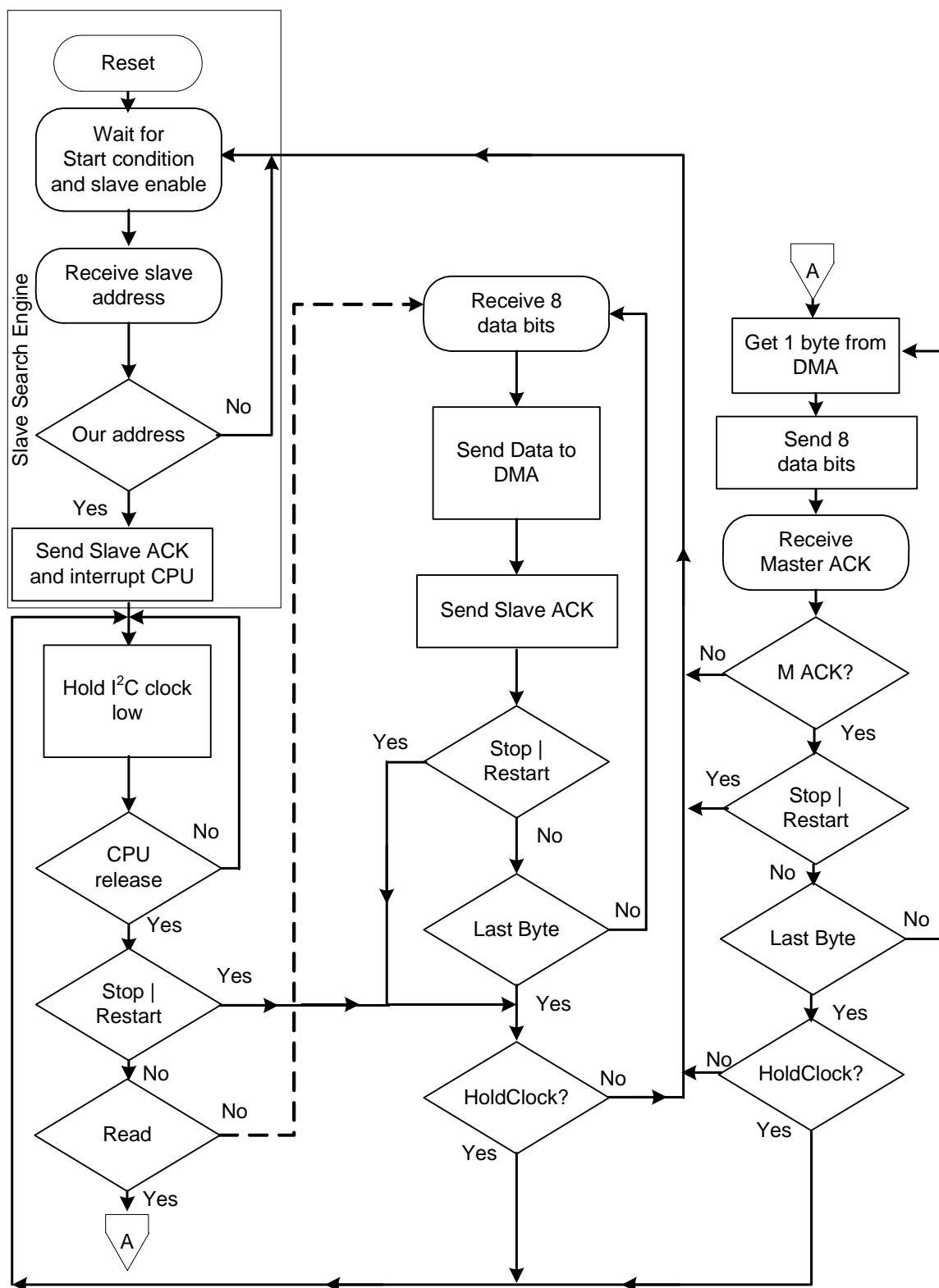


Figure 107. I²C Slave Mode Flow Chart

21.2.3. Programming Examples

21.2.3.1. Five Byte Master Write Using DMA

The example in Figure 108 shows sending five bytes from an STMP3770 operating as an I²C master to another device acting as an I²C slave.

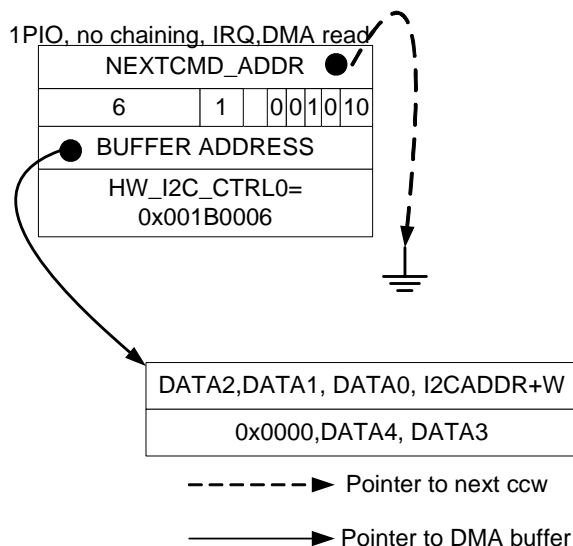


Figure 108. I²C Writing Five Bytes

The DMA command is initialized to send six bytes to the I²C controller and one word of PIO information to the HW_I2C_CTRL0 register.

Table 900. I²C 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 0 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
}

```

21.2.3.2. Reading 256 bytes from an EEPROM

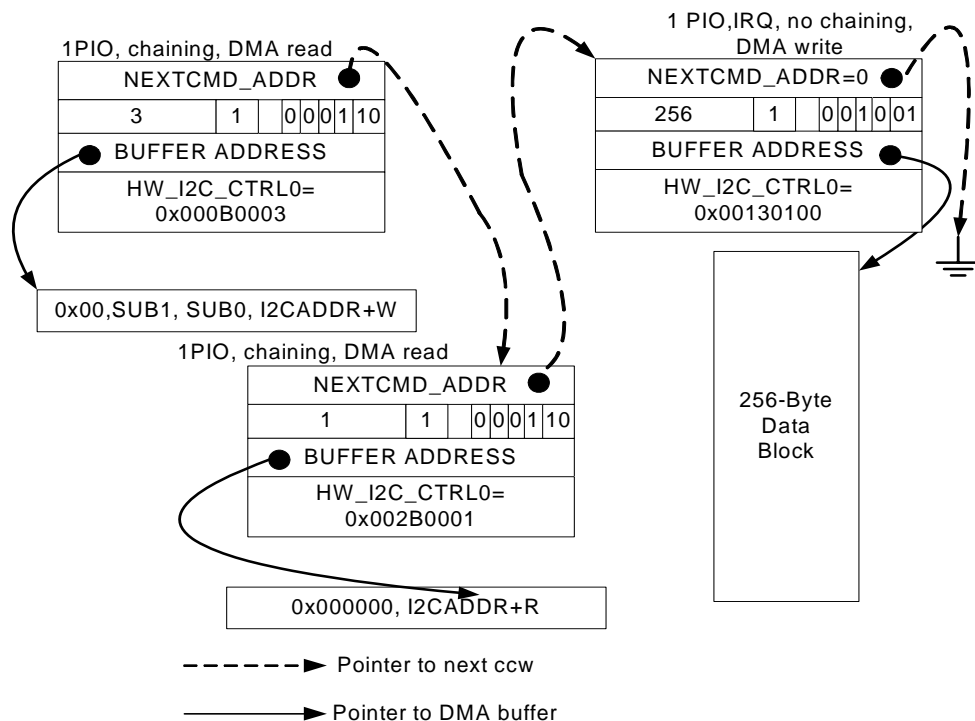


Figure 109. I²C Reading 256 Bytes from an EEPROM

```

//-----
// dma buffers to hold i2c command string for slave address+W plus sub0,
// sub 1 and the second command, a slave address+R
// eePROM write address == 0xA0, read address == 0xA1
//-----
unsigned char eeeprom_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) |
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)), // last command
    (reg32_t) &eeeprom_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) &eeeprom_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) &eeeprom_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 address 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_WrN(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 0 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
}
if (retries == SEMAPHORE_TIMEOUT) exit(2);
// the 256 bytes were read from the eePROM so return with no Error
return 0;
}

```

21.3. 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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

21.4. Programmable Registers

The following registers describe the programming interface for the slave and master I²C controller.

21.4.1. I²C Control Register 0 Description

The I²C Control Register 0 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 901. HW I2C CTRL0

SFSTRST	3 1
CLKGATE	3 0
RUN	2 9
RSVD	2 8
PRE_ACK	2 7
ACKNOWLEDGE	2 6
SEND_NAK_ON_LAST	2 5
PIO_MODE	2 4
MULTI_MASTER	2 3
CLOCK_HOLD	2 2
RETAIN_CLOCK	2 1
POST_SEND_STOP	2 0
PRE_SEND_START	1 9
SLAVE_ADDRESS_ENABLE	1 8
MASTER_MODE	1 7
DIRECTION	1 6
XFER_COUNT	1 5 1 4 1 3 1 2 1 1 1 0 0 9 0 8 0 7 0 6 0 5 0 4 0 3 0 2 0 1 0 0

Table 902. HW_I2C_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	Clear to 0 for normal operation. When this bit is set to 1 (default), then the entire block is held in its reset state. RUN = 0x0 Allow I ² C to operate normally. RESET = 0x1 Hold I ² C in reset.
30	CLKGATE	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block. RUN = 0x0 Allow I ² C to operate normally. NO_CLKS = 0x1 Do not clock I ² C gates in order to minimize power consumption.
29	RUN	RW	0x0	Set this bit to 1 to enable the I ² C 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 I ² C command in progress. RUN = 0x1 Process a slave or master I ² C command.
28	RSVD	RO	0x0	Reserved.
27	PRE_ACK	RW	0x0	Reserved for SigmaTel use.
26	ACKNOWLEDGE	RW	0x0	Set this bit to 1 to cause a pending acknowledge bit (prior to DMA transfer) to be acknowledged. Set it to 0 to NAK the pending acknowledge bit. This bit is set to 1 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 1 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 1 to enable PIO mode of operation for the I ² C 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 1 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 902. HW_I2C_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	CLOCK_HELD	RW	0x0	This bit is set to 1 by the I ² C controller state machines. It holds the I ² C 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 1. 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 1 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 1 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 1 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 1 to enable the slave address decoder. When an address match occurs, the I ² C 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 1 to select master mode. Set it 0 to select slave mode. SLAVE = 0x0 Operate in slave mode. MASTER = 0x1 Operate in master mode.
16	DIRECTION	RW	0x0	Set this bit to 1 to select an I ² C transmit operation in either slave or master mode. TRANSMIT = Write in master mode, read in slave mode. Clear to 0 to select an I ² C receive operation in either slave or master mode. RECEIVE = 0x0 I ² C receive operation for slave or master. TRANSMIT = 0x1 I ² C 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);
```

HW_I2C_TIMING0	0x80058010
HW_I2C_TIMING0_SET	0x80058014
HW_I2C_TIMING0_CLR	0x80058018
HW_I2C_TIMING0_TOG	0x8005801C

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD						HIGH_COUNT										RSVD					RCV_COUNT										

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x0	Reserved.
25:16	HIGH_COUNT	RW	0x78	Load this bit field with the APBX clock count for the high period of the I ² C clock.
15:10	RSVD	RO	0x0	Reserved.
9:0	RCV_COUNT	RW	0x30	Load this bit field with the APBX clock count for capturing read data after the I ² C clock goes high.

21.4.3. I²C Timing Register 1 Description

HW_I2C_TIMING1	0x80058020
HW_I2C_TIMING1_SET	0x80058024
HW_I2C_TIMING1_CLR	0x80058028
HW_I2C_TIMING1_TOG	0x8005802C

STMP3770

Table 908. HW_I2C_TIMING2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x0	Reserved.
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	RSVD	RO	0x0	Reserved.
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
```

21.4.5. 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 909. HW_I2C_CTRL1

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3
RSVD								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 910. HW_I2C_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	RSVD	RO	0x0	Reserved.
24	BCAST_SLAVE_EN	RW	0x0	Set this bit to 1 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 1 to match ALL 7 bit I ² C addresses.
15	BUS_FREE_IRQ_EN	RW	0x0	Set this bit to 1 to enable bus free interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. DISABLED = 0x0 No interrupt request pending. ENABLED = 0x1 Interrupt request pending.
14	DATA_ENGINE_CMPLT_IRQ_EN	RW	0x0	Set this bit to 1 to enable data engine complete interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. DISABLED = 0x0 No interrupt request pending. ENABLED = 0x1 Interrupt request pending.
13	NO_SLAVE_ACK_IRQ_EN	RW	0x0	Set this bit to 1 to enable interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. DISABLED = 0x0 No interrupt request pending. ENABLED = 0x1 Interrupt request pending.
12	OVERSIZE_XFER_TERM_IRQ_EN	RW	0x0	Set this bit to 1 to enable interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. DISABLED = 0x0 No interrupt request pending. ENABLED = 0x1 Interrupt request pending.
11	EARLY_TERM_IRQ_EN	RW	0x0	Set this bit to 1 to enable interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. DISABLED = 0x0 No interrupt request pending. ENABLED = 0x1 Interrupt request pending.
10	MASTER_LOSS_IRQ_EN	RW	0x0	Set this bit to 1 to enable interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. DISABLED = 0x0 No interrupt request pending. ENABLED = 0x1 Interrupt request pending.
9	SLAVE_STOP_IRQ_EN	RW	0x0	Set this bit to 1 to enable interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. DISABLED = 0x0 No interrupt request pending. ENABLED = 0x1 Interrupt request pending.

Table 910. HW_I2C_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
8	SLAVE_IRQ_EN	RW	0x0	Set this bit to 1 to enable interrupt requests to be routed to the interrupt collector. Clear to 0 to disable interrupts from the I ² C controller. The corresponding HW_I2C_CTRL1_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 I ² C controller because the bus has become free. This bit is cleared by software writing a 1 to its SCT clear address. This interrupt indicates that the I ² C 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 I ² C controller because the data engine transfer has completed. This bit is cleared by software writing a 1 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 I ² C 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 writing a 1 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 I ² C controller. This bit is cleared by software writing a 1 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 910. HW_I2C_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	EARLY_TERM_IRQ	RW	0x0	<p>This bit is set to indicate that an interrupt is requested by the I²C controller. This bit is cleared by software writing a 1 to its SCT clear address. This interrupt indicates that a master write transform from the STMP3770 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.</p> <p>NO_REQUEST = 0x0 No interrupt request pending. REQUEST = 0x1 Interrupt request pending.</p>
2	MASTER_LOSS_IRQ	RW	0x0	<p>This bit is set to indicate that an interrupt is requested by the I²C controller. This bit is cleared by software writing a 1 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 1 to the bus at the same time as another master writes a 0. The wired AND bus produces a 0 on the bus which is detected by the losing master.</p> <p>NO_REQUEST = 0x0 No interrupt request pending. REQUEST = 0x1 Interrupt request pending.</p>
1	SLAVE_STOP_IRQ	RW	0x0	<p>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.</p> <p>NO_REQUEST = 0x0 No interrupt request pending. REQUEST = 0x1 Interrupt request pending.</p>
0	SLAVE_IRQ	RW	0x0	<p>This bit is set to indicate that an interrupt is requested by the I²C controller. This bit is cleared by software writing a 1 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.</p> <p>NO_REQUEST = 0x0 No interrupt request pending. REQUEST = 0x1 Interrupt request pending.</p>

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
```

21.4.6. I²C Status Register Description

The I²C controller reports status information in the I²C Status Register.

HW_I2C_STAT 0x80058050

Table 911. HW I2C STAT

MASTER_PRESENT	31
SLAVE_PRESENT	30
ANY_ENABLED_IRQ	29
RSVD	28
	27
	26
	25
RCVD_SLAVE_ADDR	24
	23
	22
	21
RCVD_SLAVE_ADDR	20
	19
	18
	17
SLAVE_ADDR_EQ_ZERO	16
	15
	14
	13
SLAVE_SEARCHING	12
	11
	10
	9
CLK_GEN_BUSY	8
	7
	6
	5
SLAVE_BUSY	4
	3
	2
	1
DATA_ENGINE_DMA_WAIT	0
	31
	30
	29
DATA_ENGINE_BUSY	28
	27
	26
	25
BUS_FREE_IRQ_SUMMARY	24
	23
	22
	21
NO_SLAVE_ACK_IRQ_SUMMARY	20
	19
	18
	17
OVERSIZE_XFER_TERM_IRQ_SUMMARY	16
	15
	14
	13
EARLY_TERM_IRQ_SUMMARY	12
	11
	10
	9
MASTER_LOSS_IRQ_SUMMARY	8
	7
	6
	5
SLAVE_STOP_IRQ_SUMMARY	4
	3
	2
	1
SLAVE_IRQ_SUMMARY	0
	31
	30
	29

Table 912. HW I2C STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	MASTER_PRESENT	RO	0x1	This read-only bit indicates that the I ² C master function is present when it reads back a 1. This I ² C function is not available on a device that returns a 0 for this bit field. UNAVAILABLE = 0x0 I ² C is not present in this product. AVAILABLE = 0x1 I ² C is present in this product.
30	SLAVE_PRESENT	RO	0x1	This read-only bit indicates that the I ² C slave function is present when it reads back a 1. This I ² C function is not available on a device that returns a 0 for this bit field. UNAVAILABLE = 0x0 I ² C is not present in this product. AVAILABLE = 0x1 I ² C is present in this product.
29	ANY_ENABLED_IRQ	RO	0x0	This read-only bit indicates that the I ² C 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	RSVD	RO	0x0	Reserved.
23:16	RCVD_SLAVE_ADDR	RO	0x00	This read-only byte indicates that the state of the slave I ² C address byte received, including the read/write bit received from an address byte that matched our slave address.
15	SLAVE_ADDR_EQ_ZERO	RO	0x0	This read-only bit indicates that the I ² C slave function was searching for a transaction that matches the current slave address. When set to 1, it indicates that an address match was found for the exact address 0x00. ZERO_NOT_MATCHED = 0x0 I ² C slave search did not match a 0. WAS_ZERO = 0x1 I ² C has found an address match against address 0x00.

Table 912. HW_I2C_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
14	SLAVE_FOUND	RO	0x0	<p>This read-only bit indicates that the I²C slave function was searching for a transaction that matches the current slave address. When set to 1, 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.</p> <p>IDLE = 0x0 I²C slave search is idle. WAITING = 0x1 I²C has found an address match and is holding the I²C clock line low.</p>
13	SLAVE_SEARCHING	RO	0x0	<p>This read-only bit indicates that the I²C slave function is searching for a transaction that matches the current slave address.</p> <p>IDLE = 0x0 I²C slave search is idle. ACTIVE = 0x1 I²C is actively searching for an address match.</p>
12	DATA_ENGINE_DMA_WAIT	RO	0x0	<p>This read-only bit is set to 1 when the data engine is waiting 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.</p> <p>CONTINUE = 0x0 I²C master is not waiting on data from the DMA. WAITING = 0x1 I²C master is waiting on data from the DMA.</p>
11	BUS_BUSY	RO	0x0	<p>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.</p> <p>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.</p>
10	CLK_GEN_BUSY	RO	0x0	<p>This read-only bit indicates that the I²C clock generator is busy with a transaction.</p> <p>IDLE = 0x0 I²C clock generator is idle. BUSY = 0x1 I²C clock generator is busy performing a command.</p>
9	DATA_ENGINE_BUSY	RO	0x0	<p>This read-only bit indicates that the I²C data transfer engine is busy with a data transmit or receive operation. In addition, it can be busy, as a master, sending a start or stop condition.</p> <p>IDLE = 0x0 I²C data engine is idle. BUSY = 0x1 I²C data engine busy performing a data transfer.</p>
8	SLAVE_BUSY	RO	0x0	<p>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.</p> <p>IDLE = 0x0 I²C slave search engine is idle. BUSY = 0x1 I²C slave search engine is busy searching for an address match.</p>
7	BUS_FREE_IRQ_SUMMARY	RO	0x0	<p>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.</p> <p>NO_REQUEST = 0x0 No interrupt request pending. REQUEST = 0x1 Interrupt request pending.</p>

Table 912. HW_I2C_STAT Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	DATA_ENGINE_CMPLT_IRQ_SUMMARY	RO	0x0	This bit is set to indicate that an interrupt is requested by the I ² C 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 I ² C 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 I ² C 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 I ² C 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 I ² C 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.
1	SLAVE_STOP_IRQ_SUMMARY	RO	0x0	This bit is set to indicate that an interrupt is requested by the I ² C 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 I ² C 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_SLAVER_BUSY_IDLE_VAL) { // then wait till it finishes }
```

21.4.7. 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.

0x80058060

Table 913. HW_I2C_DATA

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
DATA																															

Table 914. 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. 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.

21.4.8. 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 915. HW_I2C_DEBUG0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
DMAREQ	DMAENDCMD	DMAKICK	TBD	DMA_STATE												START_TOGGLE	STOP_TOGGLE	GRAB_TOGGLE	CHANGE_TOGGLE	TESTMODE	SLAVE_HOLD_CLK	SLAVE_STATE									

Table 916. 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

STMP3770

Table 916. HW_I2C_DEBUG0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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 toggle
```

21.4.9. 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 917. HW_I2C_DEBUG1

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
I2C_CLK_IN	I2C_DATA_IN	RSVD	DMA_BYTE_ENABLES	RSVD	CLK_GEN_STATE	RSVD	RSVD	LST_MODE	LOCAL_SLAVE_TEST	RSVD	FORCE_CLK_ON	FORCE_CLK_IDLE	FORCE_ARB_LOSS	FORCE_RCV_ACK	FORCE_I2C_DATA_OE	FORCE_I2C_CLK_OE															

Table 918. 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 I ² C clock pad.
30	I2C_DATA_IN	RO	0x1	A copy of the pad input signal for the I ² C clock pad.
29:28	RSVD	RO	0x0	Reserved.
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 I ² C DMA state machine to track the number of bytes written by the DMA. Individual bits are cleared as they are consumed.
23	RSVD	RO	0x0	Reserved.
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 I ² C DMA state machine to track the number of bytes written by the DMA. Individual bits are cleared as they are consumed.
15:11	RSVD	RO	0x0	Reserved.
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., I ² C 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.
8	LOCAL_SLAVE_TEST	RW	0x0	Writing a 1 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	RSVD	RO	0x0	Reserved.
5	FORCE_CLK_ON	RW	0x0	Writing a 1 to this bit will force the clock generator to send a continuous stream of clocks on the I ² C bus.
4	FORCE_CLK_IDLE	RW	0x0	Writing a 1 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 1 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 1 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	Writing a 1 to this bit will force an output enable at the pad. The pad data line is tied to 0. Thus, the I ² C data line will either be high impedance or 0.
0	FORCE_I2C_CLK_OE	RW	0x0	Writing a 1 to this bit will force an output enable at the pad. The pad data line is tied to 0. Thus, the I ² C clock line will either be high impedance or 0.

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
```

0x80058090

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
MAJOR								MINOR								STEP															

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x01	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

5-37xx-DS2-1.04-031408

22. APPLICATION UART

This chapter describes the application UART included on the STMP3770, 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 3.25 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 transmit and receive internal clock from the 24-MHz UART internal reference clock input UARTCLK. XCLK is not tied to the UARTCLK in the STMP3770.

It offers similar functionality to the industry-standard 16C550 UART device and supports baud rates of up to 3.25 Mb/s (in high-speed configuration with a minimum XCLK frequency of 1.5 MHz). [Figure 110](#) 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 HW_UARTAPP_LINECTRL register controls both receive and transmit operations. However, when HW_UARTAPP_CTRL2_USE_LCR2 is set, then HW_UARTAPP_LINECTRL controls receive operations and HW_UARTAPP_LINECTRL2 controls transmit operations.

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 time-out), transmit, modem status, and error conditions.

Two DMA channels are supported, one for transmit and one for receive.

If a time-out condition occurs in the middle of a receive DMA block transfer, then the UART ends the DMA transfer and signals the end of the DMA block transfer. 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.

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.

Note: Names for the external pins used by the application UART begin with “UART2”. Pin names beginning with “UART1” are used by the Debug UART.

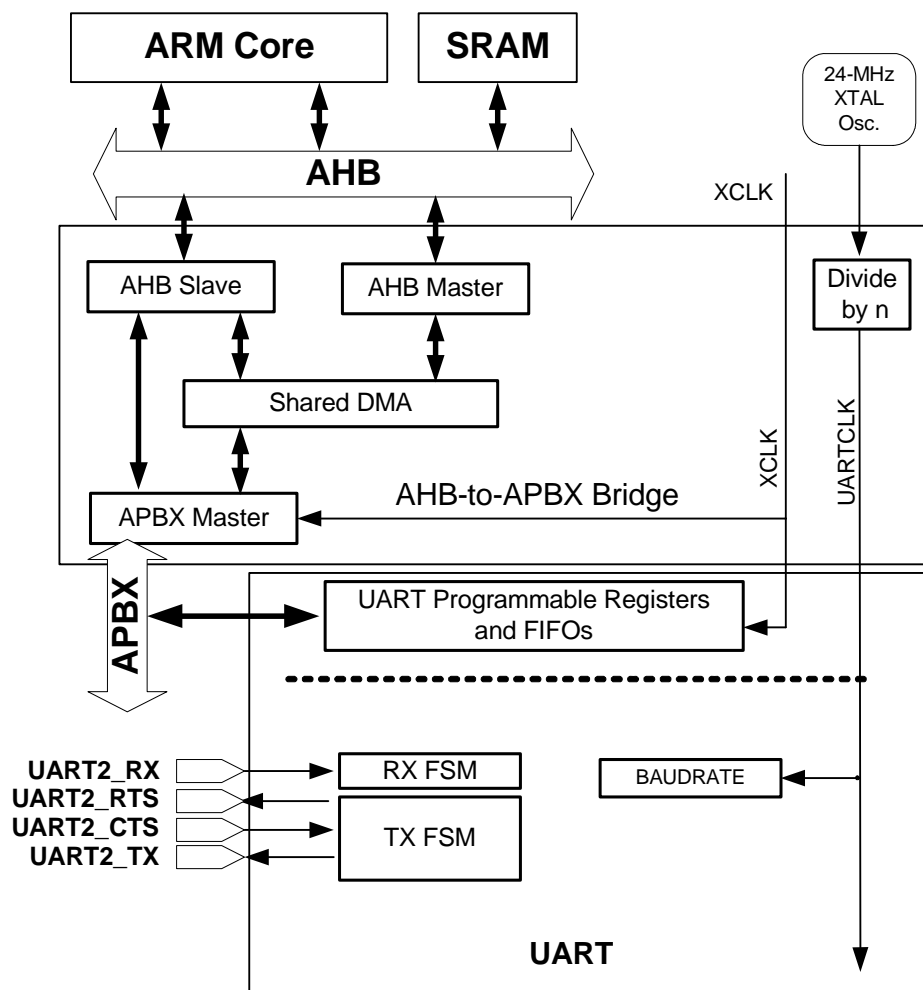


Figure 110. Application UART Block Diagram

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

If USE_LCR2 is set, the Application UART Line Control Register applies to the receive operation, and similar control data written to the Application UART Line Control 2 Register applies to the transmit operation.

22.2.1. Fractional Baud Rate Divider

The baud rate divisor is calculated from the frequency of UARTCLK and the desired baud rate by using the following formula:

$$\text{divisor} = (\text{UARTCLK} * 32) / \text{baud rate, rounded to the nearest integer}$$

The divisor must be between 0x000000EC 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 111 illustrates the UART character frame.

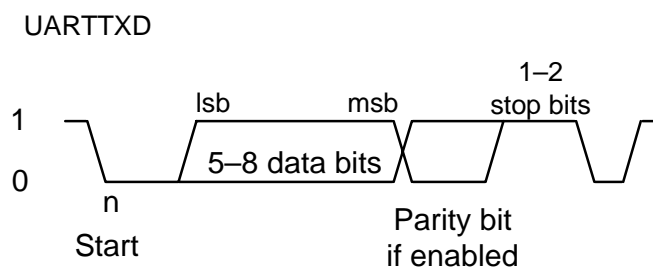


Figure 111. Application UART Character Frame

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 CTRL0. 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 time-out 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 0. 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 or UARTLCR2_H (if USE_LCR2 is set). 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 BaudClk, begins running and data is sampled on the first 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 first cycle of BaudClk, otherwise a false start bit is detected and it is ignored. If the start bit was valid, successive data bits are sampled on every second cycle of BaudClk (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 921](#)).

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 921](#) shows the bit functions of the receive FIFO.

Table 921. 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.

STMP3770

Table 927. 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	RSVD	RO	0x0	Reserved.
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. This bit is unsupported.
12	OUT1	RW	0x0	This bit is the complement of the UART Out1 (nUARTOut1) modem status output. This bit is unsupported.
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. This bit is unsupported.
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 the UART signals. When the UART is disabled in the middle of reception, it completes the current character before stopping.

Table 927. HW_UARTAPP_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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 the UART signals. When the UART is disabled in the middle of transmission, it completes the current character before stopping.
7	LBE	RW	0x0	Loopback Enable. This feature reduces the amount of external coupling required during system test. If this bit is set to 1, the UARTTXD path is fed through to the UARTRXD path. When this bit is set, the modem outputs are also fed through to the modem inputs.
6	USE_LCR2	RW	0x0	If this bit is set to 1, the Line Control 2 Register values are used.
5:3	RSVD	RO	0x0	Reserved.
2	SIRLP	RW	0x0	IrDA SIR Low Power Mode. This bit is unsupported.
1	SIREN	RW	0x0	SIR Enable. This bit is unsupported.
0	UARTEN	RW	0x0	UART Enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for the UART signals. 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 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.

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 928. 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	

STMP3770



Table 929. 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.
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.

22.4.5. UART Line Control 2 Register Description

The UART Line Control 2 Register contains integer and fractional part of the baud rate divisor value. It also contains the line control bits.

HW_UARTAPP_LINECTRL2 0x8006C040

HW_UARTAPP_LINECTRL2_SET 0x8006C044

HW_UARTAPP_LINECTRL2_CLR 0x8006C048

HW_UARTAPP_LINECTRL2_TOG 0x8006C04C

Table 930. HW_UARTAPP_LINECTRL2

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2
BAUD_DIVINT												RSVD		BAUD_DIVFRAC								SPS	WLEN		FEN	STP2	EPS	PEN	RSVD

Table 931. HW_UARTAPP_LINECTRL2 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.
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: 00 = 5 bits. 01 = 6 bits. 10 = 7 bits. 11 = 8 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.

Table 935. 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 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 simultaneously 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.

22.4.8. 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 0x8006C070

Table 939. HW_UARTAPP_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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.

22.4.10. UART Version Register Description

The UART version register can be used to read the version of the UARTAPP.

HW UARTAPP VERSION 0x8006C090

Table 940. HW UARTAPP VERSION

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
MAJOR								MINOR					STEP																		

Table 941. HW_UARTAPP_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x2	Fixed read-only value reflecting the MAJOR field of RTL version.
23:16	MINOR	RO	0x0	Fixed read-only value reflecting the MINOR field of RTL version.
15:0	STEP	RO	0x0	Fixed read-only value reflecting the stepping of RTL version.

UARTAPP Block v2.0

STMP3770



23. DEBUG UART

This chapter describes the debug UART included on the STMP3770, how to operate it, and how to disable the FIFOs. Programmable registers are described in [Section 23.3](#).

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 transmit and receive internal clock from the 24-MHz UART internal reference clock input UARTCLK. XCLK is not tied to the UARTCLK in the STMP3770.

It offers similar functionality to the industry-standard 16C550 UART device and supports baud rates of up to 115 Kb/s. [Figure 113](#) shows a block diagram of the debug UART. The debug UART operation and baud rate values are controlled by the line control register (HW_UARTDBGLCR_H, HW_UARTDBGIBRD, and HW_UARTDBGFBRD).

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 time-out), 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.

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.

Note: Names for the external pins used by the debug UART begin with “UART1”. Pin names beginning with “UART2” are used by the Application UART.

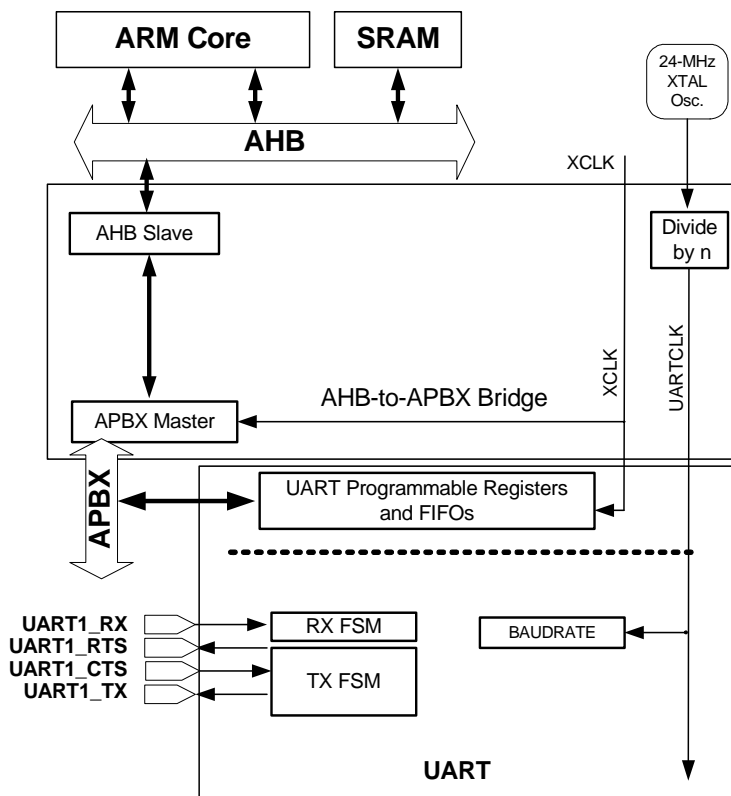


Figure 112. Debug UART Block Diagram

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 UARTCLK and the desired baud rate by using the following formula:

$$\text{divisor} = (\text{UARTCLK} * 4) / \text{baud rate, rounded to the nearest integer}$$

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 113 illustrates the UART character frame.

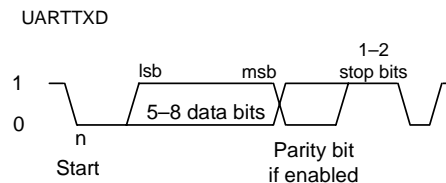


Figure 113. Debug UART Character Frame

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 942).

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.

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 942](#) shows the bit functions of the receive FIFO.

Table 942. Receive FIFO Bit Functions

FIFO BIT	FUNCTION
11	Overflow indicator
10	Break error
9	Parity error
8	Framing error
7:0	Received data

23.2.6. Disabling the FIFOs

FIFOs can be disabled. In this case, the transmit and receive sides of the 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.

23.3. Programmable Registers

This section describes the debug UART's programmable registers.

23.3.1. UART Data Register Description

HW UARTDBGDR

0x80070000

Table 943. HW UARTDBGDR

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0		
UNAVAILABLE																RESERVED				OE	BE	PE	FE	DATA									

Table 944. HW_UARTDBGDR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	UNAVAILABLE	RO	0x0	The UART only implements 16- and 8-bit registers, so the top 2 or 3 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 set to 1, this bit 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 set to 1, this bit 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:

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 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 RSR_ECR register.

Table 949. HW_UARTDBGIBRD

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
UNAVAILABLE																BAUD_DIVINT															

Table 950. HW_UARTDBGIBRD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	UNAVAILABLE	RO	0x0	The UART only implements 16- and 8-bit registers, so the top 2 or 3 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.

23.3.5. UART Fractional Baud Rate Divisor Register Description

The FBRD register is the fractional part of the baud rate divisor value.

HW_UARTDBGFBRD

0x80070028

Table 951. HW_UARTDBGFBRD

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
UNAVAILABLE																						RESERVED		BAUD_DIVFRAC							

Table 952. HW_UARTDBGFBRD Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:8	UNAVAILABLE	RO	0x0	The UART only implements 16- and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
7:6	RESERVED	RO	0x0	Reserved.
5:0	BAUD_DIVFRAC	RW	0x0	Baud Rate Fraction [5:0]. The fractional baud rate divisor.

23.3.6. UART Line Control Register, HIGH Byte Description

The LCR_H is the Line Control Register.

HW_UARTDBGLCR_H

0x8007002C

Table 953. HW_UARTDBGLCR_H

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
UNAVAILABLE																RESERVED								SPS	WLEN		FEN	STP2	EPS	PEN	BRK

Table 954. HW_UARTDBGLCR_H Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	UNAVAILABLE	RO	0x0	The UART only implements 16- and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15:8	RESERVED	RO	0x0	Reserved.
7	SPS	RW	0x0	Stick Parity Select. When bits 1, 2, and 7 of the LCR_H 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.

Table 956. HW_UARTDBGCR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7	LBE	RW	0x0	Loopback 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.
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.

23.3.8. 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 957. HW_UARTDBGIFLS

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
UNAVAILABLE																RESERVED										RXIFLSEL			TXIFLSEL		

Table 958. HW_UARTDBGIFLS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:16	UNAVAILABLE	RO	0x0	The UART only implements 16- and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15:6	RESERVED	RO	0x0	Reserved.
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:

Use the IFLS register to define the FIFO level at which the UARTRXINTR 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.

23.3.9. 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 959. HW_UARTDBGIMSC

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
UNAVAILABLE																RESERVED					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRMIM	DCDMIM	CTSMIM	RIMIM

24. AUDIOIN/ADC

This chapter describes the AUDIOIN/ADC module implemented on the STMP3770, including DMA, sample rate conversion, and internal operation. Programmable registers are described in [Section 24.4](#).

24.1. Overview

The STMP3770 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 114](#) shows a high-level block diagram of the AUDIOIN module. See [Figure 4 on page 39](#) for a diagram of the audio path and control options.

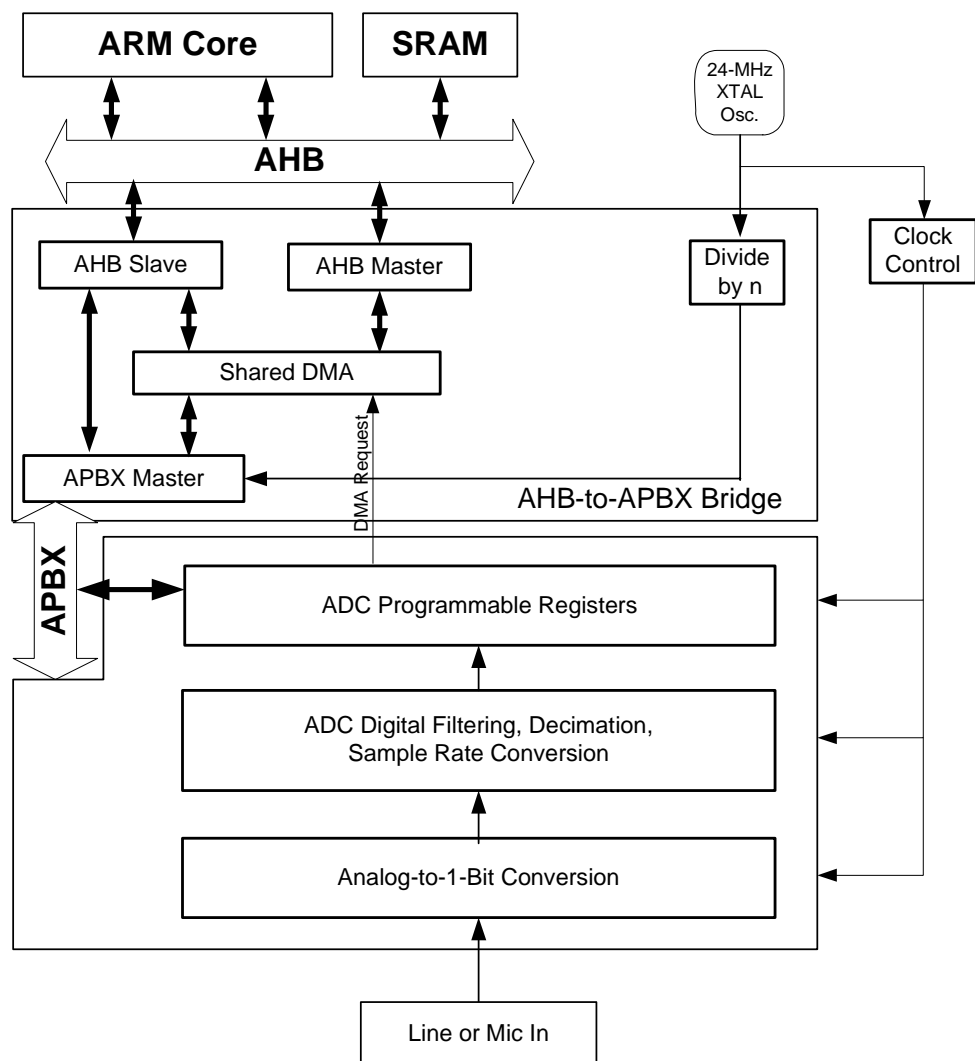


Figure 114. AUDIOIN/ADC Block Diagram

24.2. Operation

The first step in receiving audio to the AUDIOIN module requires the analog-to-digital converter (ADC). The STMP3770 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 STMP3770'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 (HW_AUDIOIN_DATA) while monitoring either the FIFO or DMA request status bits in the AUDIOIN debug register (HW_AUDIOIN_ADCDEBUG).

Also present on the STMP3770 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 (0), 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 1 in this space are affected. For example, writing a 1 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 1 to the register's clear address clear the bit, while the toggle address causes bits to invert their current state.

24.2.1. 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 1 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).

24.2.2. ADC Sample Rate Converter and Internal Operation

[Table 969](#) contains the required value of the HW_AUDIOIN_ADCSRR register for various common sample rates. To make small sample rate adjustments (for example to track F_s 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 969](#).

Table 969. Bit Field Values for Standard Sample Rates

SAMPLE RATE	HW_AUDIOIN_ADCSRR			
	BASEMULT	SRC_HOLD	SRC_INT	SRC_FRAC
$F_{\text{sample}_{\text{ADC}}}$				
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 969. Bit Field Values for Standard Sample Rates (Continued)

SAMPLE RATE	HW_AUDIOIN_ADCSRR			
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:

$$SRConv_{ADC} = 65536 * [(F_{analog}_{ADC}) / (8 * F_{sample}_{ADC})]$$

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_AUDIOIN_ANACKCTRL_ADCDIV register (see [Figure 115](#)). This divider generates sample strobes at F_{analog}_{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 115](#), 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:

$$24.576 \text{ MHz} = Q * F_{sample}_{ADC} \text{ where } Q \text{ comes from the set of integers}$$

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:

$$16.9344 \text{ MHz} = Q * F_{sample}_{ADC} \text{ where } Q \text{ comes from the set of integers}$$

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:

$$24.0 \text{ MHz} = P * F_{\text{sample_ADC}}, \text{ where } P \text{ is a rational number}$$

The A/D block includes a variable rate or rational decimator as shown in Figure 115 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 0, then a sample is produced.

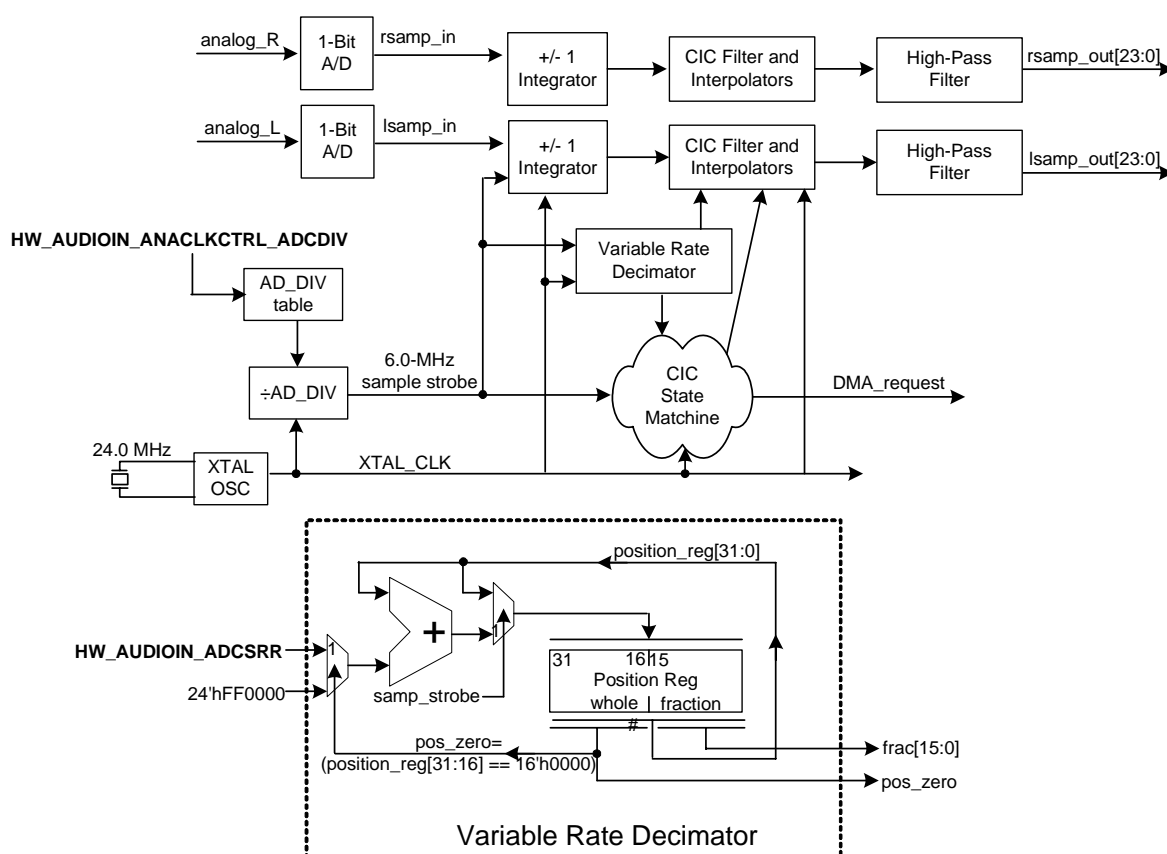


Figure 115. Variable-Rate A/D Converter

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 $F_{\text{ADC}} = 6.0 \text{ MHz}$ will be 36 times smaller than the values resulting from capturing the same source signal at $F_{\text{ADC}} = 1.0 \text{ MHz}$.

- The peak range of values seen in a capture of a signal at 44.1 kHz with $F_{\text{analog_ADC}} = 6.0 \text{ MHz}$ is ± 3200 decimal.
- The oversample ratio in this case is $\text{OSR} = 136.054$.
- Calculate a magnitude constant, K_{filter} for ADC's filter from this as $K_{\text{filter}} = \text{OSR}^2 / \text{Peak Value} = (136.054)^2 / 3200 = 5.7846$.
- For any OSR in any sample rate, the peak value can be approximated by $\text{Value}_{\text{peak}} = \text{OSR}^2 / K_{\text{filter}}$.

In signal processing, one frequently normalizes the range of values to ± 1.0 , as seen in a fixed-point scaled integer¹. 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 $\pm 2^{23}$ 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 STMP3770):

$$\text{ScaleFactor} = 2^{23} * K_{\text{filter}} / \text{OSR}^2$$

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.

24.2.3. 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 116. 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 117. 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.

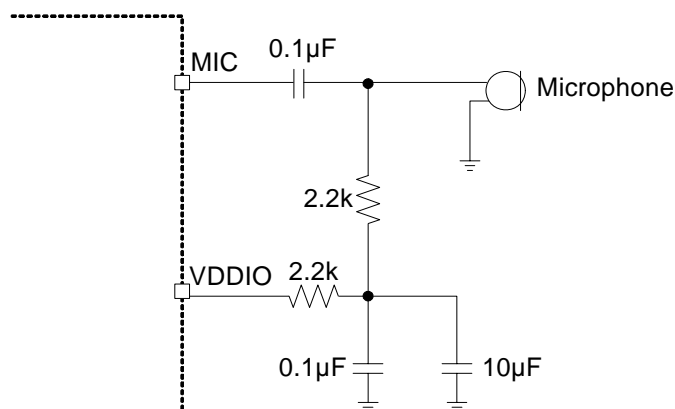


Figure 116. External Microphone Bias Generation

1. A normalized two's complement 24-bit number cannot actually express a value of +1.0 without overflowing.

STMP3770

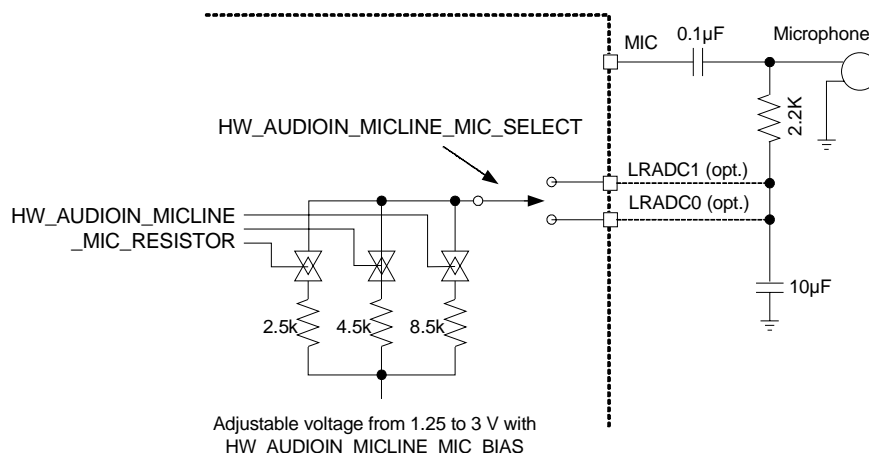


Figure 117. Internal Microphone Bias Generation

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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

24.4. Programmable Registers

The following registers provide control for programmable elements of the AUDIOIN/ACD block.

24.4.1. 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 970. 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	9	8	7	6	5	4	3	2	1	0	
SFTRST	CLKGATE	RSVD										DMAWAIT_COUNT					RSVD					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 971. HW_AUDIOIN_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	AUDIOIN Module Soft Reset. Setting this bit to 1 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 0 for normal operation. Note that the CLKGATE bit does not affect SFTRST, because it must remain writable 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 0 for normal AUDIOIN operation. Note that when this bit is set, it remains writable during clock gating so that it may be disabled by the user.
29:21	RSVD	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	RSVD	RO	0x0	Reserved.
10	LR_SWAP	RW	0x0	Left/Right Input Channel Swap Enable. Setting this bit to 1 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 971. 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 1 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 that ties the output filter chain back to the input filter chain. This bit should be cleared to 0 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 1 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 1 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.

Table 971. HW_AUDIOIN_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	FIFO_ERROR_IRQ_EN	RW	0x0	FIFO Error Interrupt Enable. Set this bit to 1 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 1 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 0 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
```

24.4.2. 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 972. HW AUDIOIN STAT

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	
ADC_PRESENT	RSVD																															

Table 973. 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 1 in products that include the AUDIOIN/ADC. If this bit is 0, the AUDIOIN/ADC is permanently disabled and cannot be operated by the user.
30:0	RSVD	RO	0x0	Reserved.

Table 975. HW_AUDIOIN_ADCSRR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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: $\text{output_sample_rate} = (6 \times 10^6 * \text{BASEMULT}) / (\text{SRC_INT.SRC_FRAC} * 8 * (\text{SRC_HOLD} + 1))$ 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	RSVD	RO	0x0	Reserved.
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: $\text{output_sample_rate} = (6 \times 10^6 * \text{BASEMULT}) / (\text{SRC_INT.SRC_FRAC} * 8 * (\text{SRC_HOLD} + 1))$ 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	RSVD	RO	0x0	Reserved.
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: $\text{output_sample_rate} = (6 \times 10^6 * \text{BASEMULT}) / (\text{SRC_INT.SRC_FRAC} * 8 * (\text{SRC_HOLD} + 1))$ 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 to output a sample rate of 48 kHz:
HW_AUDIOIN_ADCSRR.BASEMULT = 0x1; // quad-rate
HW_AUDIOIN_ADCSRR.SRC_HOLD = 0x0; // 0 for full- double- quad-rates
HW_AUDIOIN_ADCCSRR.SRC_INT = 0xF; // 15 for the integer portion
HW_AUDIOIN_ADCSRR.SRC_FRAC = 0x13FF; // the fractional portion
```

24.4.4. 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 976. HW_AUDIOIN_ADCVOLUME

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD		VOLUME_UPDATE_LEFT		RSVD		EN_ZCD		RSVD		VOLUME_LEFT							RSVD		VOLUME_UPDATE_RIGHT		RSVD		VOLUME_RIGHT								

Table 977. HW_AUDIOIN_ADCVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x00	Reserved.
28	VOLUME_UPDATE_LEFT	RO	0x0	Left Channel Volume Update Pending. This bit is set to 1 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	RSVD	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	RSVD	RO	0x0	Reserved.
23:16	VOLUME_LEFT	RW	0xFE	Left Channel Volume Setting. This bit field is used to establish the incoming audio signal strength during record. Volume ranges from -0.5 dB (0xFE - reset value) to -100 dB (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 -100 dB. Also note that a setting of 0xFF is reserved.
15:13	RSVD	RO	0x00	Reserved.

Table 977. HW_AUDIOIN_ADCVOLUME Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
12	VOLUME_UPDATE_RIGHT	RO	0x0	Right Channel Volume Update Pending. This bit is set to 1 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.
11:8	RSVD	RO	0x0	Reserved.
7:0	VOLUME_RIGHT	RW	0xFE	Right Channel Volume Setting. This bit field is used to establish the incoming audio signal strength during record. Volume ranges from -0.5 dB (0xFE - reset value) to -100 dB (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 -100 dB. Also note that a setting of 0xFF is reserved.

DESCRIPTION:

The AUDIOIN Volume Register allows independent volume control of the left and right channels. 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.
```

24.4.5. 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
```

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																															
ENABLE_ADCDMA																															
ADC_DMA_REQ_HAND_SHAKE_CLK_CROSS																															
SET_INTERRUPT3_HAND_SHAKE																															
DMA_PREQ																															
FIFO STATUS																															

Table 979. 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	RSVD	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 979. 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;
```

24.4.6. 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 980. HW_AUDIOIN_ADCVOL

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0																								
RSVD				VOLUME_UPDATE_PENDING				RSVD				EN_ADC_ZCD				MUTE				RSVD																SELECT_LEFT				GAIN_LEFT				RSVD				SELECT_RIGHT				GAIN_RIGHT			

STMP3770



Table 981. HW_AUDIOIN_ADCVOL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x0	Reserved.
28	VOLUME_UPDATE_PENDING	RO	0x0	Volume Update Pending. This bit is set to 1 by the hardware when either an ADC left or right volume update is pending, i.e., waiting on a zero-crossing. The bit is set following a write to either the GAIN_LEFT or GAIN_RIGHT bit fields and is cleared when both gain values are applied to the input coincident with zero-crossings in both the right and left channels. This status bit is not used when EN_ADC_ZCD=0.
27:26	RSVD	RO	0x0	Reserved.
25	EN_ADC_ZCD	RW	0x0	Enable Zero-Cross Detect for ADC Amplifier.
24	MUTE	RW	0x1	ADC Mute. When set, this bit mutes both the left and right channel analog inputs. 0 = Unmute. 1 = Mute.
23:14	RSVD	RO	0x000	Reserved.
13:12	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 = Reserved.
11:8	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.
7:6	RSVD	RO	0x0	Reserved.
5:4	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 = Reserved.
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.

Table 983. HW_AUDIOIN_MICLINE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
18:16	MIC_BIAS	RW	0x0	Microphone Bias Voltage Select. 0 = 1.21 V 1 = 1.46 V, Up to 7=2.96 V (in 0.25-V increments)
15:6	RSVD	RO	0x00	Reserved.
5:4	MIC_CHOPCLK	RW	0x0	Enable chopping in the microphone amplifier: 00 = Disabled. 01 = 192 kHz. 10 = 96 kHz. 11 = 48 kHz.
3:2	RSVD	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
```

24.4.8. Analog Clock Control Register Description

This register provides analog clock control.

```
HW_AUDIOIOIN_ANACKCTRL      0x8004C070
HW_AUDIOIOIN_ANACKCTRL_SET  0x8004C074
HW_AUDIOIOIN_ANACKCTRL_CLR  0x8004C078
HW_AUDIOIOIN_ANACKCTRL_TOG  0x8004C07C
```

Table 984. HW_AUDIOIN_ANACLKCTRL

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
CLKGATE	RSVD																								DITHER_OFF	SLOW_DITHER	INVERT_ADCCCLK	RSVD	ADCDIV		

Table 985. HW_AUDIOIN_ANACKCTRL 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	RSVD	RO	0x0	Reserved.

Table 987. 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 register in CPU diagnostic (non-DMA) mode
```

AUDIOIN Block v1.1

25. AUDIOOUT/DAC

This chapter describes the AUDIOOUT/DAC module implemented on the STMP3770, including DMA, sample rate conversion, internal operation, reference control settings, and headphone amplifier operation. Programmable registers are described in [Section 25.4](#).

25.1. Overview

The STMP3770 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 two possible outputs:

- Stereo Headphone Amplifier 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 118](#) shows a high-level diagram of the AUDIOOUT module. See [Figure 4](#) on [page 39](#) for a diagram of the audio path and control options.

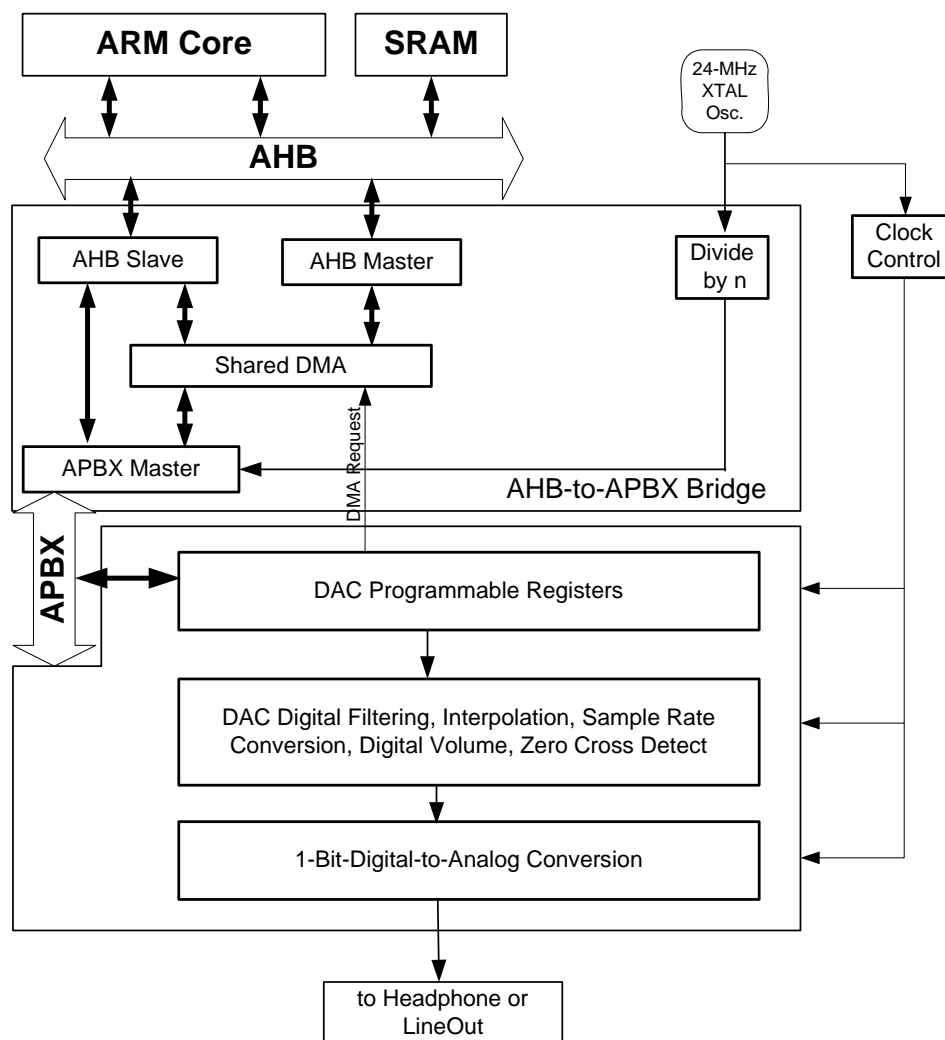


Figure 118. AUDIOOUT/DAC Block Diagram

25.2. Operation

Audio data conversion begins by either using the STMP3770'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 (HW_AUDIOOUT_DATA) while monitoring either the FIFO or DMA request status bits in the AUDIOOUT debug register (HW_AUDIOOUT_DACDEBUG).

Also present on the STMP3770 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 (0), 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 1 in this space are affected. For example, writing a 1 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 1 to the register's clear address clear the bit, while the toggle address causes bits to invert their current state.

25.2.1. **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 1 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.

25.2.2. DAC Sample Rate Converter and Internal Operation

Table 988 contains the required value of the HW_AUDIOOUT_DACSRR register for various common sample rates. Although these are the standard rates, any sample rate from 8K to 192 kHz can be programmed.

Table 988. Bit Field Values for Standard Sample Rates

SAMPLE RATE	HW_AUDIOOUT_DACSRR			
	BASEMULT	SRC_HOLD	SRC_INT	SRC_FRAC
Fsample _{DAC}				
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 988. 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

$$\text{SRConv}_{\text{DAC}} = 65536 * [(\text{Fanalog}_{\text{DAC}}) / (8 * \text{Hold}_{\text{DAC}} * \text{Fsample}_{\text{DAC}})]$$

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_AUDIOOUT_ANACKCTRL_DACDIV register (see [Figure 119](#)). This divider generates sample strobes at Fanalog_{DAC} where the divisors available come from the set {4,6,8,12,16,24}. With HW_AUDIOOUT_ANACKCTRL_DACDIV cleared to 0, 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 119](#), 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.

25.2.3. Reference Control Settings

The AUDIOOUT Reference Control Register provides control over the voltage and power for the audio analog circuits. It 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.

Use [Table 989](#) below to determine the appropriate setting for `HW_AUDIOOUT_REFCTRL_RAISE_REF` and `HW_AUDIOOUT_REFCTRL_VAG_VAL`, depending on the value in `HW_POWER_VDDACTRL_TRG`. Generally these settings put VAG at a value slightly below $VDDA/2$, which maximizes unclipped output swing.

Table 989. Audio Reference Control Settings

HW_POWER_VDDACTRL_TRG	VDDA	HW_AUDIOOUT_REFCTRL_RAISE_REF	HW_AUDIOOUT_REFCTRL_VAG_VAL	VAG LEVEL	MAXIMUM POWER OUTPUT	
					16-OHM HEADPHONES	32-OHM HEADPHONES
0x1F–0x15	2.100–2.025 V	1	0xF	1.000 V	25.31 mW	12.66 mW
0x14–0x13	2.000–1.975 V	1	0xE	0.975 V	23.93 mW	11.96 mW
0x12–0x11	1.950–1.925 V	1	0xD	0.950 V	22.58 mW	11.29 mW
0x10–0x0F	1.900–1.875 V	1	0xC	0.925 V	21.27 mW	10.63 mW
0x0E–0x0D	1.850–1.825 V	1	0xB	0.900 V	20.00 mW	10.00 mW
0x0C–0x0B	1.800–1.775 V	1	0xA	0.875 V	18.77 mW	9.38 mW
0x0A–0x09	1.750–1.725 V	1	0x9	0.850 V	17.58 mW	8.79 mW
0x08	1.700 V	1	0x8	0.825 V	16.43 mW	8.21 mW
0x07–0x06	1.675–1.650 V	0	0xC	0.809 V	15.71 mW	7.85 mW
0x05–0x04	1.625–1.600 V	0	0xB	0.788 V	14.79 mW	7.40 mW
0x03–0x02	1.575–1.550 V	0	0xA	0.766 V	13.86 mW	6.93 mW
0x01–0x00	1.525–1.500 V	0	0x9	0.744 V	12.96 mW	6.48 mW

25.2.4. Headphone

The STMP3770 supports a conventional stereo headphone drive, as shown in Figure 120.

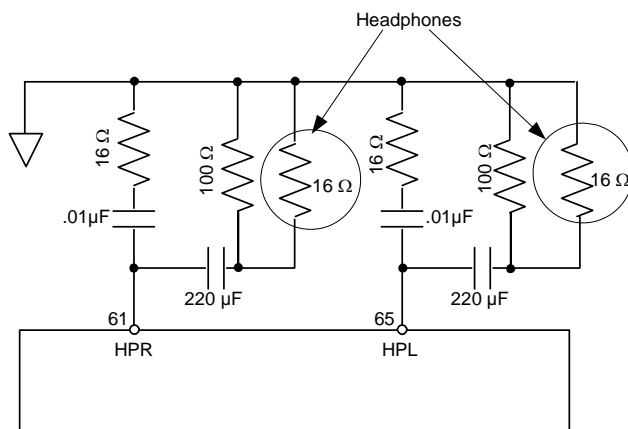


Figure 120. Conventional Stereo Headphone Application Circuit

In addition, as shown in Figure 121–Figure 123, 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.

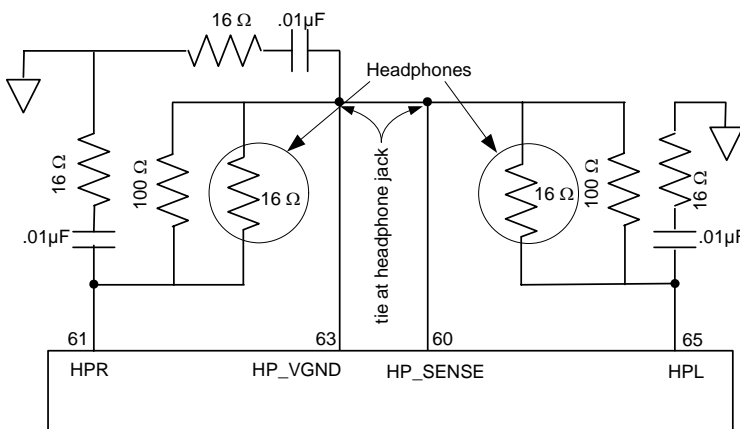


Figure 121. Stereo Headphone Application Circuit with Common Node

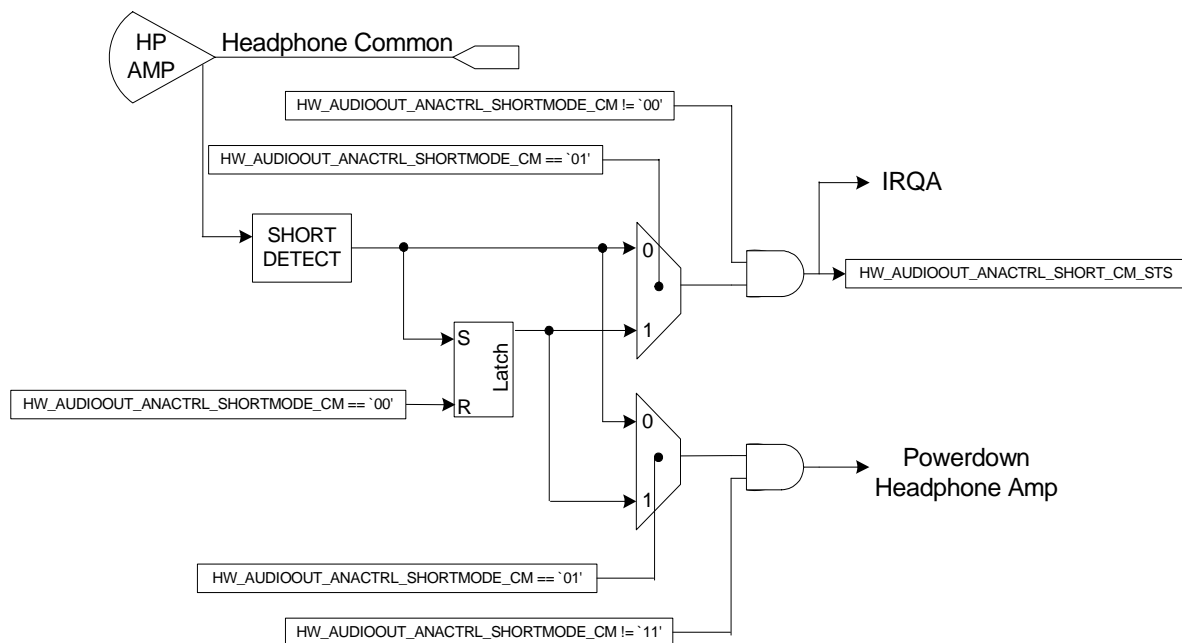


Figure 122. Stereo Headphone Common Short Detection and Powerdown Circuit

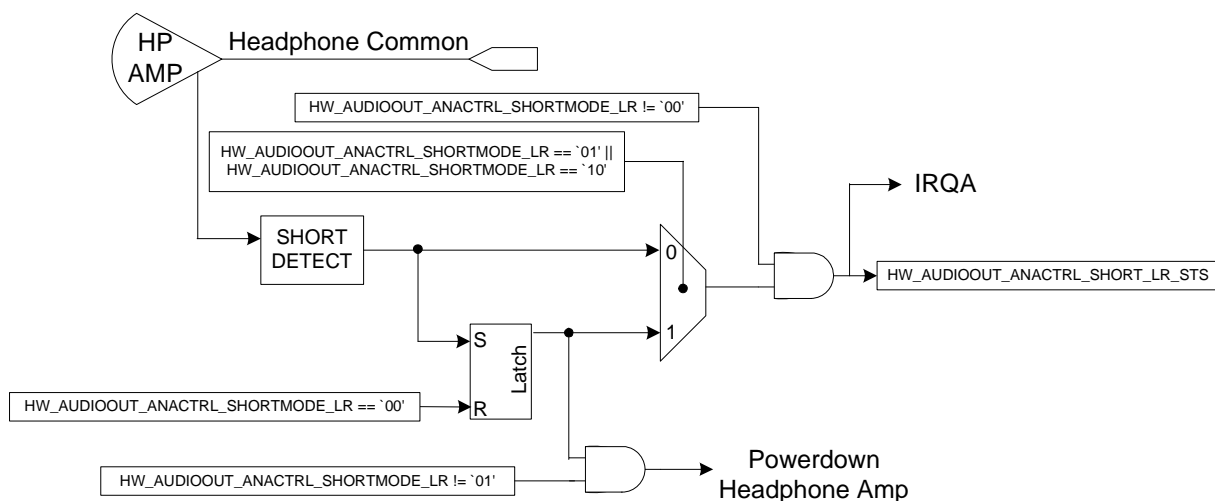


Figure 123. Stereo Headphone L/R Short Detection and Powerdown Circuit

25.2.4.1. 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 set of speakers plugged into the headphone jack 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 power-down 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.

25.2.4.2. 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:

$$\text{Power} = 1\text{mW} + V_{\text{peak}} * V_{\text{DDA}} / (16\text{ohm} * \pi) \quad (\text{per channel})$$

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:

$$\text{Power} = 1\text{mW} + V_{\text{peak}} * V_{\text{DDA}} * 2.8\text{mW} \quad (\text{per channel})$$

Again, this number is doubled in capless mode.

25.3. 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 34.4.10, "Correct Way to Soft Reset a Block" on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

25.4. Programmable Registers

The following registers provide control for programmable elements of the AUDIOOUT/DAC block.

25.4.1. 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 990. HW AUDIOOUT CTRL

SFTRST	3 1
CLKGATE	3 0
RSVD	2 9
	2 8
	2 7
	2 6
	2 5
	2 4
DMAWAIT_COUNT	2 3
	2 2
	2 1
	2 0
	1 9
	1 8
RSVD	1 7
	1 6
	1 5
	1 4
	1 3
	1 2
SS3D_EFFECT	1 1
	1 0
	0 9
	0 8
	0 7
	0 6
LOOPBACK	0 5
	0 4
	0 3
	0 2
	0 1
	0 0
RUN	

Table 991. HW_AUDIOOUT_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	AUDIOOUT Module Soft Reset. Setting this bit to 1 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 0 for normal operation. Note that the CLKGATE bit does not affect SFTRST since it must remain writable 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 0 for normal AUDIOOUT operation. Note that when this bit is set, it remains writable during clock gating so that it may be disabled by the user.
29:21	RSVD	RO	0x0	Reserved.

STMP3770



Table 991. 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	RSVD	RO	0x0	Reserved.
14	LR_SWAP	RW	0x0	Left/Right Output Channel Swap Enable. Setting this bit to 1 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	RSVD	RO	0x0	Reserved.
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	RSVD	RO	0x0	Reserved.
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 otherwise 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 991. HW_AUDIOOUT_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
4	LOOPBACK	RW	0x0	AUDIOIN-to-AUDIOOUT Loopback Enable. Setting this bit to 1 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 interface. This bit should be cleared to 0 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 1 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 1 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 1 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 1 causes AUDIOOUT operation to start, beginning with a DMA request to fill its 8-word FIFO. Clearing this bit to 0 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
```

25.4.2. 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

Table 992. HW AUDIOOUT STAT

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
DAC_PRESENT	RSVD																														

Table 993. 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 1 in products that include the AUDIOOUT/DAC. If this bit is 0, the AUDIOOUT/DAC is permanently disabled and cannot be operated by the user.
30:0	RSVD	RO	0x0	Reserved.

The AUDIOOUT Status Register provides an indication of the presence of the DAC functionality.

```
unsigned statusValue = HW_AUDIOOUT_STAT.DAC_PRESENT;
```

25.4.3. 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 994. HW AUDIOOUT DACSRR

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
OSR	BASEMULT			RSVD	SRC_HOLD			RSVD			SRC_INT			RSVD			SRC_FRAC														

Table 995. HW_AUDIOOUT_DACSRR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	OSR	RO	0x0	AUDIOOUT Oversample Rate. Note that for the STMP3770, 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	RSVD	RO	0x0	Reserved.
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: $\text{output_sample_rate} = (6 \times 10^6 * \text{BASEMULT}) / (\text{SRC_INT.SRC_FRAC} * 8 * (\text{SRC_HOLD} + 1))$. 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	RSVD	RO	0x0	Reserved.
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: $\text{output_sample_rate} = (6 \times 10^6 * \text{BASEMULT}) / (\text{SRC_INT.SRC_FRAC} * 8 * (\text{SRC_HOLD} + 1))$. 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	RSVD	RO	0x0	Reserved.
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: $\text{output_sample_rate} = (6 \times 10^6 * \text{BASEMULT}) / (\text{SRC_INT.SRC_FRAC} * 8 * (\text{SRC_HOLD} + 1))$. 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

```
// 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
```

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
```

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD		VOLUME_UPDATE_LEFT		RSVD		EN_ZCD		MUTE_LEFT		VOLUME_LEFT							RSVD		VOLUME_UPDATE_RIGHT		RSVD		MUTE_RIGHT		VOLUME_RIGHT						

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x00	Reserved.
28	VOLUME_UPDATE_LEFT	RO	0x0	Left Channel Volume Update Pending. This bit is set to 1 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	RSVD	RO	0x00	Reserved.

Table 997. 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	0xfe	Left Channel Volume Setting. Establishes the outgoing audio signal strength during playback. Volume ranges from -0.5 dB (0xFE) to -100 dB (0x37). Each increment of this bit field causes a half-dB increase in volume. The values 0x00-0x37 all produce the same attenuation level of -100 dB. Note: Do not change the DAC volume into or from the 0xFF state, or else it will create a large pop. Remaining in the 0xFF state is acceptable if all of the volume control is done in the headphone.
15:13	RSVD	RO	0x00	Reserved.
12	VOLUME_UPDATE_RIGHT	RO	0x0	Right Channel Volume Update Pending. This bit is set to 1 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	RSVD	RO	0x00	Reserved.
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	0xfe	Right Channel Volume Setting. Establishes the outgoing audio signal strength during playback. Volume ranges from -0.5 dB (0xFE) to -100 dB (0x37). Each increment of this bit field causes a half-dB increase in volume. The values 0x00-0x37 all produce the same attenuation level of -100 dB. Note: Do not change the DAC volume into or from the 0xFF state, or else it will create a large pop. Remaining in the 0xFF state is acceptable if all of the volume control is done in the headphone.

DESCRIPTION:

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.
```

25.4.5. 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 998. HW AUDIOOUT DACDEBUG

ENABLE_DACDMA	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0								
	RSVD																																							
	RAM_SS																																							
	RSVD																																							
	SET_INTERRUPT1_CLK_CROSS																																							
	SET_INTERRUPT0_CLK_CROSS																																							
	SET_INTERRUPT1_HAND_SHAKE																																							
	SET_INTERRUPT0_HAND_SHAKE																																							
	DMA_PREQ																																							
	FIFO_STATUS																																							

Table 999. 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 AUDIOOUT'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 AUDIOOUT logic without causing the audio data to pass through the AUDIOIN's FIR filter stages.
30:12	RSVD	RO	0x00	Reserved.
11:8	RAM_SS	RW	0x0	DIGFILT RAM Speed Select. Sense AMP Timing Control for DIGFILT 208x24 RAM. Do not alter unless instructed by SigmaTel.

Table 999. HW_AUDIOOUT_DACDEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:6	RSVD	RO	0x0	Reserved.
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_HANDSHAKE	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_HANDSHAKE	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;
```

25.4.6. 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

This register provides volume, mute, and input select controls for the headphone.

EXAMPLE:

```
HW_AUDIOOUT_HPVOL.MUTE = 0;
```

25.4.7. Reserved Register Description

This register is reserved.

```
HW_AUDIOOUT_RESERVED    0x80048060
```

HW_AUDIOOUT_RESERVED_SET 0x80048064

HW_AUDIOOUT_RESERVED_CLR 0x80048068

HW_AUDIOOUT_RESERVED_TOG 0x8004806C

Table 1002. HW_AUDIOOUT_RESERVED

3	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0				
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
RSVD																															

Table 1003. HW_AUDIOOUT_RESERVED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	RSVD	RO	0x00000000	Reserved.

DESCRIPTION:

This register is reserved.

EXAMPLE:

EMPTY

25.4.8. 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 1004. HW_AUDIOOUT_PWRDN

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD							LINEOUT	RSVD			SELFBIAS	RSVD		RIGHT_ADC	RSVD			DAC	RSVD			ADC	RSVD			CAPLESS	RSVD			HEADPHONE	

STMP3770

Table 1005. HW_AUDIOOUT_PWRDN Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	RSVD	RO	0x00	Reserved.
24	LINEOUT	RW	0x1	LineOut Power-Down. It is reset by a power-on reset only.
23:21	RSVD	RO	0x00	Reserved.
20	SELFBIAS	RW	0x0	This bit currently does not control any logic, altering it's value has no effect.
19:17	RSVD	RO	0x00	Reserved.
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	RSVD	RO	0x00	Reserved.
12	DAC	RW	0x1	Power Down DAC Analog Circuitry. It is reset by a power-on reset only.
11:9	RSVD	RO	0x00	Reserved.
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	RSVD	RO	0x00	Reserved.
4	CAPLESS	RW	0x1	Power Down Headphone Common Amplifier Used in Capless Headphone. It is reset by a power-on reset only.
3:1	RSVD	RO	0x00	Reserved.
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;
```

25.4.9. 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 1007. 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. These bits should only be used for test/debug, and not in an application.
15	RSVD	RO	0x0	Reserved.
14	VDDXTAL_TO_VDDD	RW	0x0	Shorts the supply of the XTAL oscillator to VDDD. This bit may be used to reduce power consumption, but should only be used on the advice of SigmaTel. This bit can only be written when both AUDIOOUT and AUDIOIN are not in reset or clock-gated. 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.
13	ADJ_ADC	RW	0x0	ADC Reference Voltage Adjust. When cleared, analog ADC reference is 1.5 V (RAISE_REF=1) or 1.31 V (RAISE_REF=0). 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): MSB is NOT used for voltage control. 0x7=1.60 V. 0x0=1.425 V, 25-mV steps, also affected by RAISE_REF. If RAISE_REF=0, 0x7=1.4 V, 0x0=1.25 V. If the MSB is high there is additional reference filtering for better 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.

Table 1007. HW_AUDIOOUT_REFCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:4	VAG_VAL	RW	0x0	VAG Reference Value (when ADJVAG set): MSB is NOT used for voltage control. 0x7=1.0 V, 0x0=0.825 V, 25-mV steps, also affected by RAISE_REF. If RAISE_REF=0, 0x7=0.875, 0x0=0.722 V. If the MSB is high the DAC reference buffer is bypassed which can improve SNR performance. See section on selecting the VAG level earlier in this chapter. See also the Reference Control Settings table earlier in this chapter for recommended settings. 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	RSVD	RO	0x0	Reserved.
2:0	DAC_ADJ	RW	0x0	These bits have no effect

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;
```

25.4.10. Miscellaneous Audio Controls Register Description

This register provides miscellaneous audio control bits.

```
HW_AUDIOOUT_ANACTRL      0x80048090
HW_AUDIOOUT_ANACTRL_SET  0x80048094
HW_AUDIOOUT_ANACTRL_CLR  0x80048098
HW_AUDIOOUT_ANACTRL_TOG  0x8004809C
```

Table 1008. HW_AUDIOOUT_ANACTRL

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4
RSVD	SHORT_CM_STS	RSVD	SHORT_LR_STS	RSVD	SHORTMODE_CM	RSVD	SHORTMODE_LR	RSVD	SHORT_LVLADJL	RSVD	SHORT_LVLADJR	RSVD	HP_HOLD_GND	HP_CLASSAB	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

STMP3770



Table 1009. HW_AUDIOOUT_ANACTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x00	Reserved. 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.
27:25	RSVD	RO	0x0	Reserved.
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.
23:22	RSVD	RO	0x0	Reserved.
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	RSVD	RO	0x00	Reserved.
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	RSVD	RO	0x0	Reserved.
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	RSVD	RO	0x0	Reserved.
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	RSVD	RO	0x0	Reserved.
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.

STMP3770



Table 1011. HW_AUDIOOUT_TEST Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25	TM_LINEOUT	RW	0x0	Testmode to connect lineout left to the microphone input to the ADC, and lineout right to the ADC. This is used for analog loopback DAC-Lineout-Mix-ADC Mode. There should be no load on the microphone 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%. SigmaTel recommends setting this bit to 0x1 to avoid headphone instability problems that happen with some loads or volume levels. This bit is reset by a power-on reset only.
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.
19:14	RSVD	RO	0x0	Reserved.
13	VAG_CLASSA	RW	0x0	Set to 1 to disable ClassAB mode in VAG Amp. Will increase current by ~200 μ A. 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 1 to double ClassA current in VAG amplifier (+240 μ A). 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.
11:3	RSVD	RO	0x00	Reserved.
2	DAC_CLASSA	RW	0x0	Set to 1 to disable ClassAB mode in DAC. Will increase power by ~600 μ A.
1	DAC_DOUBLE_I	RW	0x0	Set to 1 to double ClassA current in DAC amplifier (+360 μ A in each DAC).
0	DAC_DIS_RTZ	RW	0x0	Set to 1 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.
```


Table 1015. HW_AUDIOOUT_BISTSTAT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:0	DATA	RO	0x0	Failing data at the failing address.

DESCRIPTION:

The AUDIOOUT_BISTSTAT0 provides visibility into memory failures detected by the BIST engine.

EXAMPLE:

```
HW_AUDIOUT_BISTSTAT0.U = 0x00000000;
```

25.4.14. 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 1016. 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	9	8	7	6	5	4	3	2	1	0
RSVD			STATE					RSVD																ADDR							

Table 1017. HW_AUDIOOUT_BISTSTAT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x0	Reserved.
28:24	STATE	RO	0x0	Fail state of the BIST engine.
23:8	RSVD	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;
```

25.4.15. Analog Clock Control Register Description

This register provides analog clock control.

```
HW_AUDIOOUT_ANACKCTRL 0x800480e0
HW_AUDIOOUT_ANACKCTRL_SET 0x800480e4
HW_AUDIOOUT_ANACKCTRL_CLR 0x800480e8
HW_AUDIOOUT_ANACKCTRL_TOG 0x800480ec
```


Table 1023. HW_AUDIOOUT_LINEOUTCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x00	Reserved.
28	VOLUME_UPDATE_PENDING	RO	0x0	Volume Update Pending. This bit is set to 1 by the hardware when either a lineout left or right volume update is pending, i.e., waiting on a zero-crossing. The bit is set following a write to either the VOLUME_LEFT or VOLUME_RIGHT bit fields and is cleared when both attenuation values are applied to the output coincident with zero-crossings in both the right and left channels. This status bit is not used when EN_LINEOUT_ZCD=0.
27:26	RSVD	RO	0x0	Reserved.
25	EN_LINEOUT_ZCD	RW	0x0	Enable Zero-Cross Detect for Lineout Amplifier. Setting this bit will hold off volume changes to the analog until a zero-cross has occurred or until a time out occurs. This enables both left and right channels.
24	MUTE	RW	0x1	This bit mutes the lineout outputs, LOR and LOL. The lineout antipop startup/shutdown sequence should be followed before toggling this bit.
23:20	VAG_CTRL	RW	0x4	This bit field controls the analog ground level that LOL and LOR are centered on. It should be set to the value of VDDIO/2. The four bits of control allow programming from 1.725V(0000) to 1.35V(1111) in 25mV steps. The LINEOUT antipop startup/shutdown sequence should be followed before changing this field. Otherwise a pop will be generated at the outputs.
19:16	OUT_CURRENT	RW	0x0	This bit field controls the current in the class A output stage of the lineout. For higher impedance loads and/or lower lineout levels, good performance may be achieved with a lower level of current, thus saving power. There are only 5 current settings: 00000(min), 0001, 0011(10K load setting), 0111, and 1111(max)
15:13	CHARGE_CAP	RW	0x2	This field governs the charging of the capacitor which sets the startup behavior. Setting bit 13 alone causes the capacitor on LOVAG to charge, bit 14 alone causes the capacitor to discharge, bit 15 with bit 13 or bit 14 speeds up the time constant 10x. The time constant is nominally 100MB internally 0.1uF externally. The power cycle sequence from a cold start is to 1) make sure the capacitors are discharged using bit 14, 2) Set the min gain, nominal vag levels, and zerocross desires 3) Power up the lineout, 4) Ramp the vag capacitor, 5) Unmute the lineout 6) Ramp volume. Reverse the procedure when powering down.

26. SPDIF TRANSMITTER

This chapter describes the SPDIF transmitter provided on the STMP3770. It includes sections on interrupts, clocking, DMA operation, and PIO debug mode. Programmable registers are described in [Section 26.3](#).

26.1. Overview

The Sony-Philips Digital Interface Format (SPDIF) transmitter module transmits data according to the SPDIF digital audio interface standard (IEC-60958). [Figure 124](#) 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_FRAMECTRL 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 biphasemark 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.

26.2. Operation

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 HW_SPDIF_SRR_BASEMULT. 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 a 120-MHz clock, which is derived from a divide-by-4 of the PLL 480-MHz clock. 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.

Programming the HW_SPDIF_SRR register automatically generates the right frequency of pcm_spdif_clk from the divider in the clock controller block. There are no separate registers to control these dividers. Make sure that the CLKGATE bit in HW_CLKCTRL_SPDIF is cleared before starting the transmission.

The SPDIF module receives data by one of two methods:

- Software-directed PIO writes to the HW_SPDIF_DATA register
- Appropriate programming of the DMA-engine. (See [Chapter 12, “AHB-to-APBX Bridge with DMA” on page 335](#), 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.

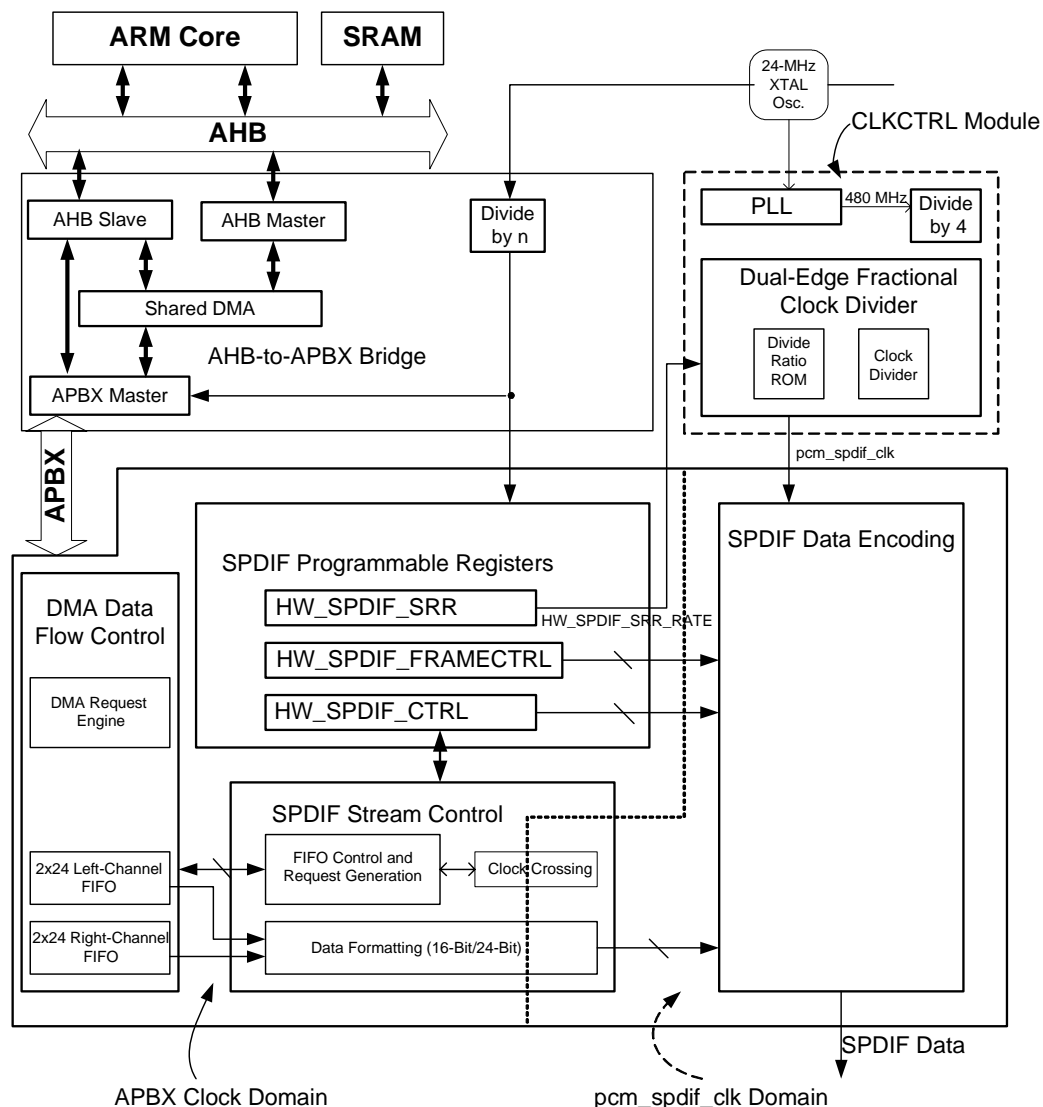


Figure 124. SPDIF Transmitter Block Diagram

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 sample for four frames before muting (sending zeros) the data stream based on the configuration of **HW_SPDIF_FRAMECTRL_AUTO_MUTE**. 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.

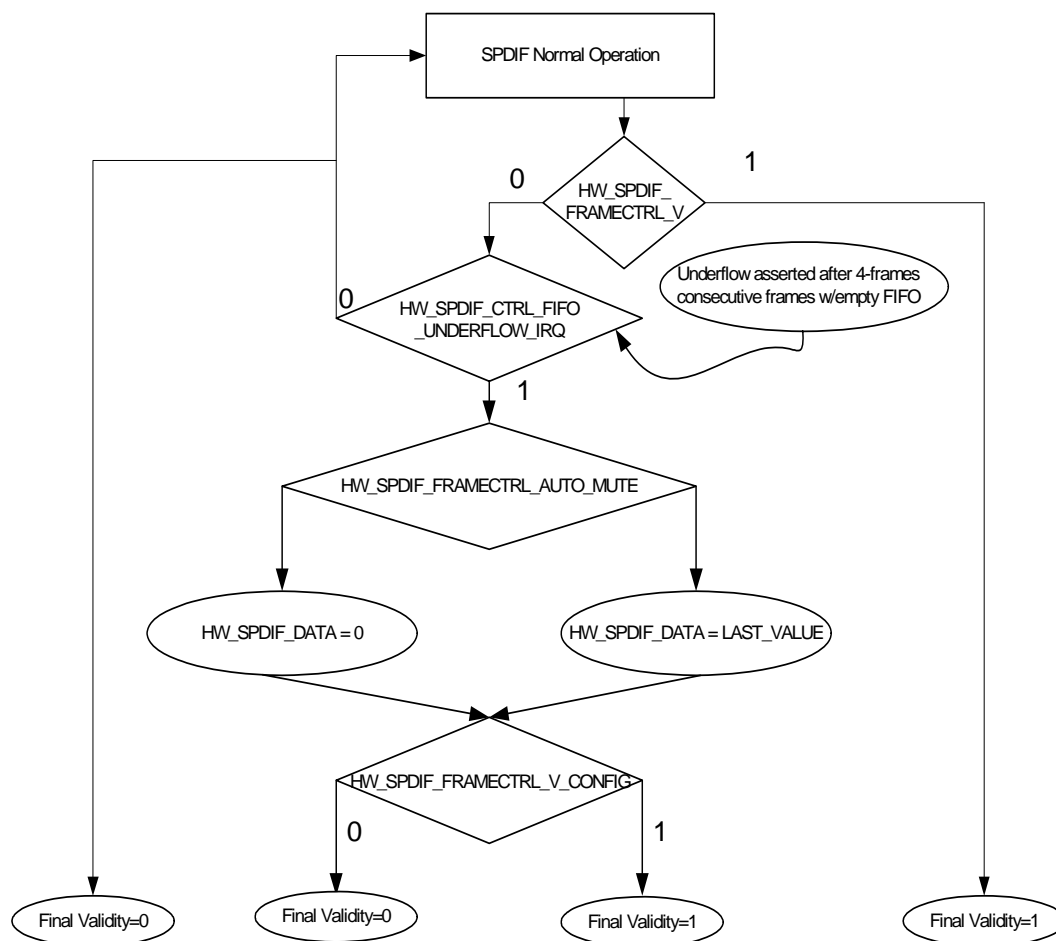


Figure 125. SPDIF Flow Chart

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 `WORD_LENGTH` bit in the `HW_SPDIF_CTR` 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 `WORD_LENGTH = 0`, 32-bit mode is enabled, and `HW_SPDIF_DATA` 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_DATA` register will be transmitted.
- Alternately, when `WORD_LENGTH = 1`, 16-bit mode is enabled, and the `HW_SPDIF_DATA` 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.

26.2.1. 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_ERROR_IRQ_EN`. On interrupt detection, the `HW_SPDIF_CTRL_FIFO_UNDERFLOW_IRQ` and `HW_SPDIF_CTRL_FIFO_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.

26.2.2. 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 STMP3770 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:

<code>F(48 kHz)</code>	≥ 6.144 MHz
<code>F(44.1 kHz)</code>	≥ 5.6448 MHz
<code>F(32 kHz)</code>	≥ 4.096 MHz
<code>F(96 kHz)</code>	≥ 12.288 MHz
<code>F(88.2 kHz)</code>	≥ 11.2896 MHz
<code>F(64 kHz)</code>	≥ 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_SPDIF_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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the `SFTRST` and `CLKGATE` bit fields.

26.2.3. 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 12, “AHB-to-APBX Bridge with DMA” on page 335](#) 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 126 describes a typical set of descriptors required to transmit two data-blocks.

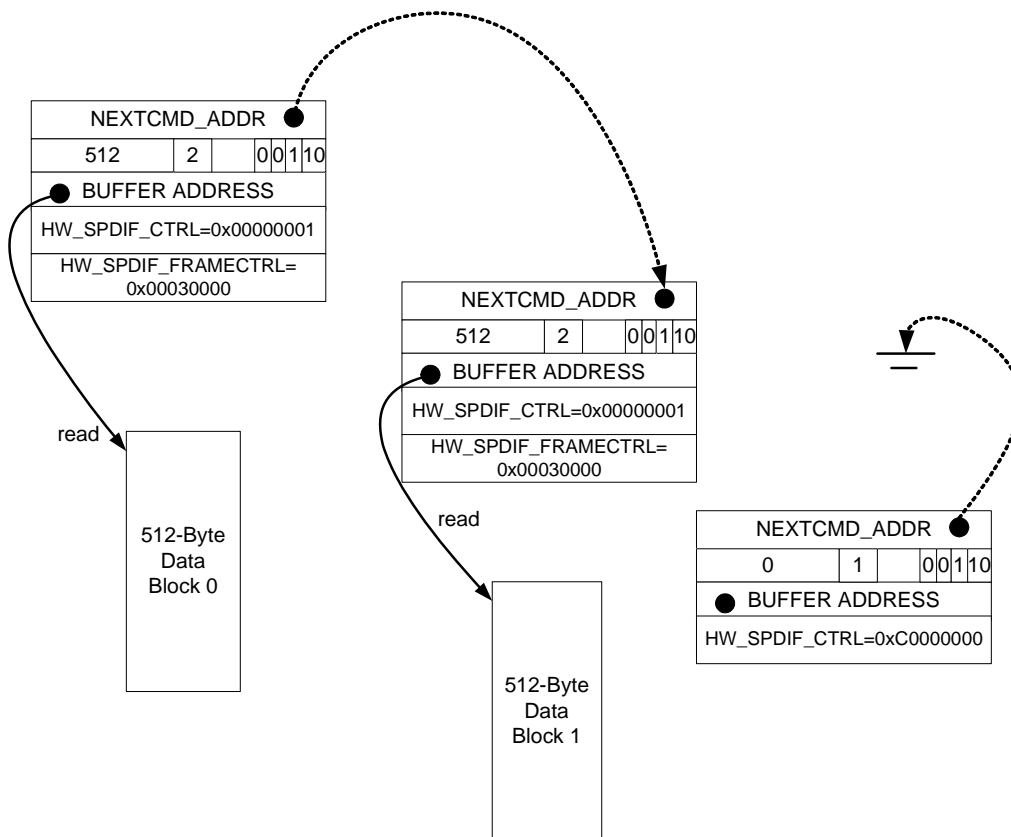


Figure 126. SPDIF DMA Two-Block Transmit Example

Here, the DMA is instructed to perform two PIO writes prior to toggling the DMA_PCMDKICK signal:

- HW_SPDIF_CTRL_FIFO_UNDERFLOW_IRQ_EN is set to enable interrupts on FIFO underflow detect
- HW_SPDIF_FRAMECTRL_AUTO_MUTE 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_COUNT 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_DMAWAIT_COUNT 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.

26.2.4. *PIO Debug Mode*

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 1 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. Clear CLKGATE in the HW_CLKCTRL_SPDIF register.
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 HW_SPDIF_CTRL_RUN.
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.

26.3. Programmable Registers

The following registers provide control for programmable elements of the SPDIF module.

26.3.1. 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 1026. HW_SPDIF_CTRL

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
SFTRST	CLKGATE	RSVD										DMAWAIT_COUNT				RSVD										WAIT_END_XFER	WORD_LENGTH	FIFO_UNDERFLOW_IRQ	FIFO_OVERFLOW_IRQ	FIFO_ERROR_IRQ_EN	RUN

STMP3770

Table 1027. HW_SPDIF_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	Setting this bit to 1 forces a reset to the entire block and then gates the clocks off. This bit must be cleared to 0 for normal operation.
30	CLKGATE	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block. WARNING: First set the CLKGATE bit in the HW_CLKCTRL_SPDIF 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_SPDIF register to 0.
29:21	RSVD	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	RSVD	RO	0x000	Reserved.
5	WAIT_END_XFER	RW	0x1	Set this bit to 1 if the SPDIF Transmitter should wait until the internal FIFO is empty before halting transmission based on deassertion of RUN. Use in conjunction with HW_SPDIF_STAT_END_XFER to determine transfer completion
4	WORD_LENGTH	RW	0x0	Set this bit to 1 to enable 16-bit mode. Clear this bit to 0 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 MSBs of HW_SPDIF_DATA will be transmitted.
3	FIFO_UNDERFLOW_IRQ	RW	0x0	This bit is set by hardware if the FIFO underflows during SPDIF transmission. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
2	FIFO_OVERFLOW_IRQ	RW	0x0	This bit is set by hardware if the FIFO overflows during SPDIF transmission. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
1	FIFO_ERROR_IRQ_EN	RW	0x0	Set this bit to 1 to enable a SPDIF interrupt request on FIFO overflow or underflow status conditions.
0	RUN	RW	0x0	Setting this bit to 1 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.

Table 1036. HW_SPDIF_DATA

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
HIGH																LOW															

Table 1037. 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 chan-  
nel in 16-bit mode  
HW_SPDIF_DATA = 0x12345678; // write 0x12345678 to either the left or right channel in 32-bit  
per sample mode.
```

26.3.7. SPDIF Version Register Description

The SPDIF Version Register is read-only and is used for debug to determine the implementation version number for the block.

```
HW SPDIF VERSION          0x80054060
```

Table 1038. HW SPDIF VERSION

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
MAJOR								MINOR				STEP																			

Table 1039. HW SPDIF VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x01	Fixed read-only value reflecting the MAJOR field of the RTL version.

STMP3770**Table 1039. HW_SPDIF_VERSION Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
23:16	MINOR	RO	0x01	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

SPDIF Block v1.1

27. DIGITAL RADIO INTERFACE (DRI)

This chapter describes the digital radio interface included on the STMP3770. Programmable registers are described in [Section 27.4](#).

27.1. Overview

The STMP3770 implements a digital interface (DRI) to SigmaTel's digital radio receiver product, the STFM1000. Details of the receiver are described the *STFM1000 Product Data Sheet*, available from SigmaTel.

The digital radio interface consists of two digital input signals that share pins with the analog line inputs. These pins can be used for either their LINE1R and LINE1L analog functions or their DRI_CLK and DRI_DATA digital functions in any given application, but not both.

[Figure 127](#) shows a block diagram of the digital radio interface, and [Figure 128](#) describes DRI synchronization and data recovery.

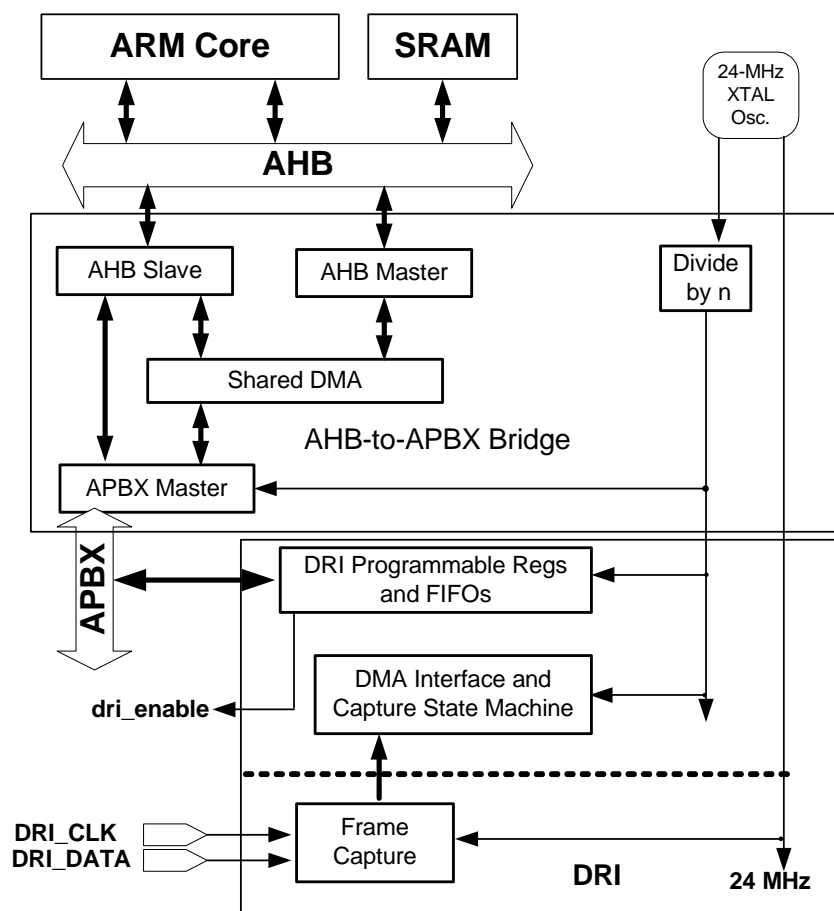


Figure 127. Digital Radio Interface (DRI) Block Diagram

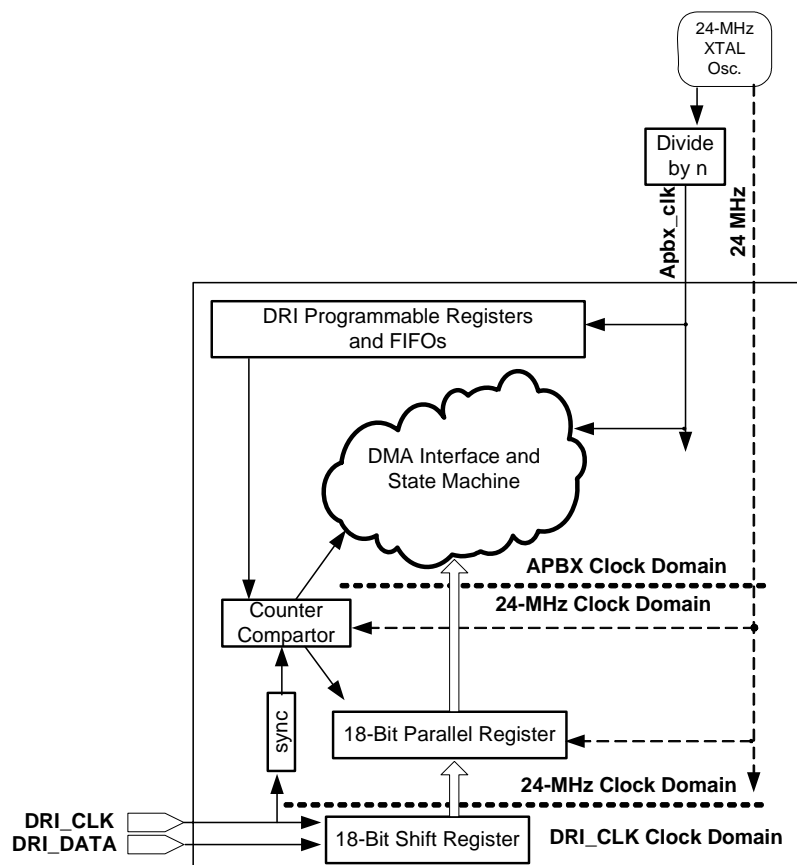


Figure 128. DRI Synchronization and Data Recovery

27.2. Operation

The frame structure used on the digital radio interface is shown in Figure 129. 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.

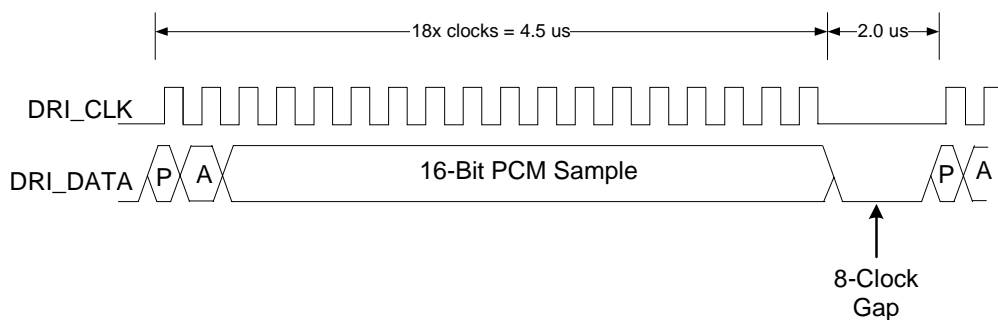


Figure 129. Digital Radio Interface (DRI) Framing

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 1040](#) and described in [Table 1041](#).

Table 1040. 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 1041. DRI Frame Bit Field Descriptions

BITS	LABEL	DEFINITION
17	PILOT PEAK	Set to 1 at the pilot peak. Clear to 0 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 1 by receiver to request attention from software. Software uses the I ² C bus to determine the cause of the attention request. When set to 1 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_PILOT_PHASE 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 ENABLE_INPUTS bit is set in HW_DRI_CTRL, the LINE1R input becomes the DRI_CLK digital input and the LINE1L input becomes the DRI_DATA digital input, as shown in [Figure 130](#).

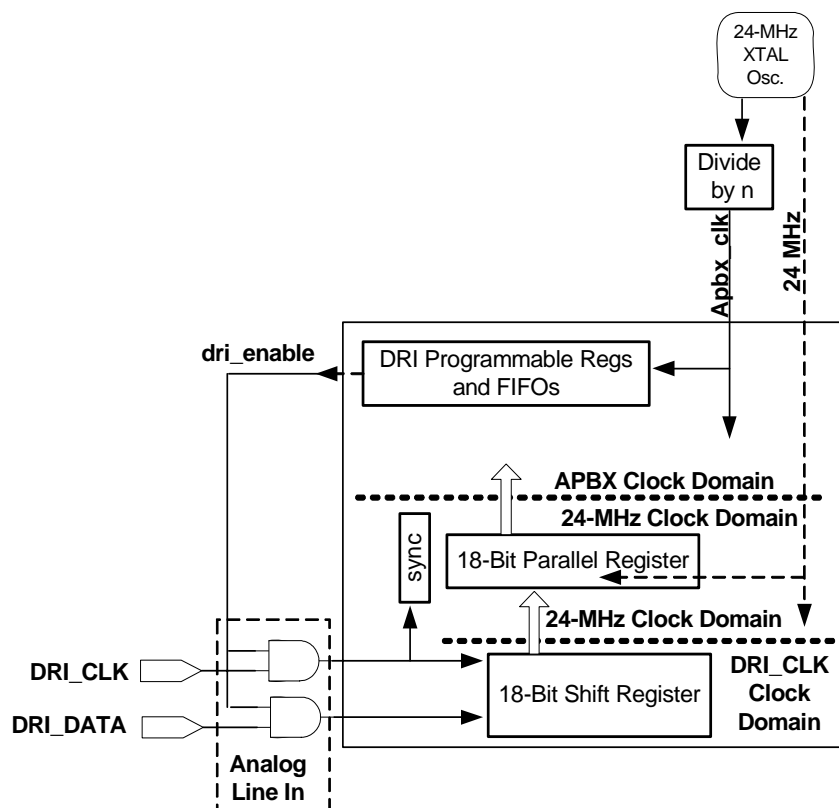


Figure 130. Digital Radio Interface (DRI) Digital Signals into Analog Line In

27.3. 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 34.4.10, "Correct Way to Soft Reset a Block" on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

27.4. Programmable Registers

The following registers describe the programming interface for the digital radio interface (DRI).

27.4.1. 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

STMP3770

Table 1043. HW_DRI_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15	REACQUIRE_PHASE	RW	0x0	Set this bit to 1 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 0 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	RSVD	RO	0x0	Reserved.
11	OVERFLOW_IRQ_EN	RW	0x0	Set this bit to enable an overflow interrupt. DISABLED = 0x0 Interrupt Request Disabled. ENABLED = 0x1 Interrupt Request Enabled.
10	PILOT_SYNC_LOSS_IRQ_EN	RW	0x0	Set this bit to enable a pilot sync loss interrupt. DISABLED = 0x0 Interrupt Request Disabled. ENABLED = 0x1 Interrupt Request Enabled.
9	ATTENTION_IRQ_EN	RW	0x0	Set this bit to enable an attention interrupt from the DRI. DISABLED = 0x0 Interrupt Request Disabled. ENABLED = 0x1 Interrupt Request Enabled.
8:4	RSVD	RO	0x0	Reserved.
3	OVERFLOW_IRQ	RW	0x0	Set this bit to indicate that an interrupt is requested by the DRI controller. This bit is cleared by software by writing a 1 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	Set this bit to indicate that an interrupt is requested by the DRI controller. This bit is cleared by software by writing a 1 to its SCT clear address. Set this bit 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.
1	ATTENTION_IRQ	RW	0x0	Set this bit to indicate that an interrupt is requested by the DRI controller. This bit is cleared by software by writing a 1 to its SCT clear address. Set this bit 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 1 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 digital radio interface. HALT = 0x0 No DRI command in progress. RUN = 0x1 Process a slave or master DRI command.

Table 1050. HW DRI DEBUG0

DMAREQ	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
DMACMDKICK																																
DRI_CLK_INPUT																																
DRI_DATA_INPUT																																
TEST_MODE																																
PILOT_REP_RATE																																
SPARE																																
FRAME																																

Table 1051. 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 1 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	Set to 1 to 1 to select a 12-sample repeat cycle for pilot sync (6-MHz case) in the test mode. Clear to 0 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 toggle
```

27.4.6. 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 1052. HW_DRI_DEBUG1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0								
INVERT_PILOT				INVERT_ATTENTION				INVERT_DRI_DATA				INVERT_DRI_CLOCK				REVERSE_FRAME				RSVD										SWIZZLED_FRAME									

Table 1053. HW_DRI_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	INVERT_PILOT	RW	0x0	Set to 1 to 1 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	Set to 1 to 1 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	Set to 1 to 1 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	Set to 1 to 1 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	Set to 1 to 1 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.
26:18	RSVD	RO	0x0	Reserved.
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,

```
frame = HW_DRI_DEBUG1.SWIZZED_FRAME; // then pilot peak is set for this frame;
```

This register indicates the version of the block for debug purposes.

Table 1054. HW DRI VERSION

Table 1055. HW_DRI_VERSION Bit Field Descriptions

DRI Block v1.1

28. LOW-RESOLUTION ADC AND TOUCH-SCREEN INTERFACE

This chapter describes the low-resolution analog-to-digital converters and touch-screen interface included on the STMP3770. It includes sections on scheduling conversions and delay channels. Programmable registers are described in [Section 28.4](#).

28.1. Overview

The sixteen-channel low-resolution ADC (LRADC) block is used for voltage measurement [Figure 131](#) shows a block diagram of the LRADC. Eight “virtual” channels can be used at one time. Each of the eight virtual channels can be mapped to any of the 16 physical channels using HW_LRADC_CTRL4. Six physical channels are available for general use.

- Channel 15 is dedicated to measuring the voltage on the VDD5V pin and can be used to detect possible issues with 5V rail drooping.
- Channel 14 is dedicated to measuring the bandgap reference voltage and can be used to calibrate out a portion of the LRADC measurement error (comparator offset, buffer amp offset, and DAC offset). In most cases, the bandgap reference error (specified to $\pm 1\%$) will dominate the total LRADC error, and this calibration will not be helpful. But if the bandgap reference is calibrated using the fuses, then it is possible that LRADC accuracy will be limited by these other sources and that using the VBG input for calibration of the LRADC can improve accuracy further.
- Channel 12 and 13 are dedicated to measuring the voltage on the USB_DP and USB_DN pins. This is to be used only in non-USB mode and can be used for special peripheral circuitry detection. HW_USBPHY_CTRL_DATA_ON_LRADC must be set to measure these inputs.
- Channel 10 and 11 are reserved inputs for analog testing.
- Channel 8 and 9 are dedicated to measuring the internal die temperature. HW_LRADC_CTRL2_TEMPSENSE_PWD must be cleared for these inputs to function. See [Section 28.2.2](#).
- Channel 7 is dedicated to measuring the voltage on the BATT pin and can be used to sense the amount of battery life remaining.
- Channel 6 is dedicated to measuring the voltage on the VDDIO Rail and is used to calibrate the voltage levels measured on the auxiliary channels when those inputs are resistor divided from the VDDIO rail.
- The other channels, LRADC0–LRADC5, measure the voltage on the six application-dependent LRADC pins (only LRADC 0 and 1 are available in the 100-pin packages). The auxiliary channels can be used for a variety of uses, including a resistor-divider-based wired remote control, external temperature sensing, touch-screen, etc.

The LRADC has 12 bits of resolution and an absolute accuracy of 1.3% limited primarily by the bandgap voltage reference accuracy. If the bandgap voltage reference is calibrated with the fuses, the LRADC absolute accuracy might be improved to better than 0.5%. 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.

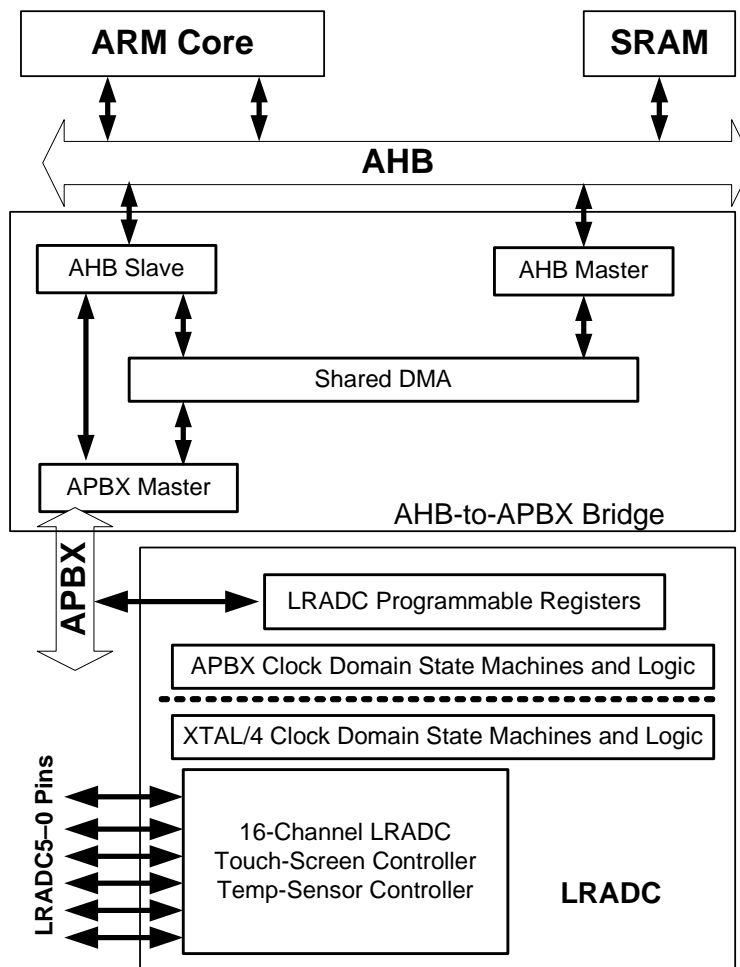


Figure 131. Low-Resolution ADC and Touch-Screen Interface Block Diagram

28.2. Operation

All channels of the LRADC share a common successive approximation style analog-to-digital converter through a common analog mux front end (see [Figure 132](#)).

- The BATT pin has a built-in 4:1 voltage divider on its analog multiplexer input that is activated only in Li-Ion battery mode.
- The Channel 15 5V input also has a built-in 4:1 divider on its input.
- The Channel 6 VDDIO input has a built-in 2:1 divider on its input. The maximum analog input voltage into the LRADC is 1.85 V.
- For input channels (other than BATT, 5V, or VDDIO) with signals larger than 1.85 V, the divide-by-two option should be set (HW_LRADC_CTRL2_DIVIDE_BY_TWO). With the DIVIDE_BY_TWO option set, the maximum input voltage is VDDIO – 50mv.

The touch-screen driver works for typical touch-screen impedances of 200–900 ohm and for high impedance touch-panels with impedances in the 50-Kohm range.

The LRADC channels 0 and 1 have optional current source outputs to allow these channels to be used with an external thermistor (or an external diode) for temperature sensing. The controls for these current sources are in HW_LRADC_CTRL2. The current source values can be changed to allow significant temperature sensing range using a thermistor or to use a diode for temperature sensing. The currents are derived using the on-chip 1% accurate bandgap voltage reference and an optionally tuned on-chip poly resistor. The accuracy of the current source is limited by the on-chip resistor, which should be 5% accurate and optionally tuned for higher accuracy (with eFuses). Most thermistors are no more than 5% accurate, so this level of current source accuracy is acceptable for most applications. For temperature sensing with higher accuracy, customers can use a 1% resistor divider from VDDIO with the thermistor. In this case, the thermistor will be the dominant source of error.

28.2.1. External Temperature Sensing with a Diode

Using a diode instead of a thermistor for external temperature sensing can be cheaper and provide greater temperature range for a given accuracy level. A cheap diode like a 1N4148 is connected between ground and either LRADC 0 or 1. Two voltage measurements are taken—first with the HW_LRADC_CTRL2_TEMP_ISRC current source set at 300 μ A, then another voltage measurement with the current source set at 20 μ A. The temperature will be roughly:

$$\text{degrees Kelvin} = (V_{\text{max}} - V_{\text{min}}) / 0.409 \text{ mV}$$

or, from the LRADC conversion (LSB=0.45mV):

$$\text{degrees Kelvin} = (\text{Codemax} - \text{Codemin}) * 1.104$$

SigmaTel recommends taking 5–10 samples for the min and max and then averaging them to get a good reading.

The temperature reading error will likely be dominated by part-to-part matching of the diodes. Some manufacturers' diodes show substantially less variation than others. SigmaTel has shown 3sigma accuracy of +/-7.5C using Fairchild MMBD914 (from multiple batches of diodes).

If better accuracy is required, SigmaTel recommends using a thermistor for external temperature sensing. The thermistor will be more accurate, but over a smaller temperature range than the diode method.

Any routing impedance to the diode will cause a shift in the temperature reading. This can be measured and corrected in software for each design.

Two ohms of routing impedance would cause $(2 * (300\mu\text{A} - 20\mu\text{A}))$ error of 0.56 mV or 1.25 degrees C error.

28.2.2. Internal Die Temperature Sensing

The STMP3770 has a new internal die temperature sensor that uses two of the sixteen physical LRADC channels. To use the internal die temperature sensor, HW_LRADC_CTRL2_TEMPSENSE_PWD should be cleared. (This bit can be left cleared after power up. There is no need to toggle it on and off.) Two of the eight virtual LRADC channels need to be mapped to the temperature sensing channels 8 and 9 using HW_LRADC_CTRL4. Then, these virtual LRADC channels should BOTH be converted using the LRADC conversion scheduler described below. The temperature in degrees Kelvin will be equal to:

$$(\text{Channel9} - \text{Channel8}) * \text{Gain_correction} / 4$$

The Gain_correction corrects a mean gain error in the temperature conversion and should be 1.012. After this correction factor, the three-sigma error of the

temperature sensor should be within $\pm 1.5\%$ in degrees Kelvin. Additionally, the temperature sampling has a three-sigma sample-to-sample variation of 2 degrees Kelvin. If desired, this error can be removed by oversampling and averaging the temperature result.

Prior to starting a battery charge cycle, the internal die temperature sensing could be used for an approximate ambient temperature. During high-current battery charging, the temperature sensor can be used as extra protection to avoid excessive die temperatures (to throttle the charging current).

28.2.3. Scheduling Conversions

The APBX clock domain logic schedules conversions on a per-channel basis and handles interrupt processing back to the CPU. Each of the eight virtual channels has its own interrupt request enable bit and its own interrupt request status bit.

A schedule request bit, `HW_LRADC_CTRL0_SCHEDULE`, exists for each virtual channel. Setting this bit causes the LRADC to schedule a conversion for that virtual channel. Each virtual 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 virtual 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.

28.2.4. 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.

NOTE: The `DELAY` fields in `HW_LRADC_DELAY0`, `HW_LRADC_DELAY1`, `HW_LRADC_DELAY2`, and `HW_LRADC_DELAY3` must be non-zero; otherwise, the LRADC will not trigger the delay group. The `ACCUMULATE` bit in the appropriate channel register `HW_LRADC_CHn` must be set to 1 if `NUM_SAMPLES` is greater than 0; otherwise, the IRQs will not fire.

Consider the case of a touch-screen that requires 4x 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 set up 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, set up 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 TRIGGER_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 133](#).

NOTE: If a delay group schedules channels to be sampled and a manual write to the schedule field in CTRL0 occurs while the block is discarding samples, the LRADC will switch to the new schedule and will not sample the channels that were previously scheduled. The time window for this to happen is very small and lasts only while the LRADC is discarding samples.

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.

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 34.4.10, “Correct Way to Soft Reset a Block” on page 1013](#) for additional information on using the SFTRST and CLKGATE bit fields.

Chapter 28: Low-Resolution ADC and Touch-Screen Interface 5-37xx-DS2-1.04-031408

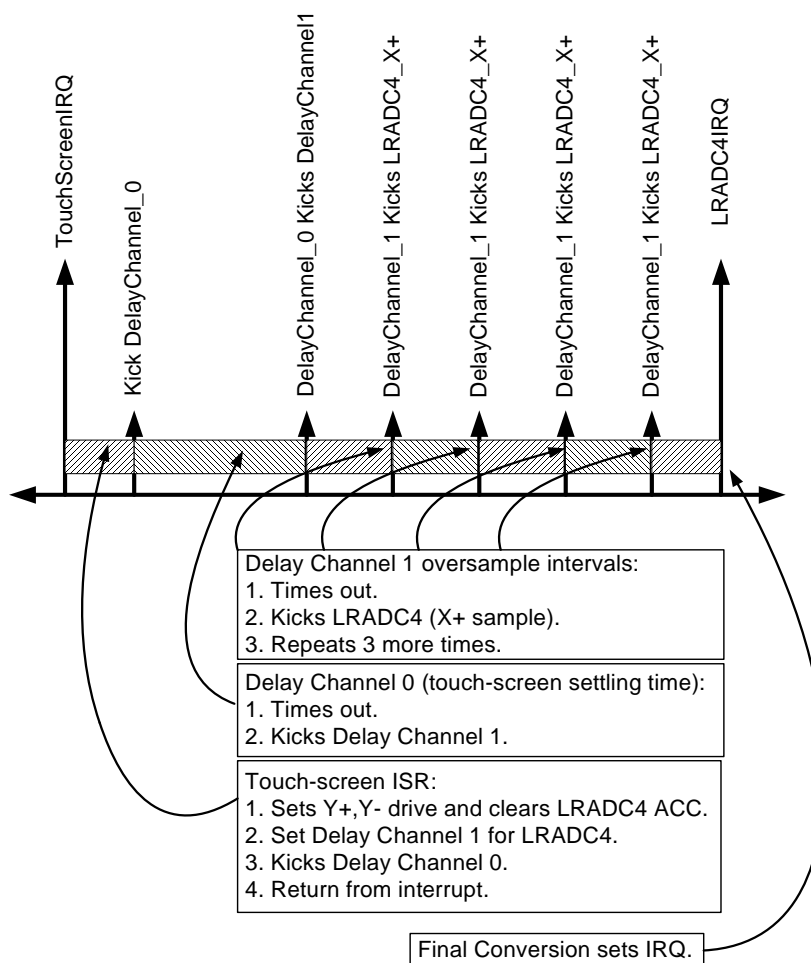


Figure 133. Using Delay Channels to Oversample a Touch-Screen

28.4. Programmable Registers

28.4.1. 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 1056. HW_LRADC_CTRL0

[illegible]

Table 1057. HW_LRADC_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	SFTRST	RW	0x1	When set to 1, 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. Clear this bit to 0 for normal operation.
30	CLKGATE	RW	0x1	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block.
29:22	RSVD	RO	0x0	Reserved.
21	ONCHIP_GROUNDREF	RW	0x0	Set this bit to 1 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 1 to enable touch-panel touch detector. OFF = 0x0 Turn it off. ON = 0x1 Turn it on. When set, the “softpu_xplus” and the switch is activated and the Y- pin is pulled down (although it doesn’t set the “drive_yminus” bit).
19	YMINUS_ENABLE	RW	0x0	Set this bit to 1 to enable yminus pulldown on the LRADC5 pin. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
18	XMINUS_ENABLE	RW	0x0	Set this bit to 1 to enable 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 1 to enable yplus pullup on the LRADC3 pin. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.
16	XPLUS_ENABLE	RW	0x0	Set this bit to 1 to enable xplus pullup on the LRADC2 pin. OFF = 0x0 Turn it off. ON = 0x1 Turn it on.

Table 1057. HW_LRADC_CTRL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:8	RSVD	RO	0x00	Reserved.
7:0	SCHEDULE	RW	0x00	Setting a bit to 1 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 1. Thus, any thread can request a conversion asynchronously from any other thread.

DESCRIPTION:

The LRADC Control Register 0 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);
```

28.4.2. 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 1058. 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					
RSVD							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	RSVD												TOUCH_DETECT_IRQ	LRADC7_IRQ	LRADC6_IRQ	LRADC5_IRQ	LRADC4_IRQ	LRADC3_IRQ	LRADC2_IRQ	LRADC1_IRQ	LRADC0_IRQ

Table 1059. HW_LRADC_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:25	RSVD	RO	0x00	Reserved.
24	TOUCH_DETECT_IRQ_EN	RW	0x0	Set to 1 to enable an interrupt for the touch detector comparator. DISABLE = 0x0 Disable interrupt request. ENABLE = 0x1 Enable interrupt request.

STMP3770

Table 1059. HW_LRADC_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23	LRADC7_IRQ_EN	RW	0x0	Set to 1 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 1 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 1 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 1 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 1 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 1 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 1 to enable an interrupt for Channel 1 conversions. DISABLE = 0x0 Disable interrupt request. ENABLE = 0x1 Enable interrupt request.
16	LRADC0_IRQ_EN	RW	0x0	Set to 1 to enable an interrupt for Channel 0 conversions. DISABLE = 0x0 Disable interrupt request. ENABLE = 0x1 Enable interrupt request.
15:9	RSVD	RO	0x00	Reserved.
8	TOUCH_DETECT_IRQ	RW	0x0	This bit is set to 1 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 interrupt. 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 1 upon completion of a scheduled conversion for Channel 7 (BATT). It is ANDed with its corresponding interrupt enable bit to request an interrupt. 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 1059. HW_LRADC_CTRL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	LRADC6_IRQ	RW	0x0	This bit is set to 1 upon completion of a scheduled conversion for Channel 6 (VDDIO). It is ANDed with its corresponding interrupt enable bit to request an interrupt. 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 1 upon completion of a scheduled conversion for Channel 5. It is ANDed with its corresponding interrupt enable bit to request an interrupt. 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 1 upon completion of a scheduled conversion for Channel 4. It is ANDed with its corresponding interrupt enable bit to request an interrupt. 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 1 upon completion of a scheduled conversion for Channel 3. It is ANDed with its corresponding interrupt enable bit to request an interrupt. Once set by the conversion hardware, this bit remains set until cleared by software. CLEAR = 0x0 Interrupt request cleared. PENDING = 0x1 Interrupt request pending.
2	LRADC2_IRQ	RW	0x0	This bit is set to 1 upon completion of a scheduled conversion for Channel 2. It is ANDed with its corresponding interrupt enable bit to request an interrupt. 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 1 upon completion of a scheduled conversion for Channel 1. It is ANDed with its corresponding interrupt enable bit to request an interrupt. 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 1 upon completion of a scheduled conversion for Channel 0. It is ANDed with its corresponding interrupt enable bit to request an interrupt. 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

```
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;
}
```

The LRADC Control Register 2 provides overall control of the eight low-resolution analog-to-digital converters.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
DIVIDE_BY_TWO								BL_AMP_BYPASS		BL_ENABLE		BL_MUX_SELECT		BL_BRIGHTNESS								TEMPSENSE_PWD		RSVD		EXT_EN1		EXT_EN0		RSVD		TEMP_SENSOR_IENABLE1		TEMP_SENSOR_IENABLE0		TEMP_ISRC1								TEMP_ISRC0							

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 1 causes the A/D converter to use its analog divide-by-two circuit for the conversion of the corresponding channel.
23	BL_AMP_BYPASS	RW	0x0	The analog feedback control signal is normally gained up by 4X. When this bit is 1, the feedback control signal bypasses the gain of 4 stage. DISABLE = 0x0. ENABLE = 0x1.
22	BL_ENABLE	RW	0x0	Enables the back light.
21	BL_MUX_SELECT	RW	0x0	0 = Use pin LRADC4 for feedback control. 1 = Use pin LRADC1 for feedback control.

Table 1061. HW_LRADC_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20:16	BL_BRIGHTNESS	RW	0x0	Sets the voltage comparison level for the analog feedback control. Each step is -1.293 dB with the max voltage of 1.212 V. BL_AMD_BYPASS 10 11111 = 1.213 V 0.303 V 11110 = 1.046 V 0.262 V 00001 = 0.0136 V 0.0034 V 00000 = 0.0117 V 0.0029 V
15	TEMPSENSE_PWD	RW	0x1	PWD the tempsense block. ENABLE = 0x0 When this is low, the tempsense gain block muxes to LRADC channels 8 and 9. DISABLE = 0x1.
14	RSVD	RO	0x00	Reserved.
13	EXT_EN1	RW	0x0	These bits are not supported. DISABLE = 0x0. ENABLE = 0x1.
12	EXT_EN0	RW	0x0	When cleared to 0 (default), the mux amp is bypassed when the LRADC input channel is not using the divide-by-two. When set to 1, the mux amp is never bypassed (old behavior).
11:10	RSVD	RO	0x00	Reserved.
9	TEMP_SENSOR_IENABLE1	RW	0x0	Set this bit to 1 to enable the current source onto LRADC1. DISABLE = 0x0 Disable temperature sensor current source. ENABLE = 0x1 Enable temperature sensor current source.
8	TEMP_SENSOR_IENABLE0	RW	0x0	Set this bit to 1 to enable the current source onto LRADC0. DISABLE = 0x0 Disable temperature sensor current source. ENABLE = 0x1 Enable temperature sensor current source.

Table 1061. HW_LRADC_CTRL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
7:4	TEMP_ISRC1	RW	0x0	<p>When the output voltage is lower than 1 V, the output current is 1 μA higher than the decode shown above. This extra current drops to 0 as the output voltage raises above 1.5 V.</p> <p>This four-bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC1.</p> <p>300 = 0xF 300 μA. 280 = 0xE 280 μA. 260 = 0xD 260 μA. 240 = 0xC 240 μA. 220 = 0xB 220 μA. 200 = 0xA 200 μA. 180 = 0x9 180 μA. 160 = 0x8 160 μA. 140 = 0x7 140 μA. 120 = 0x6 120 μA. 100 = 0x5 100 μA. 80 = 0x4 80 μA. 60 = 0x3 60 μA. 40 = 0x2 40 μA. 20 = 0x1 20 μA. ZERO = 0x0 0 μA.</p>
3:0	TEMP_ISRC0	RW	0x0	<p>When the output voltage is lower than 1 V, the output current is 1 μA higher than the decode shown above. This extra current drops to 0 as the output voltage raises above 1.5 V.</p> <p>This four-bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC0.</p> <p>300 = 0xF 300 μA. 280 = 0xE 280 μA. 260 = 0xD 260 μA. 240 = 0xC 240 μA. 220 = 0xB 220 μA. 200 = 0xA 200 μA. 180 = 0x9 180 μA. 160 = 0x8 160 μA. 140 = 0x7 140 μA. 120 = 0x6 120 μA. 100 = 0x5 100 μA. 80 = 0x4 80 μA. 60 = 0x3 60 μA. 40 = 0x2 40 μA. 20 = 0x1 20 μA. ZERO = 0x0 0 μA.</p>

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) ;
```


STMP3770

Table 1063. HW_LRADC_CTRL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
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	RSVD	RO	0x0	Reserved.
5:4	HIGH_TIME	RW	0x0	Changes the duty cycle (time high) for the LRADC clock. When CYCLE_TIME = 00, only 00 and 01 are valid for HIGH_TIME. When CYCLE_TIME = 01, only 00, 01, and 10 are valid. 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.
3:2	RSVD	RO	0x0	Reserved.
1	DELAY_CLOCK	RW	0x0	Set this bit to 1 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 1 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 1 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 DC-DC converter. NORMAL = 0x0 Run the clock in normal, that is, not inverted mode. INVERT = 0x1 Invert the clock.

DESCRIPTION:

The LRADC Control Register 3 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);
```


Table 1067. HW_LRADC_CH0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:18	RSVD	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 256K. 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 individually 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;

```

28.4.7. 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 1068. HW_LRADC_CH1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
TOGGLE	RSVD	ACCUMULATE	NUM_SAMPLES					RSVD						VALUE																	

Table 1069. HW_LRADC_CH1 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	RSVD	RO	0x0	Reserved.
29	ACCUMULATE	RW	0x0	Set this bit to 1 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. Clear this field to 0 for a single conversion per interrupt.
23:18	RSVD	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 256K. 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 individually 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.
}

```


Table 1073. HW_LRADC_CH3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:18	RSVD	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 256K. 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 individually 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;

```

28.4.10. 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 1074. HW_LRADC_CH4

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
TOGGLE	RSVD	ACCUMULATE	NUM_SAMPLES					RSVD					VALUE																			

Table 1075. HW_LRADC_CH4 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	RSVD	RO	0x0	Reserved.
29	ACCUMULATE	RW	0x0	Set this bit to 1 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. Clear this field to 0 for a single conversion per interrupt.
23:18	RSVD	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 256K. 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 individually 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;

```

28.4.11. LRADC 5 Result Register Description

The LRADC 5 Result Register returns the 12-bit result for low-resolution analog-to-digital converter Channel 5.

HW_LRADC_CH5	0x800500A0
HW_LRADC_CH5_SET	0x800500A4
HW_LRADC_CH5_CLR	0x800500A8
HW_LRADC_CH5_TOG	0x800500AC

Table 1076. HW_LRADC_CH5

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
TOGGLE	RSVD	ACCUMULATE	NUM_SAMPLES					RSVD						VALUE																	

Table 1077. HW_LRADC_CH5 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	RSVD	RO	0x0	Reserved.
29	ACCUMULATE	RW	0x0	Set this bit to 1 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. Clear this field to 0 for a single conversion per interrupt.
23:18	RSVD	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 256K. 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 individually 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)
{

```


Table 1081. HW_LRADC_CH7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
28:24	NUM_SAMPLES	RW	0x0	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Clear this field to 0 for a single conversion per interrupt.
23:18	RSVD	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 256K. 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 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 individually 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;

```

28.4.14. LRADC Scheduling Delay Register 0 Description

The LRADC Scheduling Delay Register 0 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 1082. HW LRADC DELAY0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
TRIGGER_LRDCS								RSVD		KICK	TRIGGER_DELAYS					LOOP_COUNT					DELAY										

Table 1083. HW_LRADC_DELAY0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	TRIGGER_LRADC	RW	0x00	Setting a bit in this bit field to 1 causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches 0. 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	RSVD	RO	0x0	Reserved.
20	KICK	RW	0x0	Setting this bit to 1 initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADC or TRIGGER_DELAYS will start.
19:16	TRIGGER_DELAYS	RW	0x0	Setting a bit in this bit field to 1 causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches 0. 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 0. At 0, 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.

Table 1085. HW_LRADC_DELAY1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	KICK	RW	0x0	Setting this bit to 1 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 1 causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches 0. 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 0. At 0, 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 delay channel 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 1.

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.
```

28.4.16. LRADC Scheduling Delay Register 2 Description

The LRADC Scheduling Delay Register 2 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 1086. 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								RSVD		KICK	TRIGGER_DELAYS					LOOP_COUNT					DELAY										

Table 1087. 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 1 causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches 0. 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	RSVD	RO	0x0	Reserved.
20	KICK	RW	0x0	Setting this bit to 1 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 1 causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches 0. 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 0. At 0, 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.

Table 1089. HW_LRADC_DELAY3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
20	KICK	RW	0x0	Setting this bit to 1 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 1 causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches 0. 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 0. At 0, 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 delay channel 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 1.

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.
```

28.4.18. 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 1093. HW_LRADC_DEBUG1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15:13	RSVD	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 that is decremented upon each Channel 7 conversion. When that counter decrements to 0, 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	RSVD	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 DEBUG1 register 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 ) ;
```

28.4.20. 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 1094. HW_LRADC_CONVERSION

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD											AUTOMATIC	RSVD	SCALE_FACTOR	RSVD	SCALED_BATT_VOLTAGE																

Table 1095. HW_LRADC_CONVERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	RSVD	RO	0x0	Reserved.
20	AUTOMATIC	RW	0x0	Control the automatic update mode of the BATT_VAL bit field in the HW_POWER_BATTMONITOR 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	RSVD	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	RSVD	RO	0x0	Reserved.
9:0	SCALED_BATT_VOLTAGE	RW	0x80	This is the LRADC battery voltage after scaling by the SCALE_FACTOR, such that the LSB is ~8mV. This value is used by the DCDC converter.

DESCRIPTION:

This register controls the voltage scaling multiplier, as shown in the following equations:

- For NiMH: LRADC battery voltage * 29 divided by 512
- For dual NiMH: LRADC battery voltage * 29 divided by 256
- For Li-Ion: LRADC battery voltage * 29 divided by 128

EXAMPLE:

```
HW_LRADC_CONVERSION.AUTOMATIC = 1;
```


Table 1097. HW_LRADC_CTRL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
23:20	LRADC5SELECT	RW	0x5	This bit field selects which analog mux input is used for conversion on LRADC Channel 5. 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 USB_DP (must also set usb_data_on_lradc to use this) CHANNEL13 = 0xD USB_DN (must also set usb_data_on_lradc to use this) CHANNEL14 = 0xE VBG (Can be used to calibrate the LRADC) CHANNEL15 = 0xF 5V input
19:16	LRADC4SELECT	RW	0x4	This bit field selects which analog mux input is used for conversion on LRADC Channel 4. 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 USB_DP (must also set usb_data_on_lradc to use this) CHANNEL13 = 0xD USB_DN (must also set usb_data_on_lradc to use this) CHANNEL14 = 0xE VBG (Can be used to calibrate the LRADC) CHANNEL15 = 0xF 5V input
15:12	LRADC3SELECT	RW	0x3	This bit field selects which analog mux input is used for conversion on LRADC Channel 3. 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 USB_DP (must also set usb_data_on_lradc to use this) CHANNEL13 = 0xD USB_DN (must also set usb_data_on_lradc to use this) CHANNEL14 = 0xE VBG (Can be used to calibrate the LRADC) CHANNEL15 = 0xF 5V input

Table 1097. HW_LRADC_CTRL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11:8	LRADC2SELECT	RW	0x2	This bit field selects which analog mux input is used for conversion on LRADC Channel 2. 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 USB_DP (must also set usb_data_on_lradc to use this) CHANNEL13 = 0xD USB_DN (must also set usb_data_on_lradc to use this) CHANNEL14 = 0xE VBG (Can be used to calibrate the LRADC) CHANNEL15 = 0xF 5V input
7:4	LRADC1SELECT	RW	0x1	This bit field selects which analog mux input is used for conversion on LRADC Channel 1. 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 USB_DP (must also set usb_data_on_lradc to use this) CHANNEL13 = 0xD USB_DN (must also set usb_data_on_lradc to use this) CHANNEL14 = 0xE VBG (Can be used to calibrate the LRADC) CHANNEL15 = 0xF 5V input
3:0	LRADC0SELECT	RW	0x0	This bit field selects which analog mux input is used for conversion on LRADC Channel 0. 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 USB_DP (must also set usb_data_on_lradc to use this) CHANNEL13 = 0xD USB_DN (must also set usb_data_on_lradc to use this) CHANNEL14 = 0xE VBG (Can be used to calibrate the LRADC) CHANNEL15 = 0xF 5V input

DESCRIPTION:

Each virtual channel of the LRADC can be directed to use any of the 16 analog mux input sources for it conversion. This register specifies the analog mux to channels to be used for LRADC virtual channels 0 through 7.

STMP3770



29. POWER SUPPLY

This chapter describes the power supply subsystem provided on the STMP3770. It includes sections on the DC-DC converter, linear regulators, PSWITCH pin functions, battery monitor and charger, and silicon speed sensor. Programmable registers are described in [Section 29.11](#).

29.1. Overview

The STMP3770 integrates a comprehensive power supply subsystem, including the following features:

- One integrated DC-DC converter that supports 1-cell alkaline/NiMH and Li-Ion batteries.
- Three 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 VDDD, VDDA, VDDIO, 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 (1-cell alkaline/NiMH only).

The STMP3770 power supply is designed to offer maximum flexibility and performance, while minimizing external component requirements. [Figure 134](#) shows a functional block diagram of the power supply components including three linear regulators, battery charge support, as well as battery monitoring, supply brownout detection, and silicon process sensors. [Figure 135](#) and [Figure 136](#) show the three-output single-inductor DC-DC converter for the two supported battery types. These three figures can be used to understand which register and status bits relate to which subsystems, but they are not intended to be a complete architecture description.

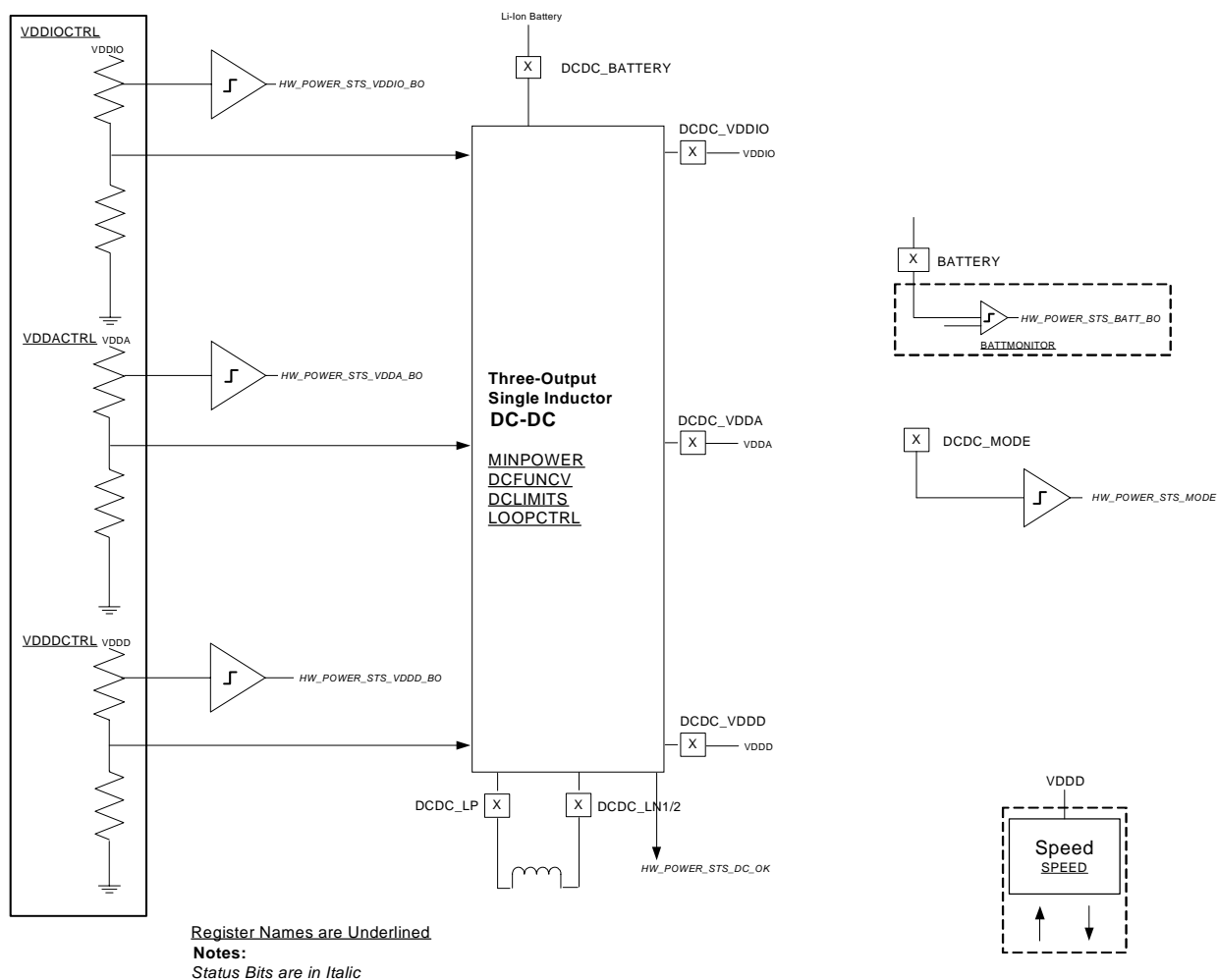


Figure 135. Li-Ion DC-DC Supply Block Diagram

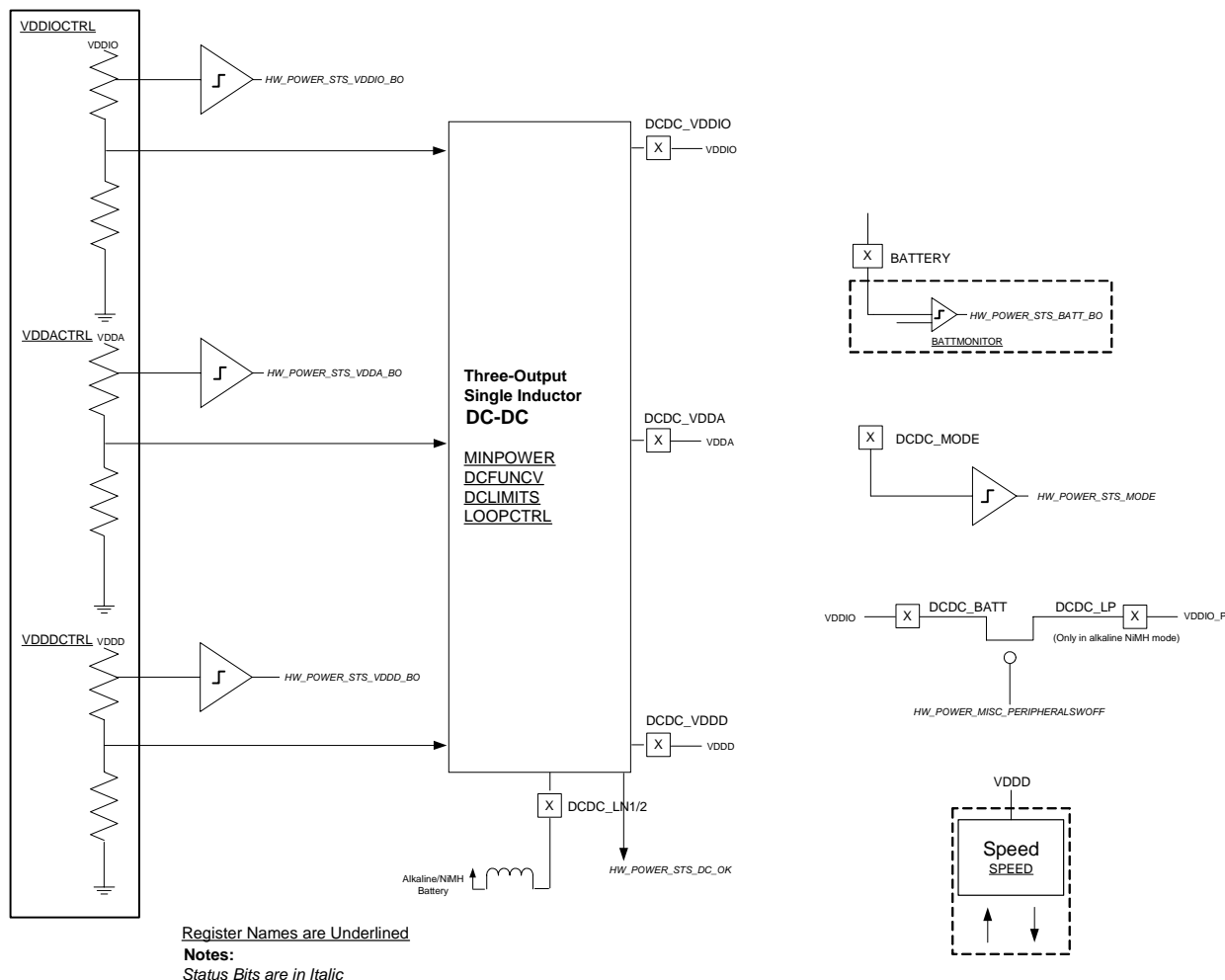


Figure 136. 1-Cell Alkaline/NIMH DC-DC Supply Block Diagram

29.2. DC-DC Converters

The DC-DC converter efficiently scales battery voltage to the required supply voltages. The DC-DC converters include several advanced features:

- Flexible battery support for either 1-cell alkaline/NiMH or Li-Ion batteries
- Single inductor architecture
- Programmable output voltages
- Programmable brownout detection thresholds
- Pulse frequency modulation (PFM) mode for low-current load operation

29.2.1. 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 configuration is used. [Table 1100](#) lists the two DC-DC battery modes.

Table 1100. DC-DC Battery Modes

DC-DC MODE PIN	MODE	BATT	VDDD	VDDA	VDDIO	INDUCTOR	PEAK POWER	COMMENTS
Open	1	1-Cell alkaline/NiMH 0.9V to 1.6V	Boost	Boost	Boost	1	250mW @ 1.0V	Low cost and area
Short	0	Li-Ion 2.9V to 4.2V	Buck	Buck	Buck/Boost	1	1.5W @ 3.1V	Low cost and area

The STMP3770 DC-DC converter buck/boost architecture maintains a fixed VDDIO voltage even when the battery voltage drops below the VDDIO voltage. 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 nears the VDDIO voltage.

29.2.2. DC-DC Operation

The STMP3770 DC-DC converter enables a low-power system and features programmable output voltages and control modes. Most products adjust VDDD dynamically to provide the minimum voltage required for proper system operation. VDDIO and VDDA are typically set once during system initialization and not changed during operation.

29.2.2.1. 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 DISABLE_XTALOK 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 will typically disable the functionality of PWDN_5VBRNOUT and PWDN_BATTBRNOUT after system configuration is complete, as shown in [Figure 137](#). 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 output target voltage is set to some voltage margin above the minimum operating level via the TRG field in the HW_POWER_VDDCTRL, HW_POWER_VDDACTRL, and HW_POWER_VDDIOCTRL registers. The brownout detection threshold is also set via the BO_OFFSET field in the same three registers. The BO_OFFSET field determines how far the brownout voltage is below the

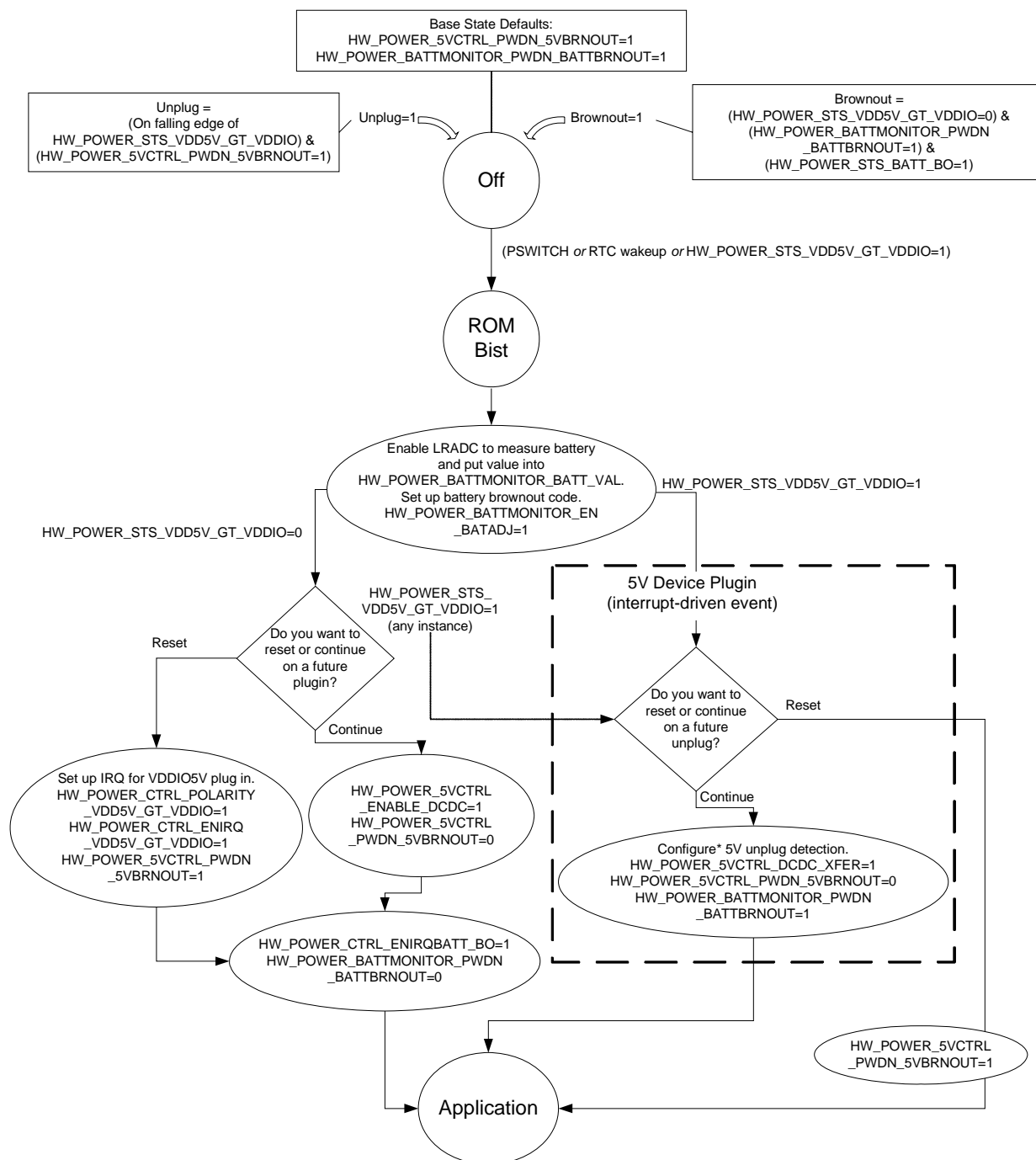
output target voltage for each supply and might typically be set 75–100 mV below the target voltage. If the voltage drops to the brownout detector's level, then it optionally triggers a CPU Fast Interrupt (FIQ). The CPU can then alleviate the problem and/or shut down the system elegantly. See [Chapter 2, “Characteristics and Specifications” on page 43](#) 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 STMP3770 should include at least 10 μ F 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 137](#).

29.2.2.2. DC-DC Extended Battery Life Features

The DC-DC converter has several other power-reducing programmable modes that are useful for maximizing battery life:

- **Li-Ion Buck/Boost**—The Li-Ion battery configuration supports 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.
- **Transient Loading Optimizations**—Several new incremental improvements have been made to the control architecture of the switching converters. At this time, SigmaTel recommends setting the following bits via software in HW_POWER_LOOPCTRL to obtain maximum efficiency and minimum supply ripple: TOGGLE_DIF, EN_CM_HYST, and EN_RCSCALE=0x1. Also, set HW_POWER_BATTMONITOR_EN_BATADJ after programming VDDD and VDDIO fields in the HW_POWER_DCFUNCV register. The complete settings to optimize transient loading will be dependent on the application, component selection, and board layout.
- **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. The DC-DC converter can be separately placed in PFM mode via the EN_DC_PFM bit in HW_POWER_MINPWR.
- **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 DC_HALFCLK in HW_POWER_MINPWR. The DC-DC converter can also be programmed to a frequency that is based on the PLL using the SEL_PLLCLK and FREQSEL fields in the HW_POWER_MISC register.
- **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 DC_STOPCLK in HW_POWER_MINPWR. This bit is not intended to power down the whole system. Use the HW_POWER_RESET bits to power the system off.



Note:* See [Section 29.3.2.1](#).

Figure 137. Brownout Detection Flowchart

29.3. Linear Regulators

The STMP3770 integrates three linear regulators that are typically used when the system is powered from a 5-V supply. All of these regulators have an output impedance of approximately one-quarter ohm. This means that the measured output voltage will be slightly dependent on the current consumed on each supply. Architecturally, one regulator generates VDDIO from the VDD5V pin, one generates VDDA from VDDIO, and the other generates VDDD from the VDDA supply. Therefore, all of the current is supplied by the VDD5V->VDDIO regulator.

The circuitry implemented to meet USB in-rush requirements places restrictions on the static current allowed in the system to successfully power up from 5V.

For system current limits when powering up from 5V = 4.35 V:

- $IVDDD + IVDDA < 35 \text{ mA}$
- $IVDDD + IVDDA + IVDDIO < 70 \text{ mA}$

After power-up is complete, the linear regulators automatically change current limits.

For system current limits after power-up when using linear regulators, 5V = 4.7 V:

- $IVDDD < 100 \text{ mA}$ ($VDD \leq 1.45 \text{ V}$)
- $IVDDD + IVDDA < 200 \text{ mA}$ ($VDDA = 1.8 \text{ V}$)
- $IVDDD + IVDDA + IVDDIO < 300 \text{ mA}$ ($VDDIO = 3.1 \text{ V}$)

It should also be noted that the VDD5V voltage can dynamically change as the product is plugged into a USB port or other 5-V supply. The STMP3770 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.

29.3.1. 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 <100 mA, nominally 50 mA, and is active by default. This current limit is disabled in hardware after power-up, but can be re-enabled via the HW_POWER_5VCTRL_ENABLE_ILIMIT by software. System designers must understand that the current inrush limit during 5-V power-up places restrictions on application current consumption until it is disabled. Specifically, after connection to 5V, 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 5V. This requirement can be met by setting HW_POWER_5VCTRL_ILIMIT_EQ_ZERO when the OTG application is active. The comparators required for OTG can be enabled by HW_POWER_5VCTRL_OTG_PWRUP_CMPS. It is also possible to change the threshold of the Vbus valid comparator using HW_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 must fully power down after VDD5V becomes invalid. It can auto restart with the DC-DC converter if HW_RTC_PERSISTENT0_AUTO_RESTART is set.

29.3.2. 5V to Battery Power Interaction

The STMP3770 supports several different options related to the interaction of the switching converters with the linear regulators. The two primary options are a reset

on 5V insertion/removal or a handoff to the DC-DC converters that is invisible to the end-user of the application. [Figure 138](#) includes these two options as the two system architecture decision boxes.

29.3.2.1. Battery Power to 5-V Power

By default, the DC-DC converter turns off when VDD5V becomes valid and the system does not reset. If the system is operating from the DC-DC converter and using more current than the linear regulators can supply, then the VDDD, VDDA, and VDDIO rails will droop when 5V is attached and the system may brownout and shut down. To avoid this issue, set the ENABLE_DCDC bit and set the LINREG_OFFSET fields to 0b2 in anticipation of VDD5V becoming present. The ENABLE_DCDC bit will cause the DC-DC converter to remain on even after 5V is connected and, thus, guarantee a stable supply voltage until the system is configured for removal of 5V. The LINREG_OFFSET fields = 0b2 cause the linear regulators to regulate to a lower target voltage than the switching converter and prevent unwanted interaction between the two power supplies. After the system is configured for removal of 5V, ENABLE_DCDC can be set low and ENABLE_ILIMIT set low in register HW_POWER_5VCTRL to allow the linear regulators to supply the system power, if desired.

29.3.2.2. 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 5V.

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 28, “Low-Resolution ADC and Touch-Screen Interface” on page 801](#). 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 5V 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 prevent high current contention between the linear regulator and DC-DC converter, it is very important to set the LINREG_OFFSET = 0b1X in the HW_POWER_VDDIOCTRL, HW_POWER_VDDACTRL, and HW_POWER_VDDDCTRL registers. Finally, set DCDC_XFER and clear PWDN_5VBRNOUT in the HW_POWER_5VCTRL register. Following this exact sequence is crucial because it is safe to disable the power-down-on-unplug functionality of the device only after the system is completely ready for a transition to battery power.

29.3.2.3. 5-V Power and Battery Power

Battery charge can also be enabled in Li-Ion configurations to provide additional power efficiency when connected to 5V, if the switching converter is also enabled. This can be useful because the buck switching converters will efficiently convert the battery voltage to the desired VDDA, VDDD, and VDDIO voltages.

29.4. Power-Up and Power-Down

29.4.1. Power-Up Sequence

The DC-DC converter controls the power-up and reset of the STMP3770. The power-up sequence begins when the battery is connected to the BATT pin of the device (or a 5V source is connected to the VDD5V pin). Either the BATT pin or VDD5V provides power 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 VDDD and VDDA supplies are held at ground, while the VDDIO rail is either shorted to the battery (alkaline) or held at ground (Li-Ion). 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 V 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 VDDD, VDDA and VDDIO rails to their default voltages. If VDD5V is not valid, then the DC-DC supplies the VDDD, VDDA, and VDDIO 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 or 5V is removed, the system returns to the off state.

The power-up time is dependent on the VDDD/VDDA/VDDIO load and battery or VDD5V voltage, but should be less than 100 ms. The VDDD/VDDA/VDDIO load should be minimal during power up to ensure proper startup of the DC-DC converter.

There is an integrated 5-K Ω resistor that can be switched in between the VDDXTAL pin and PSWITCH. If enabled using HW_RTC_PERSISTENT0_AUTO_RESTART, then the device immediately begins the power-up sequence after power-down.

29.4.2. Power-Down Sequence

Power-down is also controlled by the DC-DC converters. When the DC-DC converter detects a power-down event, it returns 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.
- Error condition occurs, as described in [Section 29.2.2.1](#).
- The watchdog timer expires while enabled.
- Fast falling edge (<10 ns) on PSWITCH pin.

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 an unwanted power down due to falling edges. This can also be disabled in register HW_POWER_RESET_PWD_OFF.

29.4.2.1. Powered-Down State

While the chip is powered down, the VDDD and VDDA rail are pulled down to ground. The VDDIO rail is either shorted to ground (Li-Ion) or to the battery (alkaline). 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, Persistent Bits” on page 617](#) for more information about operating a crystal and the RTC in the powered-down state.

To support peripherals that need to be completely powered down in the off state, the alkaline configuration integrates a peripheral power switch. This switch separates the VDDIO 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.

29.4.3. Reset Sequence

[Figure 138](#) shows a flowchart for the power-up, power-down, and reset sequences. A reset event can be triggered by unlocking the HW_POWER_RESET register and setting the HW_CLKCTRL_RESET_DIG bit. This reset affects the digital logic only, although the digital logic also includes most of the registers that control the analog portions of the chip. The persistent bits within the RTC block and the power module control bits are not reset using this method. To reset the analog as well as the digital logic, set the HW_CLKCTRL_RESET_CHIP bit. The DC-DC converter and/or linear regulators continue to maintain the power supply rails during the reset.

29.5. PSWITCH Pin Functions

The PSWITCH pin has several functions whose operation is determined by the STMP3770-based product's hardware and software design.

29.5.1. Power On

When the PSWITCH pin voltage is higher than approximately 0.9Volts for >100 ms, the DC-DC converter begins its startup routine. This is the primary method of starting the system via PSWITCH. All products based on the STMP3770 must have a mechanism of bringing PSWITCH high to power up via a battery.

29.5.2. 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 STMP3770-based systems do not use the PSWITCH fast-falling-edge power-down and include the RC filter to prevent it from occurring accidentally.

29.5.3. Software Functions/Recovery Mode

When the PSWITCH pin voltage is pulled up to at least 0.9 V the lower HW_POWER_STS_PSWITCH 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_POWER_STS_PSWITCH bit is also set. If this bit is set for more

STMP3770

than five seconds during ROM boot, the system executes the SigmaTel USB Firmware Recovery function, as described in [Chapter 31, “Boot Modes” on page 889](#). 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 0.9 V on the PSWITCH pin.

Refer to the SigmaTel STMP3770 reference schematics for example configurations of the PSWITCH pin.

Software-Controlled Resets Normal Power-Up Flow

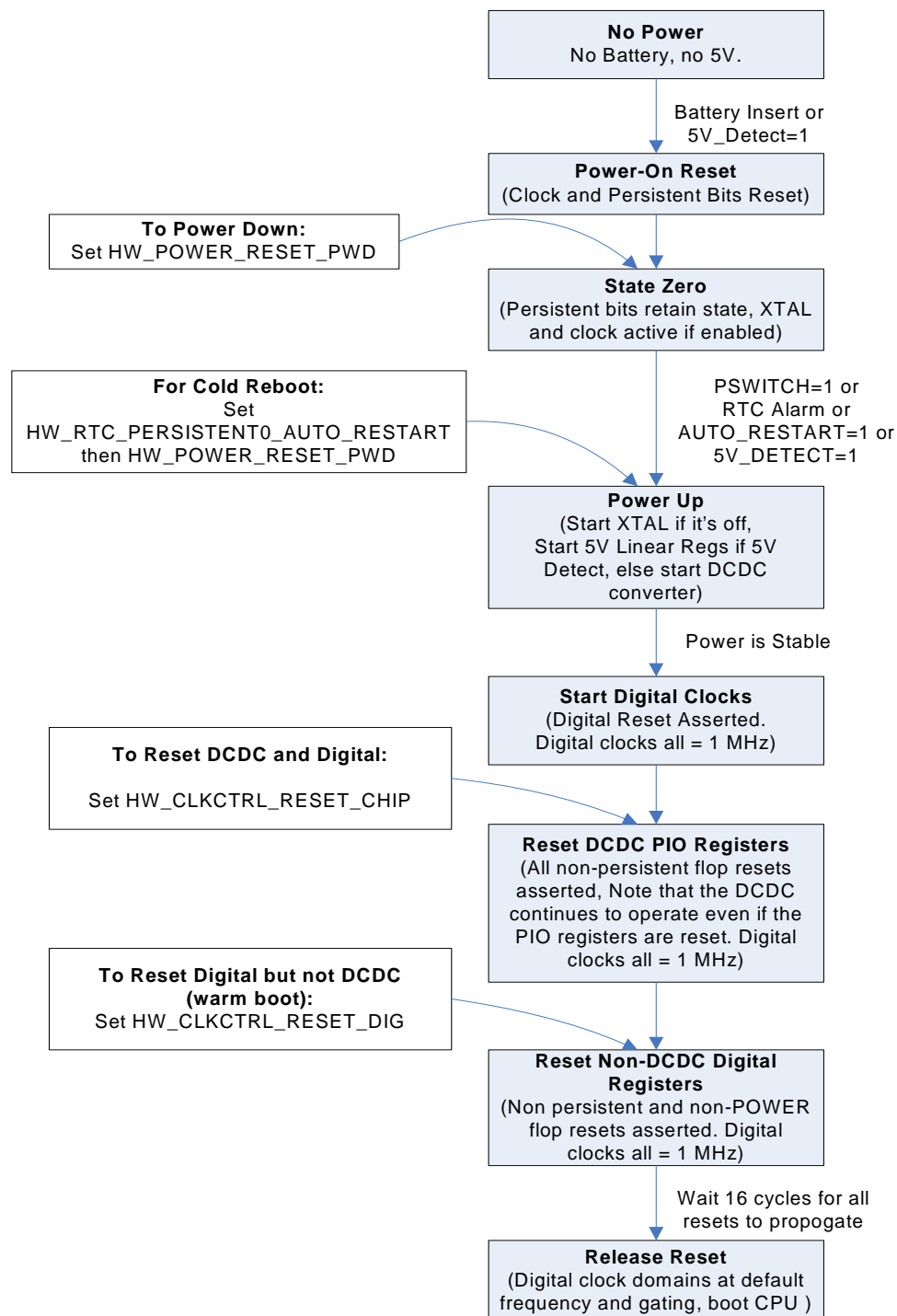


Figure 138. Power-Up, Power-Down and Reset Flow Chart

29.6. 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, 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.

29.7. Battery Charger

Some products in the STMP3770 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_POWER_CHARGE 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 mode 1, then the charge voltage is limited to 1.75 V. If the DC-DC converter is configured for mode 0, then 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_POWER_CHARGE 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 fine) 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, SigmaTel recommends not leaving 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 STMP3770, especially at currents above 400 mA. The dissipated power can be estimated as: $(5V - \text{battery_volt}) * \text{current}$. 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.

29.8. Silicon Speed Sensor

The STMP3770 integrates a silicon speed sensor to measure the performance characteristics of an individual die at its ambient temperature and process parameters. It consists of a ring oscillator and a frequency counter. The ring oscillator runs on the VDDD 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 8-bit counter connected to the ring oscillator that performs the frequency measurement. See the HW_POWER_SPEED 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.

29.9. Interrupts

The power system supports nine CPU interrupt events that are programmable within the HW_POWER_CTRL register. The interrupts are listed in [Table 1101](#).

Table 1101. Power System Interrupts

HW_POWER_CTRL INTERRUPT BIT	DESCRIPTION
VDDA_BO_IRQ	VDDA Brownout
VDDD_BO_IRQ	VDDD Brownout
VDDIO_BO_IRQ	VDDIO Brownout
BATT_BO_IRQ	Battery Brownout
VDD5V_GT_VDDIO_IRQ	$VDD5V > (VDDIO + 0.6)$

Table 1101. Power System Interrupts (Continued)

HW_POWER_CTRL INTERRUPT BIT	DESCRIPTION
DC_OK_IRQ	Voltage Supplies Ok after target voltage change
VBUSVALID_IRQ	VDD5V > Vbusvalid Threshold
LINREG_OK_IRQ	Debug use only
PSWITCH_IRQ	PSWITCH Status

The VDDA_BO_IRQ, VDDD_BO_IRQ, VDDIO_BO_IRQ, and BATT_BO_IRQ each have their own interrupt line back to the interrupt collector. However, the remaining five interrupts—VDD5V_GT_VDDIO_IRQ, DC_OK_IRQ, VBUSVALID_IRQ, LINREG_OK_IRQ, and PSWITCH_IRQ—all share a single interrupt line. In this case, software must read the interrupt status bits to discover which event caused the interrupt.

29.10. DC-DC Converter Efficiency

Figure 139 shows estimates of typical efficiencies of the DC-DC converter under nominal conditions.

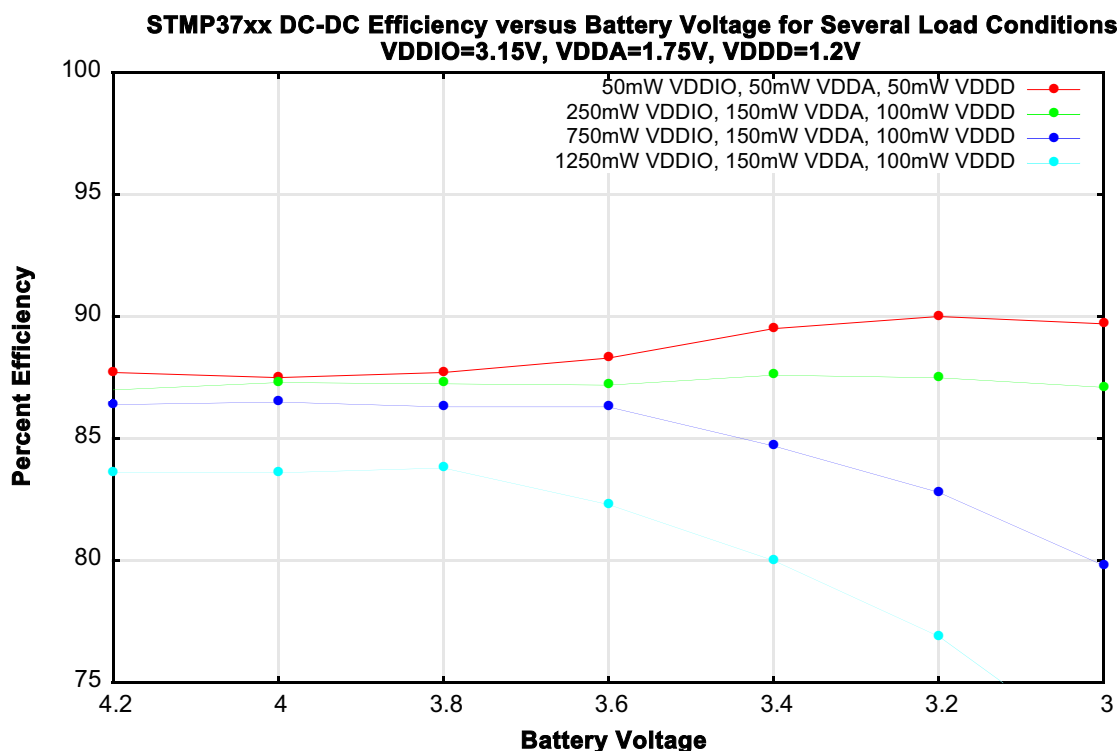


Figure 139. DC-DC Converter Efficiency

Figure 139 shows measurements made with typical devices at room temperature with specific static loads. Therefore, it should not be interpreted as a specification, but as an estimate of typical efficiencies under nominal conditions. Note 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.

Table 1103. HW_POWER_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17	LINREG_OK_IRQ	RW	0x0	Debug use only. Interrupt status for LINREG_OK signal. Interrupt polarity is set using POLARITY_LINREG_OK. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
16	ENIRQ_LINREG_OK	RW	0x0	Debug use only. Enable interrupt for LINREG_OK.
15	DC_OK_IRQ	RW	0x0	Interrupt Status for DC_OK. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
14	ENIRQ_DC_OK	RW	0x0	Enable interrupt for DC_OK.
13	BATT_BO_IRQ	RW	0x0	Interrupt Status for BATT_BO. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
12	ENIRQBATT_BO	RW	0x0	Enable interrupt for battery brownout.
11	VDDIO_BO_IRQ	RW	0x0	Interrupt Status for VDDIO_BO. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
10	ENIRQ_VDDIO_BO	RW	0x0	Enable interrupt for VDDIO brownout.
9	VDDA_BO_IRQ	RW	0x0	Interrupt Status for VDDA_BO. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
8	ENIRQ_VDDA_BO	RW	0x0	Enable interrupt for VDDIO brownout.
7	VDDD_BO_IRQ	RW	0x0	Interrupt Status for VDDD_BO. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
6	ENIRQ_VDDD_BO	RW	0x0	Enable interrupt for VDDD brownout.
5	POLARITY_VBUSVALID	RW	0x1	Set to 1 to check for 5V connected using VBUSVALID status bit. Clear to 0 to check for 5V disconnected.
4	VBUSVALID_IRQ	RW	0x0	Interrupt status for VBUSVALID signal. Interrupt polarity is set using POLARITY_VBUSVALID. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
3	ENIRQ_VBUS_VALID	RW	0x0	Enable interrupt for 5V detection using VBUSVALID.
2	POLARITY_VDD5V_GT_VDDIO	RW	0x1	Set to 1 to check for 5V connected. Clear 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. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
0	ENIRQ_VDD5V_GT_VDDIO	RW	0x0	Enable interrupt for 5V detection.

29.11.2. 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 1104. 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	9	8	7	6	5	4	3	2	1	0										
RSVD																				VBUSVALID_TRSH		RSVD		PWDN_5VBRNOUT		ENABLE_ILIMIT		DCDC_XFER		EN_BATT_PULLDN		VBUSVALID_5VDETECT		VBUSVALID_TO_B		ILIMIT_EQ_ZERO		OTG_PWRUP_CMPS		ENABLE_DCDC	

Table 1105. HW_POWER_5VCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:10	VBUSVALID_TRSH	RW	0x0	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5V and includes hysteresis to minimize the need for software debounce of the detection. 00 = 4.4 V on 5V insertion, 4.21 V on 5V removal 01 = 4.17 V on 5V insertion, 4.0 V on 5V removal 10 = 2.5 V on 5V insertion, 2.45 V on 5V removal 11 = 4.73 V on 5V insertion, 4.48 V on 5V removal
9	RSVD	RO	0x0	Reserved.
8	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 or the VBUSVALID_STATUS status signal based on the VBUSVALID_5VDETECT bit.
7	ENABLE_ILIMIT	RW	0x0	Enable the current limit in the linear regulators. The current limit is active during powerup from 5V and automatically disables before the ROM runs. This limit is needed to meet the USB in-rush current specification of 100 mA + 50 uC. Note that this bit does not affect the battery charger current.

Table 1105. HW_POWER_5VCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	DCDC_XFER	RW	0x0	Enable automatic transition to switching DC-DC converter 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. It is also important to set the EN_BATADJ field.
5	EN_BATT_PULLDN	RW	0x0	Add very small current source (less than 5 μ A) 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.
4	VBUSVALID_5VDETECT	RW	0x0	Select VBUSVALID comparator as detection circuit for 5V in the switching converter. Default is for the switching converter to use the VDD5V_GT_VDDIO status bit to determine the presence of 5V in the system. The VBUSVALID comparator provides a more accurate and adjustable threshold to determine the presence of 5V in the system and is the recommended method of detecting 5V. The OTG_PWRUP_CMPS bit should be set BEFORE this bit is set to allow the comparator time to power up before its output is used.
3	VBUSVALID_TO_B	RW	0x0	This bit muxes the Bvalid comparator to the VBUSVALID comparator and is used for test purposes only.
2	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 0 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.
1	OTG_PWRUP_CMPS	RW	0x0	VBUSVALID comparators are enabled.
0	ENABLE_DCDC	RW	0x0	Enables the switching DC-DC converter when 5V is present. It is important to set all LINREG_OFFSET bits when enabling this functionality. Setting these other bits is important because they will prevent unwanted currents to flow between the linear regulators and the switching converter which could cause unstable supply behavior. Additionally, if the linear regulators are not intended to supply power, then ILIMIT_EQ_ZERO and ENABLE_ILIMIT should be set as well.

29.11.3. DC-DC Minimum Power and Miscellaneous Control Register Description

This register controls options to drop the power used by the switching DC-DC converter. 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 1106. HW_POWER_MINPWR

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																				PWD_BO	USB_I_SUSPEND	ENABLE_OSC	SELECT_OSC	VBG_OFF	DOUBLE_FETS	HALF_FETS	LESSANA_I	PWD_XTAL24	DC_STOPCLK	EN_DC_PFM	DC_HALFCLK

Table 1107. HW POWER MINPWR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11	PWD_BO	RW	0x0	Powers down supply brownout comparators. This should only be used when monitoring supply brownouts is not needed.
10	USB_I_SUSPEND	RW	0x0	Powers down analog comparators used in the power module, including the VDD and VDDA brownout comparators, and changes the reference used by the DC-DC converter to a less accurate, low-power reference. This bit should only be set to reach absolute minimum system power when running from the linear regulators and when supply accuracy and VDDD/VDDA brownout detection are not important.
9	ENABLE_OSC	RW	0x0	Enables the internal oscillator. This oscillator is less accurate, but lower power than the 24-MHz xtal.
8	SELECT_OSC	RW	0x0	Switch internal 24-MHz clock reference to the less accurate internal oscillator. This bit should be set only when the accuracy of the 24-MHz clock is not important. The value of this bit should only be changed when ENABLE_OSC = 1 and PWD_XTAL24 = 0.
7	VBG_OFF	RW	0x0	Powers down the bandgap reference. This should only be used in 5V-powered applications when absolute minimum power is more important than supply voltage accuracy. USB_I_SUSPEND must be set before this bit is set.
6	DOUBLE_FETS	RW	0x0	Approximately doubles the size of power transistors in DC-DC converter. This maybe be useful in high-power conditions.
5	HALF_FETS	RW	0x0	Disable half the power transistors in DC-DC converter. This maybe be useful in low-power conditions when the increased resistance of the power FETs is acceptable.
4	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.

STMP3770

Table 1107. HW_POWER_MINPWR Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3	PWD_XTAL24	RW	0x0	Powers down the 24-MHz oscillator, even when the system is still operating. This can be used with ENABLE_OSC and SELECT_OSC to reduce current in USB suspend and other low-power modes where the accuracy of the clock is not important.
2	DC_STOPCLK	RW	0x0	Stop the clock to internal logic of switching DC-DC converter. This bit will take effect only after the switching FETs are off, due to battery configuration, PFM mode, or internal linear regulator operation.
1	EN_DC_PFM	RW	0x0	Forces DC-DC 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.
0	DC_HALFCLK	RW	0x0	Slow down DC-DC clock from 1.5 MHz to 750 kHz. This may be useful to improve efficiency at light loads or improve EMI radiation, although peak-to-peak voltage on the supplies will increase.

29.11.4. Battery Charge Control Register Description

This register controls the battery charge features for both NiMH slow charge and Li-Ion charge.

HW_POWER_CHARGE	0x80044030
HW_POWER_CHARGE_SET	0x80044034
HW_POWER_CHARGE_CLR	0x80044038
HW_POWER_CHARGE_TOG	0x8004403C

Table 1108. HW_POWER_CHARGE

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
RSVD												ENABLE_FAULT_DETECT	CHRG_STS_OFF	RSVD	USE_EXTERN_R	PWD_BATTCHRG	RSVD				STOP_ILIMIT				RSVD				BATTCHRG_I			

Table 1109. HW_POWER_CHARGE Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	RSVD	RO	0x0	Reserved.
20	ENABLE_FAULT_DETECT	RW	0x0	Enable fault detection in the battery charger. When enabled, this bit will power down the battery charger when the VDD5V voltage falls below the battery voltage. The fault detection is visible via HW_POWER_STS_VDD5V_FAULT.
19	CHRG_STS_OFF	RW	0x0	Setting this bit disables the CHRGSTS status bit. Disabling CHRGSTS should only be done when the switching converter is enabled during battery charge if noise from the switching converter causes CHRGSTS to toggle excessively.
18	RSVD	RW	0x0	Program this field to 0x0.
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	RSVD	RO	0x0	Reserved.
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) SigmaTel recommends setting this value to 10% of the charge current.
7:6	RSVD	RO	0x0	Reserved.
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)

29.11.5. VDDD Supply Targets and Brownouts Control Register Description

This register controls the voltage targets and brownout targets for the VDDD supply generated from the switching DC-DC converter and integrated linear regulators. The brownout comparators default to being enabled.

HW_POWER_VDDDCTRL 0x80044040

Table 1110. HW POWER VDDDCCTRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
ADJTN				RSVD				ALKALINE_CHARGE	DISABLE_STEPPING	LINREG_FROM_BATT		ENABLE_LINREG	DISABLE_FET		RSVD		LINREG_OFFSET		RSVD				BO_OFFSET				RSVD				TRG			

Table 1111. HW_POWER_VDDDCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:28	ADJTN	RW	0x0	Two's complement number that can be used to adjust the duty cycle of VDDD. This is intended for test purposes only.
27:25	RSVD	RO	0x0	Reserved.
24	ALKALINE_CHARGE	RW	0x0	Enables charging of the inductor in boost mode from the VDDD supply instead of ground when the battery is 100 mV above the programmed VDDD voltage. This will improve efficiency in some alkaline applications. When this bit is set, ENABLE_LINREG should be set, LINREG_OFFSET should be set to 0x3, and the battery voltage present in the BATT_VAL field. Also, DISABLE_FET needs to be set or cleared based on the battery voltage.
23	DISABLE_STEPPING	RW	0x0	Disables the default behavior of the voltage stepping algorithm when the TRG field is updated. By default, the control loop steps sequentially through 25 mV steps on both target and brownout voltage to minimize transients during TRG adjustments. When this bit is set, the entire target voltage and brownout adjustment takes place at one time, which will speed transitions, but will result in increased supply transients as well as possible brownouts when the new (TRG - BO_OFFSET) >= old TRG. Thus, it is recommended to change TRG by less than the (BO_OFFSET-2) value when using DISABLE_STEPPING. This bit should be set high when powering VDDD from the integrated linear regulators.
22	LINREG_FROM_BATT	RW	0x0	Switches the source of the VDDD regulator to the battery from VDDA. The default source is the VDDA supply. This bit should only be set in alkaline-powered applications when the battery is greater than 100 mV above VDDD. If this bit is set, ENABLE_LINREG should also be set.

Table 1111. HW_POWER_VDDDCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21	ENABLE_LINREG	RW	0x1	Enables the VDDD linear regulator converter when the switching converter is active. By default, the linear regulator is not active when the switching converter is active.
20	DISABLE_FET	RW	0x1	Disables the VDDD switching converter output. When using alkaline configurations, this bit should only be cleared when the battery voltage is below the programmed VDDD voltage + 100 mV.
19:18	RSVD	RO	0x0	Reserved.
17:16	LINREG_OFFSET	RW	0x1	Number of 25-mV steps between linear regulator output voltage and switching converter target. 00b = 0 steps—Recommended when powering VDDD only from linear regulator and ENABLE_DCDC=DCDC_XFER=0. SigmaTel also recommends setting DISABLE_STEPPING when powering VDDD from the linear regulators. 01b = 1 step above, default. 1Xb = 1 step below—Important when powering VDDD from DC-DC converter and linear regulator simultaneously, including when ALKALINE_CHARGE is set.
15:11	RSVD	RO	0x0	Reserved.
10:8	BO_OFFSET	RW	0x7	Brownout voltage offset in 25-mV steps below the TRG value. Note that the hardware only supports brownout voltages between 0.8 V and 1.475 V, and values outside this range should not be programmed. The brownout trip voltage will adjust as the target voltage changes.
7:5	RSVD	RO	0x0	Reserved.
4:0	TRG	RW	0x10	Voltage level of the VDDD supply. The step size of this field is 25 mV. 0x00 = 0.8 V, 0x1F = 1.575 V, and the reset value = 1.2 V. This field should not be set above 1.45V as this may cause damage to the device. It is recommended to limit changes in TRG to eight or less steps before waiting for DC_OK to be high as this will minimize supply transients. It is also recommended to set DISABLE_STEPPING when powering VDDD from the integrated linear regulators. Setting DISABLE_STEPPING does set additional restrictions on TRG adjustments.

29.11.6. VDDA Supply Targets and Brownouts Control Register Description

This register controls the voltage targets and brownout targets for the VDDA supply generated from the switching DC-DC converter and integrated linear regulators. The brownout comparators default to being enabled.

HW_POWER_VDDACTRL 0x80044050

Table 1115. HW_POWER_VDDIOCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
15	DISABLE_STEPPING	RW	0x0	Disables the default behavior of the voltage stepping algorithm when the TRG field is updated. By default, the control loop steps sequentially through 25mV steps on both target and brownout voltage to minimize transients during TRG adjustments. When this bit is set, the entire target voltage and brownout adjustment takes place at one time, which will speed transitions, but will result in increased supply transients as well as possible brownouts when the new (TRG - BO_OFFSET) >= old TRG. Thus, it is recommended to change TRG by less than the (BO_OFFSET-2) value when using DISABLE_STEPPING. This bit should be set high when powering VDDIO from the integrated linear regulators.
14	DISABLE_FET	RW	0x0	Disables the VDDIO switching converter output. The switching converter is enabled by default when the battery is not present or the ENABLE_DCDC is set.
13:12	LINREG_OFFSET	RW	0x1	Number of 25-mV steps between linear regulator output voltage and switching converter target. 00b = 0 steps—Recommended when powering VDDIO from linear regulator and ENABLE_DCDC=DCDC_XFER=0. SigmaTel also recommends setting DISABLE_STEPPING when powering VDDIO from the linear regulators. 01b = 1 above, default. 1Xb = 1 step below—Important when powering VDDIO from DC-DC converter and linear regulator simultaneously.
11	RSVD	RO	0x0	Reserved.
10:8	BO_OFFSET	RW	0x7	Brownout voltage offset in 25-mV steps below the TRG value. Note that the hardware only supports brownout voltages between 2.7 V and 3.475 V, and values outside this range should not be programmed. The brownout trip voltage will adjust as the target voltage changes.
7:5	RSVD	RO	0x0	Reserved.
4:0	TRG	RW	0x0C	Voltage level of the VDDIO supply. The step size of this field is 25 mV. 0x00 = 2.8 V, 0x1F = 3.575 V, and the reset value = 3.1 V. It is recommended to limit changes in TRG to eight or less steps before waiting for DC_OK to be high as this will minimize supply transients. It is also recommended to set DISABLE_STEPPING when powering VDDIO from the integrated linear regulators. Setting DISABLE_STEPPING does set additional restrictions on TRG adjustments.

29.11.8. DC-DC Multi-Output Converter Modes Control Register Description

This register contains controls that may need to be adjusted to optimize DC-DC converter performance using the battery voltage information.

HW_POWER_DCFUNCV

0x80044070

Table 1116. HW_POWER_DCFUNCV

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4
RSVD				VDDD								RSVD				VDDIO											

Table 1117. HW_POWER_DCFUNCV Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x0	Reserved.
25:16	VDDD	RW	0x000	This field should be initially programmed before EN_BATADJ and should be updated whenever the target value for either VDDD or VDDA is changed. For Li-Ion, write: $(VDDA - VDDD) / 6.25e-03$ For alkaline batteries, write: $(VDDD * VDDA) / ((VDDA - VDDD) * 6.25e-03)$ The values of VDDD and VDDA are in volts, and the final result can be rounded. In rare cases for alkaline batteries, the desired programmed value may exceed 0x3FF. If this situation occurs, use 0x3FF and program ADJTN in VDDDCTRL register after consulting SigmaTel.
15:10	RSVD	RO	0x00	Reserved.
9:0	VDDIO	RW	0x000	This field should be initially programmed before EN_BATADJ and should be updated whenever the target value for either VDDIO or VDDA is changed. For Li-Ion batteries, write $(VDDIO - VDDA) / 6.25e-03$, and for alkaline batteries, write $(VDDIO * VDDA) / ((VDDIO - VDDA) * 6.25e-03)$. The values of VDDIO and VDDD are in volts, and the final result can be rounded.

STMP3770



29.11.9. DC-DC Miscellaneous Register Description

This register contains controls that may need to be adjusted to optimize DC-DC converter performance using the battery voltage information.

HW POWER MISC

0x80044080

Table 1118. HW_POWER_MISC

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																											FREQSEL	DELAY_TIMING	TEST	SEL_PL1CLK	PERIPHERALSWOFF

Table 1119. HW_POWER_MISC Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:6	RSVD	RO	0x0	Reserved.
5:4	FREQSEL	RW	0x0	This register will select the PLL-based frequency that the DC-DC uses when SEL_PLLCLK is set high. The decode is as follows: 0x0 = Reserved. 0x1 = 20 MHz. 0x2 = 24 MHz. 0x3 = 19.2 MHz.
3	DELAY_TIMING	RW	0x0	This bit delays the timing of the output FETs in the switching DC-DC converter. This may provide improved ground noise performance in high-power applications.
2	TEST	RW	0x0	Reserved. Do not set.
1	SEL_PLLCLK	RW	0x0	This bit selects the source of the clock used for the DC-DC converter. The default is to use the 24-MHz clock. Setting this bit selects the PLL clock as a clock source for the DC-DC converter. It is required to program FREQSEL before setting this bit.
0	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.

29.11.10.DC-DC Duty Cycle Limits Control Register Description

This register defines the upper and lower duty cycle limits of DC-DC. These values depend on details of switching converter implementation and should not be changed without guidance from SigmaTel.

HW POWER DCLIMITS

0x80044090

STMP3770

Table 1123. HW_POWER_LOOPCTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:21	RSVD	RO	0x0	Reserved.
20	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.
19	HYST_SIGN	RW	0x0	Invert the sign of the hysteresis in DC-DC analog comparators. This bit should set when using PFM mode.
18	EN_CM_HYST	RW	0x0	Enable hysteresis in switching converter common mode analog comparator. This feature will improve transient supply ripple and efficiency.
17	EN_DF_HYST	RW	0x0	Enable hysteresis in switching converter differential mode analog comparators. This feature will improve transient supply ripple and efficiency.
16	CM_HYST_THRESH	RW	0x0	Increase the threshold detection for common mode analog comparator.
15	DF_HYST_THRESH	RW	0x0	Increase the threshold detection for common mode analog comparator.
14	RCSCALE_THRESH	RW	0x0	Increase the threshold detection for RC scale circuit.
13:12	EN_RCSCALE	RW	0x0	Enable analog circuit of DC-DC converter to respond faster under transient load conditions. 00 = Disabled. 01 = 2X increase. 10 = 4X increase. 11 = 8X increase.
11	RSVD	RO	0x0	Reserved.
10:8	DC_FF	RW	0x0	Two's complement feed forward step in duty cycle in the switching DC-DC converter. 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	DC_R	RW	0x2	Magnitude of proportional control parameter in the switching DC-DC converter control loop.
3:2	RSVD	RO	0x0	Reserved.
1:0	DC_C	RW	0x1	Ratio of integral control parameter to proportional control parameter in the switching DC-DC converter. Can be used to optimize efficiency and loop response. 00 = Maximum. 01 = Decrease ratio 2X. 10 = Decrease ratio 4X. 11 = Lowest ratio.

29.11.12.Power Subsystem Status Register Description

This register contains status information for the battery charger, DC-DC converter and USB/OTG connections.

HW POWER STS

0x800440B0

Table 1124. HW POWER STS

BATT_CHRG_PRESENT	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0																				
	RSVD												PSWITCH				AVALID_STATUS		BVALID_STATUS		VBUSVALID_STATUS		SESEND_STATUS		MODE		BATT_BO		VDD5V_FAULT		CHRGSTS		LINREG_OK		DC_OK		VDDIO_BO		VDDA_BO		VDDD_BO		VDD5V_GT_VDDIO		AVALID		BVALID		VBUSVALID		SESEND	

Table 1125. 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.
30:20	RSVD	RO	0x0	Reserved.
19:18	PSWITCH	RO	0x0	These read-only bits reflect the current state of the PSWITCH comparators. The LSB is high when voltage on the PSWITCH pin is above 0.8 V, and the MSB is high when the voltage on the PSWITCH pin is above 1.6 V.
17	AVALID_STATUS	RO	0x0	Indicates VBus is valid for a A-peripheral. This bit is a read-only version of the state of the analog signal. It cannot be overwritten by software like the AVALID bit below.
16	BVALID_STATUS	RO	0x0	Indicates VBus is valid for a B-peripheral. This bit is a read-only version of the state of the analog signal. It cannot be overwritten by software like the BVALID bit below.
15	VBUSVALID_STATUS	RO	0x0	VBus valid for USB OTG. This bit is a read-only version of the state of the analog signal. It cannot be overwritten by software like the VBUSVALID bit below.
14	SESSEND_STATUS	RO	0x0	Session End for USB OTG. This bit is a read-only version of the state of the analog signal. It cannot be overwritten by software like the SESSEND bit below.
13	MODE	RO	0x0	Battery configuration of system. 0 = Li-Ion. 1 = Single AA or AAA.
12	BATT_BO	RO	0x0	Output of battery brownout comparator.

STMP3770



Table 1125. HW_POWER_STS Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11	VDD5V_FAULT	RO	0x0	Battery charging fault status. If the battery charger is not powered down, the bit is high when the 5V supply falls below the battery voltage. If the charger is powered down, the bit asserts high when 5V falls to below roughly VDDIO/2. If the charger is not powered down, the bit is sticky and remains set until the PWD_BATTCHRG bit is cycled. Otherwise, the bit is cleared when 5V is restored.
10	CHRGSTS	RO	0x0	Battery charging status. High during Li-Ion battery charge until the charging current falls below the STOP_ILIMIT threshold.
9	LINREG_OK	RO	0x0	Debug use only.
8	DC_OK	RO	0x0	High when switching DC-DC converter control loop has stabilized after a voltage target change. When linear regulators are active, this bit goes high when the actual voltage is above the target voltage. Therefore, DC_OK will go high when changing a linear regulator output to a lower value before the actual voltage decreases to the new target value.
7	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 HW_POWER_MINPWR register.
6	VDDA_BO	RO	0x0	Output of VDDA brownout comparator. High when a brownout is detected. It is not possible to power-down this comparator.
5	VDDD_BO	RO	0x0	Output of VDDD brownout comparator. High when a brownout is detected. It is not possible to power-down this comparator.
4	VDD5V_GT_VDDIO	RO	0x0	Indicates the voltage on the VDD5V pin is higher than VDDIO by a V_t voltage, nominally 500 mV.
3	AVALID	RW	0x0	Indicates VBus is above the VA_SESS_VLD threshold. High if VBus greater than 2.0. Low if VBus less than 0.8. Otherwise unknown.
2	BVALID	RW	0x0	Indicates VBus is above the VB_SESS_VLD threshold. High if VBus greater than 4.0. Low if VBus less than 0.8. Otherwise unknown.
1	VBUSVALID	RW	0x0	Accurate detection of the presence of 5V power. This can be used for detection of 5V in all modes of operation including USB OTG. See HW_POWER_5VCTRL to enable and set threshold for comparison.
0	SESSEND	RW	0x0	Indicates VBus is below the VB_SESS_END threshold, i.e., 0 if VBus is greater than 0.8 V, 1 if VBus is less than 0.2 V, otherwise unknown. See HW_POWER_5VCTRL to enable comparators.

29.11.13. Transistor Speed Control and Status Register Description

This register contains the setup and controls needed to measure silicon speed.

HW_POWER_SPEED 0x800440C0
 HW_POWER_SPEED_SET 0x800440C4
 HW_POWER_SPEED_CLR 0x800440C8
 HW_POWER_SPEED_TOG 0x800440CC

Table 1126. HW_POWER_SPEED

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2
RSVD								STATUS								RSVD								CTRL					

Table 1127. HW_POWER_SPEED Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	RSVD	RO	0x0	Reserved.
23:16	STATUS	RO	0x0	Result from the speed sensor. This result is only valid when SPEEDCTRL=0b11; otherwise this field contains debug information from the switching DC-DC converter.
15:2	RSVD	RO	0x0	Reserved.
1:0	CTRL	RW	0x0	Speed Control. 00 = Speed sensor off. 01 = Speed sensor enabled. 11 = Enable speed sensor measurement. Every time a measurement is taken, the sequence of 00, 01, 11 must be repeated. This sequence should proceed no faster than 1.5 MHz to ensure proper operation.

Table 1131. HW_POWER_RESET Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
1	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.
0	PWD	RW	0x0	Powers down the chip.

29.11.16. Power Module Debug Register Description

HW_POWER_DEBUG	0x800440F0
HW_POWER_DEBUG_SET	0x800440F4
HW_POWER_DEBUG_CLR	0x800440F8
HW_POWER_DEBUG_TOG	0x800440FC

Table 1132. HW POWER DEBUG

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																												VBUSVALIDPILOCK	AVALIDPILOCK	BVALIDPILOCK	SESENDPILOCK

Table 1133. HW_POWER_DEBUG Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:4	RSVD	RO	0x0	Reserved.
3	VBUSVALIDPIOLOCK	RW	0x0	Reserved.
2	AVALIDPIOLOCK	RW	0x0	Reserved.
1	BVALIDPIOLOCK	RW	0x0	Reserved.
0	SESENDPIOLOCK	RW	0x0	Reserved.

29.11.17. Power Module Special Register Description

HW_POWER_SPECIAL	0x80044100
HW_POWER_SPECIAL_SET	0x80044104
HW_POWER_SPECIAL_CLR	0x80044108
HW_POWER_SPECIAL_TOG	0x8004410C

Table 1134. HW POWER SPECIAL

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0		
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
TEST																															

Table 1135. HW POWER SPECIAL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	TEST	RW	0x0	Reserved.

29.11.18. Power Module Version Register Description

This register indicates the version of the block.

```
HW POWER VERSION      0x80044110
```

Table 1136. HW POWER VERSION

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
MAJOR								MINOR								STEP															

Table 1137. HW_POWER_VERSION Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:24	MAJOR	RO	0x02	Fixed read-only value reflecting the MAJOR field of the RTL version.
23:16	MINOR	RO	0x00	Fixed read-only value reflecting the MINOR field of the RTL version.
15:0	STEP	RO	0x0000	Fixed read-only value reflecting the stepping of the RTL version.

POWER Block v2.0

STMP3770



30. SERIAL JTAG (SJTAG)

This chapter describes the one-wire serial JTAG (SJTAG) function included on the STMP3770 and how to use it. There are no registers in this module.

30.1. Overview

The STMP3770 provides a one-wire serial JTAG (SJTAG) interface to connect to various external JTAG debugger dongles through a SigmaTel-defined FPGA/CPLD. SJTAG supports the Green Hills Slingshot and ETM probe debugger dongles, as well as those made by ARM, Abatron, and Lauterbach.

The SJTAG block provides the following functions.

- Maps one-wire protocol to six-wire JTAG interface on the ARM926 core.
- Detects presence of external debugger when it is connected to the one-wire SJTAG pin (DEBUG) and it issues clock or JTAG reset commands.
- Signals JTAG presence to external FPGA/CPLD translator.
- Detects glitches and false JTAG clocks to improve noise immunity.

The one-wire SJTAG interface is illustrated in [Figure 140](#).

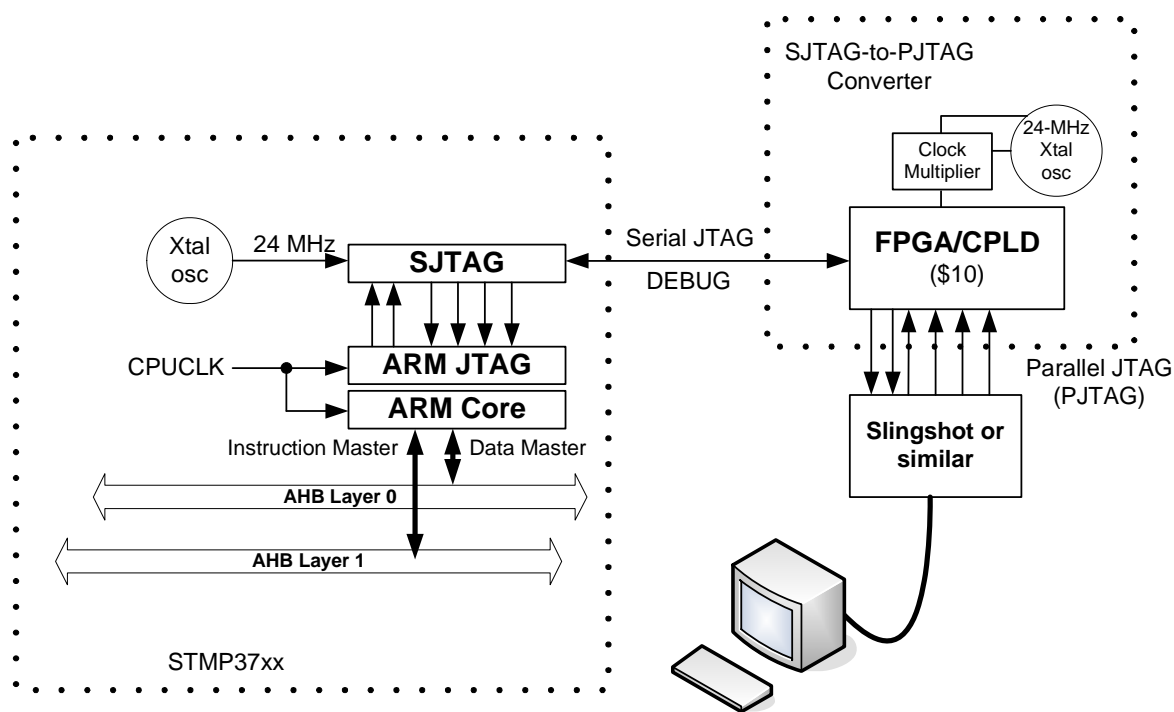


Figure 140. Serial JTAG (SJTAG) Block Diagram

30.2. Operation

The architecture of the one-wire serial JTAG interface depends on the FPGA/CPLD to always be present and for it to do all of the “heavy lifting” associated with synchronizing data back and forth across the interface. To that end, the FPGA/CPLD is programmed to use its digital clock modules to generate an 8x oversample clock to interface with the 24-MHz on-chip crystal oscillator circuits. [Figure 141](#) defines the clock relationships.

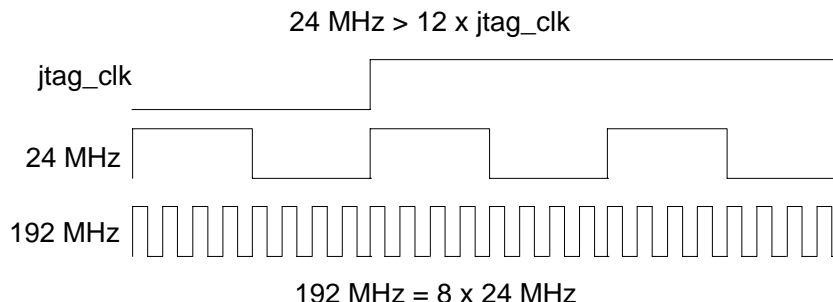


Figure 141. SJTAG Clock Relationships

The following high-level steps describe using the FPGA/CPLD and the debugger with the STMP3770 SJTAG interface:

- The FPGA/CPLD synchronizes the debugger’s JTAG clock into its 192-MHz domain and edge-detects it.
- The rising edge of the debugger’s clock triggers an asynchronous start phase in which the FPGA/CPLD drives the DEBUG pin high for one FPGA/CPLD clock period.
- The DEBUG pin is pulled lightly high until the SJTAG block on the chip recognizes the start condition on the wire.
- The timing mark phase is entered at that point, and the chip pulls the DEBUG line back low.
- This falling edge is detected by the oversample clock in the FPGA/CPLD, and it uses this timing mark to drive all subsequent transfers to the chip.
- The 8x oversample clock then works on pseudo-synchronous timing points derived by counting out the 8x clocks.

The following sections describe these phases illustrated in [Figure 142](#) in more detail.

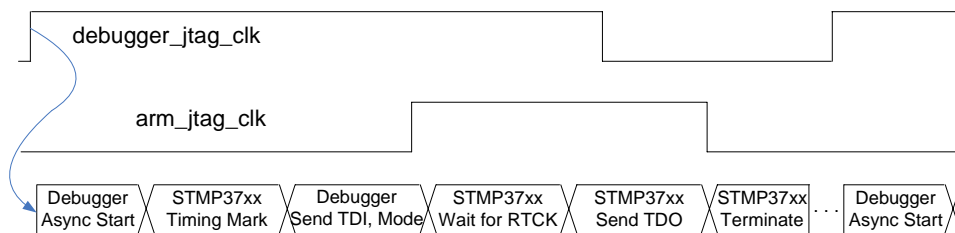


Figure 142. SJTAG Phases of Operation for One JTAG Clock

30.2.1. *Debugger Async Start Phase*

- This phase begins when a rising edge is detected on the JTAG clock signal coming from the debugger.
- This phase lasts a fixed number of 192-MHz clock periods. That is, the DEBUG pin is driven high for a period corresponding to half a 24-MHz clock, i.e., 20.8 ns. This drive is completely asynchronous to the on-chip 24-MHz clock, i.e., the relationship is unknown at this point.
- The soft pullup is turned on by the FPGA/CPLD and left on as the phase ends.
- This phase ends when the strong driver is turned off and the timing mark phase is entered.

30.2.2. *STMP3770 Timing Mark Phase*

- This phase is entered when the FPGA/CPLD releases its strong driver, i.e., when it three-states the driver.
- The SJTAG controller on the chip detects the rising edge on the DEBUG pin and synchronizes it.
- This action starts a shift register timing chain that runs through this and the next phase.
- When the synchronized edge is recognized by the on-chip SJTAG controller, it pulls the DEBUG line back down clock to Q to pad after the rising edge of its 24-MHz clock. This is the critical timing mark that is detected in the FPGA/CPLD and used to time data in next phase.
- The timing mark phase ends when the on-chip SJTAG stops driving the serial JTAG wire low for one cycle.

30.2.3. *Debugger Send TDI, Mode Phase*

- During the first 24-MHz clock period of this phase, the FPGA/CPLD sends a one clock-wide signal that either tells the on-chip SJTAG that it is present and a JTAG clock begins, or it tells the on-chip SJTAG to do a JTAG reset operation to the ARM JTAG TAP controller.
- If a noise glitch falsely triggered the ASYNC Start Phase, then the on-chip SJTAG will treat it as a TAP controller reset in most cases.
- If the debugger is performing a JTAG clock cycle operation then, it next sends the state of the debugger TDI and MODE pins sequentially on the wire, i.e., one in each of the following two 24-MHz clocks. Notice that for this phase, the FPGA/CPLD knows the correct timing to change these three data elements on

the wire because of the timing information it learned from the Timing Mark Phase.

- This phase ends after the FPGA/CPLD drives the serial wire low on the fourth 24-MHz clock of this phase.

To review, the first data element sent is the signal that distinguishes clock cycles from TAP reset cycles. The next two bits sent are the JTAG MODE and TDI bits from the debugger. Finally, the line is driven low and pulled down for half a 24-MHz clock and the driver is turned off and the pulldown left on. This phase ends when the half-clock pulldown is complete. The rising edge of the JTAG clock is sent to the ARM TAP controller during this phase.

30.2.4. STMP3770 Wait For Return Clock Phase

During this phase:

- The on-chip SJTAG controller waits for the ARM TAP controller to send back the return clock. This is an asynchronous event for both the on-chip TAP controller and the FPGA/CPLD controller.
- The on-chip controller drives the serial wire high for one 24-MHz clock period to tell the FPGA/CPLD that the variable length wait for return clock period is complete.
- This phase ends when the on-chip SJTAG detects the return clock going high and drives the serial high for one 24-MHz clock.

30.2.5. STMP3770 Sends TDO and Return Clock Timing Phase

During this phase:

- The on-chip SJTAG controller sends the value of the ARM TAP controller's TDO signal on the wire for one full 24-MHz period that begins on the rising edge of the on-chip 24-MHz clock.
- This phase ends when the TDO value has been sent.

30.2.6. STMP3770 Terminate Phase

The primary purpose of this phase to leave the SJTAG serial wire in the low state.

- The on-chip SJTAG controller accomplishes this by driving it low for half a 24-MHz clock, releasing it at the falling edge of its internal 24-MHz clock.
- This allows the next JTAG cycle to be started by the FPGA/CPLD around $\frac{3}{4}$ of a 24-MHz clock after this phase is entered.
- When this phase ends, the on-chip SJTAG controller resets its "Active" flip-flop and returns to its idle state in both its timing chain and its state machine.

The on-chip SJTAG always drives out the serial wire at the rising edge of the 24-MHz clock. It may drive for one 24-MHz clock cycle (return clock and TDO) or half cycle (terminate phase).

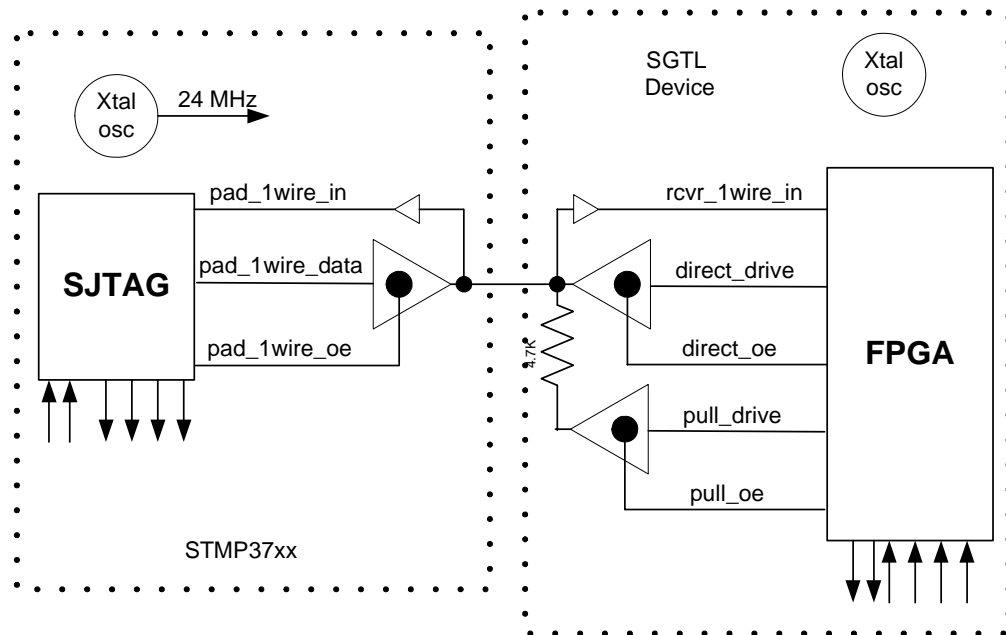


Figure 143. SJTAG Drivers

30.2.7. SJTAG External Pin

The SJTAG interface uses a single bidirectional interface pin (DEBUG) running at VDDIO to communicate with external debuggers. The DEBUG pad itself is wired as a conventional 8-mA 3.3-V driver. The DEBUG pin is completely dedicated to this one function. The signal on this interface is actively pulled up or down by the SOC as well as by the external debugger. However, the SOC will never drive this interface until it is first driven high by the external debugger.

The external JTAG debugger interface circuit includes a switched 4.7 Kohm pull-down resistor on its board. The DEBUG pin has a Schmitt trigger.

If the DEBUG pin is unused, SigmaTel recommends pulling the DEBUG pin to ground through a 100K resistor. It is also possible to short the DEBUG pin to ground directly, but doing this will prohibit debugging on a production player.

30.2.8. Selecting Serial JTAG or Six-Wire JTAG Mode

The HW_DIGCTL_CTRL_USE_SERIAL_JTAG bit in the digital control block selects whether the serial JTAG interface or the alternative six-wire parallel JTAG interface is used.

- When this bit is cleared to 0, parallel six-wire JTAG is enabled and is mapped to a collection of module pins that must be enabled by programming their PINMUXSEL bits in the PINCTRL block.
- When this bit is set to 1, serial JTAG is enabled and uses the dedicated DEBUG pin.

The ROM bootcode writes this field prior to enabling JTAG, selecting which type of JTAG pin signaling to use.

STMP3770



31. BOOT MODES

This chapter describes the boot modes implemented on the STMP3770.

31.1. Boot Modes

[Table 1138](#) lists all of the boot modes supported by the STMP3770 ROM. The boot mode can be selected either through external resistors (similar to STMP36xx behavior) or via OTP eFuse bit programming.

Table 1138. Boot Modes

PORT	BOOT MODE
USB	Encrypted/unencrypted USB slave boot mode.
I ² C	Encrypted/unencrypted I ² C master—Boots from 1.8-V and 3.3-V EEPROM.
SPI	Encrypted/unencrypted SPI master from SSP1—Boots from 1.8-V and 3.3-V flash memory.
SPI	Encrypted/unencrypted SPI master from SSP2—Boots from 1.8-V and 3.3-V flash and EEPROM.
SSP1	Encrypted/unencrypted SD/MMC master from SSP1—Boots from 1.8-V and 3.3-V 1-bit, 4-bit, and 8-bit SD/MMC cards.
SSP2	Encrypted/unencrypted SD/MMC master from SSP2—Boots from 1.8-V and 3.3-V 1-bit, 4-bit, and 8-bit SD/MMC cards.
GPMI	Encrypted/unencrypted NAND 3.3-V and 1.8-V 8-bit wide and ECC4 and ECC8.
JTAG_WAIT	Unencrypted startup —Waits for JTAG debugger connection.

31.1.1. Boot Pins Definition and Mode Selection

Boot pins are located on LCD_RS and LCD_DATA[5:0]. To enable boot mode selection from the LCD data pins, pull up LCD_RS. The ROM probes the LCD_RS pin and then, if valid, decodes the boot mode vector from the pins LCD_DATA[5:0].

If LCD_RS is pulled down, then the boot mode is determined by OTP eFuse bits, as defined in [Table 1141](#).

[Table 1139](#) shows the boot pins, and [Table 1140](#) lists the boot mode vectors.

Table 1139. Boot Pins

PIN NAME	BOOT FUNCTION	BIT NAME
LCD_RS	Determines if there is a need to examine the other boot pins.	
LCD_DATA[5]	Test Boot Mode Bit 1/ETM Enable	TBM1
LCD_DATA[4]	Test Boot Mode Bit 0/Voltage Selector	TBM0
LCD_DATA[3]	Boot Mode Bit 3	BM3
LCD_DATA[2]	Boot Mode Bit 2	BM2
LCD_DATA[1]	Boot Mode Bit 1	BM1

Table 1139. Boot Pins (Continued)

PIN NAME	BOOT FUNCTION	BIT NAME
LCD_DATA[0]	Boot Mode Bit 0	BM0

These pads are powered during the initial startup 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 GPIOs unless BM3:0=0xF. The ETM sets drive strength to 8mA on the ETM pins.

31.1.2. Boot Mode Selection Map

Table 1140. Boot Mode Selection Map

ETM ENABLE/ LCD_ DATA[5]	VOLTAGE SELECTOR/ LCD_ DATA[4]	BM3/ LCD_ DATA[3]	BM2/ LCD_ DATA[2]	BM1/ LCD_ DATA[1]	BM0/ LCD_ DATA[0]	PORT	BOOT MODE
0/1	x	0	0	0	0	USB	USB (unencrypted vs. encrypted is under OTP control)
0/1	0	0	0	0	1	I ² C	I ² C master, 3.3-V
0/1	1	0	0	0	1	I ² C	I ² C master, 1.8-V
0/1	0	0	0	1	0	SPI	SPI master SSP1 boot from flash, 3.3-V
0/1	1	0	0	1	0	SPI	SPI master SSP1 boot from flash, 1.8-V
0/1	0	0	0	1	1	SPI	SPI master SSP2 boot from flash, 3.3-V
0/1	1	0	0	1	1	SPI	SPI master SSP2 boot from flash, 1.8-V
0/1	0	0	1	0	0	GPMI	NAND 8-bit, ECC4, 3.3-V
0/1	1	0	1	0	0	GPMI	NAND 8-bit, ECC4, 1.8-V
0/1	x	0	1	1	0	JTAG_WAIT	Startup waits for JTAG debugger connection
x	x	0	1	1	1	–	Reserved
0/1	0	1	0	0	0	SPI	SPI master SSP2 boot from EEPROM, 3.3-V
0/1	1	1	0	0	0	SPI	SPI master SSP2 Boot from EEPROM, 1.8-V
0/1	0	1	0	0	1	SSP1	SD/MMC master on SSP1, 3.3-V
0/1	1	1	0	0	1	SSP1	SD/MMC master on SSP1, 1.8-V
0/1	0	1	0	1	0	SSP2	SD/MMC master on SSP2, 3.3-V
0/1	1	1	0	1	0	SSP2	SD/MMC master on SSP2, 1.8-V
x	x	1	0	1	1	–	Reserved
0/1	0	1	1	0	0	GPMI	NAND 8-bit, ECC8, 3.3-V
0/1	1	1	1	0	0	GPMI	NAND 8-bit, ECC8, 1.8-V
x	x	1	1	1	0	–	Reserved
x	x	1	1	1	1	–	Reserved

31.2. OTP eFuse and Persistent Bit Definitions

31.2.1. OTP eFuse

The STMP3770 contains a 1-Kbit array of OTP eFuse bits, some of which are used by the ROM. The bits listed in [Table 1141–Table 1143](#) can be configured by customers and are typically programmed on the customer's board assembly line. For more information about the OTP bits, see [Chapter 8, “On-Chip OTP \(OCOTP\) Controller”](#) on page 169.

Table 1141. General ROM Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM0:0x8002C1A0:31:29	Undefined
HW_OCOTP_ROM0:0x8002C1A0:28	TBM0
HW_OCOTP_ROM0:0x8002C1A0:27	BM3
HW_OCOTP_ROM0:0x8002C1A0:26	BM2
HW_OCOTP_ROM0:0x8002C1A0:25	BM1
HW_OCOTP_ROM0:0x8002C1A0:24	BM0
HW_OCOTP_ROM0:0x8002C1A0:21:20	POWER_GATE_GPIO—SD/MMC card power gate GPIO pin select. 00 = PWM3 01 = PWM4 10 = ROTARYA 11 = None
HW_OCOTP_ROM0:0x8002C1A0:19:14	POWER_UP_DELAY—SD/MMC card power up delay required after enabling GPIO power gate. 000000 = 20 ms (default) 000001 = 10 ms 000010 = 20 ms 111111 = 630 ms
HW_OCOTP_ROM0:0x8002C1A0:13:12	SD_BUS_WIDTH—SD/MMC card bus width. 00 = 4-bit 01 = 1-bit 10 = 8-bit 11 = Reserved
HW_OCOTP_ROM0:0x8002C1A0:11:8	Index to SSP clock speed. By default 0x0, the clock speed is set to 12 MHz. The value of the index will modify the SPI clock speed accordingly.
HW_OCOTP_ROM0:0x8002C1A0:6	DISABLE_SPI_NOR_FAST_READ—Blow to disable SPI NOR fast reads, which are used by default. Some SPI NORs do not support this functionality.
HW_OCOTP_ROM0:0x8002C1A0:5	ENABLE_USB_BOOT_SERIAL_NUMBER—If set, the device serial number is reported to the host during USB boot mode initialization, else no serial number is reported.
HW_OCOTP_ROM0:0x8002C1A0:4	ENABLE_UNENCRYPTED_BOOT—If clear, allows only booting of encrypted images. If set, both encrypted/unencrypted images are valid.

Table 1141. General ROM Bits (Continued)

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM0:0x8002C1A0:3	SD_MBR_BOOT—Set to enable SD Master Boot Record (MBR) mode. The SD/MMC card should have a valid MBR to boot successfully in this mode. If this bit is not set, ROM will try to boot in default mode, BCB (Boot Control Block).
HW_OCOTP_ROM0:0x8002C1A0:2	DISABLE_RECOVERY_MODE—If set, does not allow booting in recovery mode.

Table 1142. NAND-Related Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM1:0x8002C1B0:12	USE_ALTERNATE_CE—If set, the NAND driver uses the NAND alternate chip selects.
HW_OCOTP_ROM1:0x8002C1B0:11:8	BOOT_SEARCH_COUNT—Number of 64-page blocks skipped by the NAND driver when searching for information saved into the NAND (see Section 31.9 for details).
HW_OCOTP_ROM1:0x8002C1B0:2:0	NUMBER_OF_NANDS—Indicates the number of external NANDs. A 0 value means that the NAND driver will scan the chip selects to dynamically find the correct number of NANDs.

Table 1143. USB-Related Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM2:0x8002C1C0:31:16	USB_VID—Vendor ID used in recovery mode. If the field is 0, SigmaTel vendor ID is used.
HW_OCOTP_ROM2:0x8002C1C0:15:0	USB_PID—Product ID used in recovery mode.

31.2.2. Persistent Bits

Persistent bits are used to control certain features in the ROM, as shown [Table 1144](#). For more information on the persistent bits, see [Chapter 19, “Real-Time Clock, Alarm, Watchdog, Persistent Bits”](#) on page 617.

Table 1144. Persistent Bits

PERSISTENT BIT	FUNCTION
HW_RTC_PERSISTENT1:0x8005C070:3	SD_SPEED_ENABLE—If this bit is set, ROM will put the SD/MMC card in high-speed mode.
HW_RTC_PERSISTENT1:0x8005C070:2	NAND_SDK_BLOCK_REWRITE—The NAND driver sets this bit to indicate to the SDK that the boot image has ECC errors that reached the warning threshold. The SDK must regenerate the firmware by copying it from the backup image. The SDK will clear this bit.
HW_RTC_PERSISTENT1:0x8005C070:1	NAND_SECONDARY_BOOT—When this bit is set, the ROM attempts to boot from the secondary image if the boot driver supports it. This bit is set by the ROM boot driver and cleared by the SDK after repair.
HW_RTC_PERSISTENT1:0x8005C070:0	FORCE_RECOVERY—When this bit is set, the ROM code forces the system to boot in recovery mode, regardless of the selected mode. The ROM will clear the bit.

31.3. Memory Map

[Figure 144](#) shows the memory map for the boot loader.

ROM boot code resides in the top 64K address space. The boot code uses the top 16K of OCRM for data, and the 4K just below it should be reserved for patching the ROM. This leaves 492K of OCRM for loading application data.

If a system uses external memory, then a boot image may be created that first loads a small SDRAM initialization program into OCRM. The program will setup the SDRAM, and then the rest of the boot image may continue to load, overwriting the initialization program in OCRM.

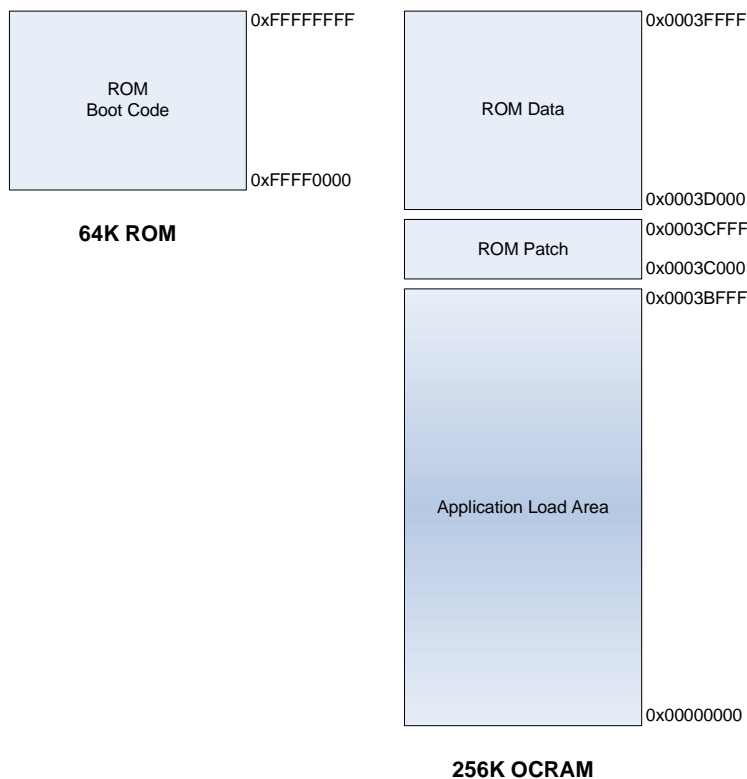


Figure 144. Boot Loader Memory Map

31.4. General Boot Procedure

During ROM startup, the boot mode is determined, and then control is passed to the boot loader. The loader first calls an init function for the boot driver responsible for reading boot images from the target boot port. The driver initializes the hardware port and external device, and then finds the boot image on that device. The loader then requests boot image data from the driver. The boot image contains commands for loading code and data into memory, so the loader will interpret these commands and load the boot image into memory. At the end of a boot image, the loader passes control to the code that was loaded.

Boot images are encrypted, and customers have full control over the encryption keys by setting the CRYPTO_KEY TOP bits. Boot images are created by the SigmaTel-supplied elftosb application.

31.4.1. Preparing Bootable Images

Preparing a bootable image for all boot modes includes the following high-level steps:

- Prepare the ELF file for the firmware that is to be booted by the STMP3770 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 mode sections that follow.

31.4.2. **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 145.

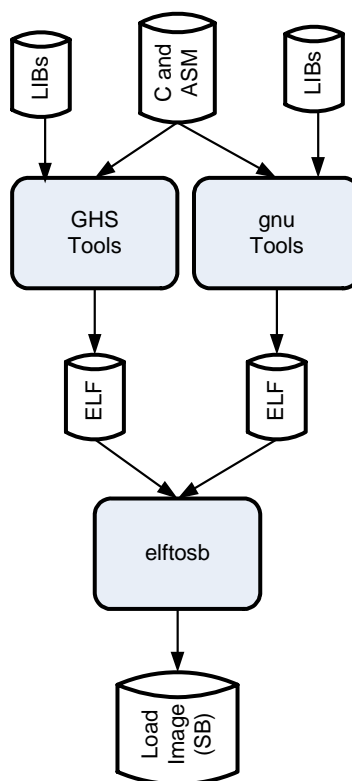


Figure 145. Creating a Boot Loader Image

31.5. NOR Boot Mode

Both 8-bit and 16-bit NORs are supported. Boot images must start at byte address 0x0.

31.6. I²C Boot Mode

EEPROMs must have the slave address 0xA0 (i.e., '1010000R', where R indicates a read if 1 and a write if 0). Also, the EEPROM must have exactly a two-byte "sub" address.

Boot images must start at address 0x0 of the EEPROM and cannot exceed 64 Kbytes in size. The I²C port is set to run at 400 kHz.

31.7. SPI Boot Mode

SPI memories are either EEPROMs or NORs.

By default, the SPI serial clock is set to 0.9 MHz for EEPROMs and 12 MHz for NORs. The SSP_SCK_INDEX OTP bits are used to change the SPI serial clock from defaults. These bits serve as the index for the SSP HAL clock rate array (see [Section 31.7.2](#) for details on the clock rate array).

The defaults may also be changed by using the ConfigBlock.Clocks field ([Section 31.7.2](#)). If ConfigBlock.Clocks.SspClockConfig is non-zero, then that struct will be used to change the SPI SCK rate and will override the SSP_SCK_INDEX OTP setting.

This driver supports only 2-byte addresses for EEPROMs and 3-byte addresses for NORs.

This boot mode supports SPI mode 0 only.

31.7.1. Media Format

The media is arbitrarily partitioned into 128-byte “sectors”. An optional configuration block may reside on the media at byte address 0. This block has the following format:

```

/// \brief Spi media configuration block structs
typedef struct _spi_ConfigBlockFlags
{
    uint32_t  DisableFastRead:1; // Ignored for Spis
                                // 0 - Do not disable fast reads
                                // 1 - Disable fast reads
} spi_ConfigBlockFlags_t;

typedef struct _spi_ConfigBlockClocks
{
    uint32_t      SizeOfSspClockConfig; // sizeof(ssp_ClockConfig_t)
    ssp_ClockConfig_t  SspClockConfig; // SSP clock configuration structure. A null
                                      // structure indicates no clock change.
} spi_ConfigBlockClocks_t;

typedef struct _spi_ConfigBlock // Little Endian
{
    uint32_t  Signature; // 0x4D454D53, or "SMEM"
    uint32_t  BootStartAddr; // Start address of boot image. Must be >=
                            // sizeof(spi_ConfigBlock_t)
    uint32_t  SectorSize; // Sector size in bytes. Overrides ROM default
                        // of 128-bytes. Max is 1024-bytes. 0 is
                        // default 128-bytes.
    spi_ConfigBlockFlags_t  Flags; // Various flags. See spi_ConfigBlockFlags
    spi_ConfigBlockClocks_t  Clocks; // SCK clock update structure.
} spi_ConfigBlock_t;

```

If the block is present, then the boot image is found at the address specified by BootStartAddr. If the block is not present, then it assumes that the boot image resides on the media starting at byte address 0.

31.7.2. SSP

The SSP is used for the SPI boot mode.

The following table is used to look up a requested speed. If the speed is not an exact match, the boot ROM picks the next lowest value. Speed values can range from 1 to 50 MHz. A speed value of 0 is not allowed.

```

/// Lookup Table entry
typedef struct _ssp_clockConfig

```

```

{
    int    clkSel      :1;    //!< Clock Select (0=io_ref 1=xtal_ref)
    int    io_frac     :6;    //!< IO FRAC 18-35 (io_frac+16)
    int    ssp_frac    :9;    //!< SSP FRAC (1=default)
    int    ssp_div     :8;    //!< Divider: Must be an even value 2-254
    int    ssp_rate    :8;    //!< Serial Clock Rate
}
ssp_clockConfig_t;

```

The table is loaded with the clock rates listed in [Table 1145](#).

Table 1145. SCK Clock Standard Values Lookup Table

SCK	N/A	.9	1	2	4	6	8	10	12	16	20	24	40	48	X	X
INDEX	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CLK_SEL	X	1	1	1	1	1	0	0	1	0	0	0	0	0		X
IOFRAC	X	X	X	X	X	X	18	18	X	18	18	18	18	18		X
SSP_FRAC	X	X	X	X	X	X	6	6	X	5	6	10	6	5		X
SSP_CLK		24	24	24	24	24	80	80	24	96	80	48	80	96		X
SSP_DIV	X	26	24	12	6	4	10	8	2	6	4	2	2	2		X
SSP_RATE	X	0	0	0	0	0	0	0	0	0	0	0	0	0		X

31.8. SD/MMC Boot Mode

SD/MMC boot mode supports booting from SD/MMC cards adhering to the following specifications:

- iNand Product Manual, Version 2.1
- SD Specifications, Part 1, Physical Layer Specification, Version 1.10
- SD Specifications, Part 1, Physical Layer Specifications, Version 2.0 Draft
- MultiMediaCard System Specification, Version 4.1
- MultiMediaCard System Specification, Version 4.2

Note, however, that this mode does not support dynamic insertion removal, so systems will typically not include removable cards.

The following modes are supported:

- SD/MMC on SSP1 at 1.8-V
- SD/MMC on SSP1 at 3.3-V
- SD/MMC on SSP2 at 1.8-V
- SD/MMC on SSP2 at 3.3-V

The SD_POWER_GATE_GPIO eFuse bits indicate which GPIO pin to use for controlling an external power gate for the connected device.

SD_POWER_GATE_GPIO	GATE GPIO
00b	PWM3
01b	PWM4
10b	ROTARYA
11b	NONE

If a gate GPIO is used, then the driver will use the SD_POWER_UP_DELAY eFuse to determine the amount of time, in 10-ms increments, to wait unit starting the 1-ms initialization sequence. This eFuse field is 6-bits wide, providing from 10–600 ms of

delay. If the field is 000000b, then the delay is a default 20 ms. If no gate GPIO is specified in SD_POWER_GATE_GPIO, then the delay is skipped.

The SSP ports on the STMP3770 top out at 50 MHz with 20–40 pF loading. By default, the serial clock is set to 12 MHz. If the SD_SPEED_ENABLE persistent bit is set, then the driver will use a maximum speed based on the results of device identification and limited by choices available in the SSP clock index.

This mode supports the 1-bit, 4-bit, and 8-bit data MMC/SD buses. The SD_BUS_WIDTH efuse bits selects how many bus pins are physically available for the SSP port. Bus width will be limited based on these bits, as well as the bus width capabilities indicated by the connected device.

SD_BUS_WIDTH	WIDTH
00b	4-bit
01b	1-bit
10b	8-bit
11b	Reserved

31.8.0.1. Boot Control Block (BCB)

The last physical sector of the device contains a media configuration block. This block contains the sector address of the boot image. The config block has the following format:

```
typedef struct {
    drive_type_t    eDriveType;
    uint32_t        Tag;
    uint32_t        Reserved[5];
} media_regions_t;

typedef struct {
    uint32_t        Signature;
    uint32_t        Version;
    uint32_t        Reserved[1];
    uint32_t        NumRegions;
    media_regions_t Regions[];
} media_config_block_t;
```

The driver will first verify the Signature and Version, then search all NumRegions for the appropriate Tag. [Table 1146](#) shows the expected values for these parameters.

Table 1146. Media Config Block Parameters

FIELD	VALUE
Signature	0x4D454D53
Version	0x00000001
Tag	0x00000050

31.8.0.2. Master Boot Record (MBR)

The first block contains block of the device contains the master boot record (MBR). The MBR is identified by its signature located at offset 0x1FE of the first sector. The partition table is stored at address 0x1BE. The SigmaTel partition is identified by MBR_SIGMATEL_ID at an offset 0x04 from partition table. The boot image address

is stored at offset 0x08 of partition table and size of the image at offset 0x10 of partition table.

FIELD	VALUE
MBR Signature	0x55aa
MBR_SIGMATEL_ID	'S'

31.8.0.3. Device Identification

SD/MMC boot mode uses the identification processes specified in SD Specifications, Part 1, Physical Layer Specifications, Version 2.0 Draft and MultiMediaCard System Specification, Version 4.2.

31.9. NAND Boot Mode

31.9.1. NAND Control Block (NCB)

As part of the NAND media initialization, the ROM driver uses safe NAND timings to search for a NAND Control Block (NCB) that contains the optimum NAND timings and the Factory Marked Bad Block Table. A flowchart is shown in [Figure 146](#).

An initial assumption is that the NAND driver determines the ECC size from the boot mode. If the NCB is found, the optimum NAND timings are loaded for further reads. If the ECC fails or the fingerprints do not match, the Block Search state machine increments 64 pages and reads that page until 2^n efNANDBootSearchLimit pages have been read. The 64-page stride value was picked as the smallest block size in many of the current NANDs.

If a NCB cannot be found, the driver sets the NAND_SDK_BLOCK_REWRITE persistent bit and retries the Block Search looking for NCB2. If the second search fails, the NAND driver responds with an error and the boot ROM enters recovery mode.

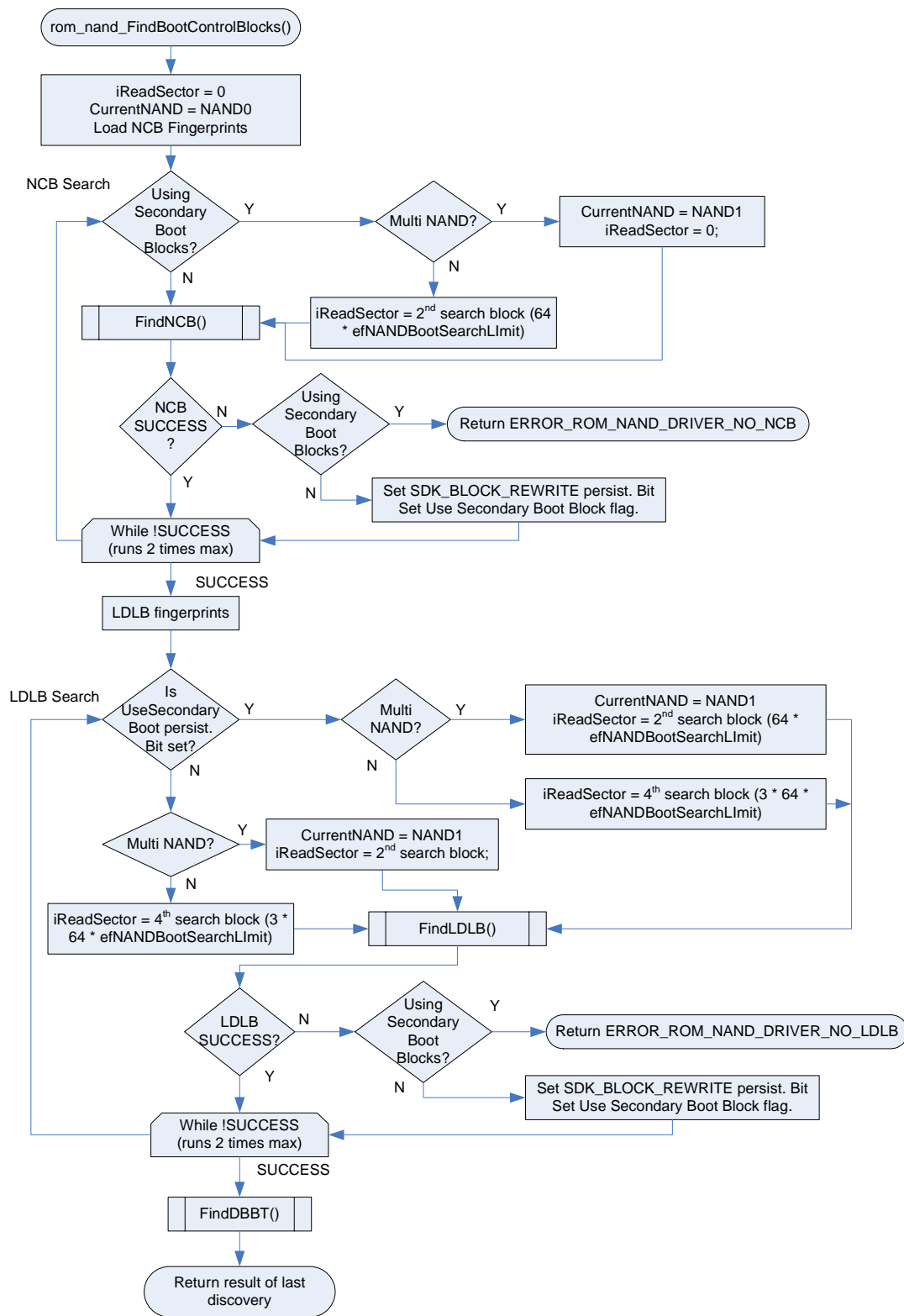


Figure 146. FindBootControlBlocks Flowchart

Upon finding the NCB, the NAND parameters are read from the NCB, and the NAND driver updates the fail-safe parameters with the new parameters. These parameters include NAND timing, page size, and block size.

The next step is to find the Logical Drive Layout Block (LDLB), which uses the same search method as the NCB but the starting address will be different. The LDLB contains the Media Table, a pointer to the Discovered Bad Block Table (DBBT), and the sector information for the initial boot image. After the LDLB is read, the DBBT is loaded, and the initial boot image is loaded using the initial boot image starting address pointer.

If the ECC fails or the fingerprints do not match, the Block Search state machine will increment 64 pages and read that page until 2^n efNANDBlockSearchLimit pages have been read. If a LDLB cannot be found, the driver sets the NAND_SDK_BLOCK_REWRITE persistent bit and retries the Block Search. If the second search fails, the NAND driver responds with an error, and the boot ROM enters recovery mode. The block search flowchart is shown in Figure 147.

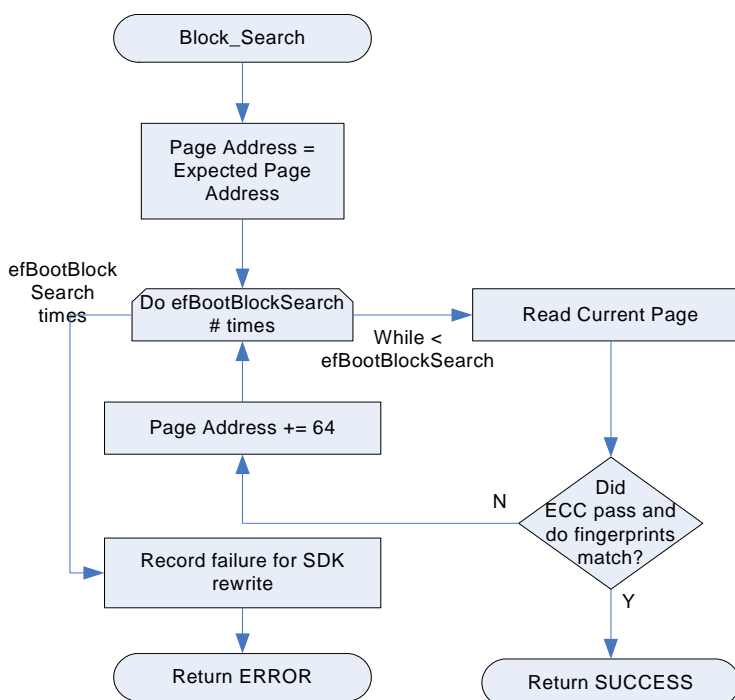


Figure 147. Block Search Flowchart

Once the LDLB is identified, more information about the boot image geometry is determined, including information required to calculate the appropriate boot image starting NAND, starting sector, stride, etc.

The NCB and LDLB search and load function also monitors the ECC correction threshold and sets the NAND_SDK_BLOCK_REWRITE persistent bit if the threshold exceeds three corrections in the four-bit ECC case and seven corrections in the eight-bit ECC case. One bit is used for all boot block images. If the NAND_SDK_BLOCK_REWRITE bit is set, the ROM continues loading the image, but the SDK will need to “refresh” the boot blocks at a later time.

31.9.2. Expected NAND Layout

The ROM expects the layout described in this section and illustrated in [Figure 148](#) when booting from a single NAND configuration. The first search area contains the NAND Configuration Block (NCB) followed by an alternate NCB in the second search area. The third search area contains the Logical Drive Layout Block (LDLB), which is followed by an alternate LDLB in the fourth search area.

NAND Control Block (NCB1) NAND Physical Params with appropriate ECC <input type="checkbox"/> 1. # of NANDs <input type="checkbox"/> 2. Timing Parameters <input type="checkbox"/> Factory Marked Bad Block Table <input type="checkbox"/> Fingerprints for identification	1 st Search Block
Alternate NCB (NCB2) NAND Physical Params with appropriate ECC <input type="checkbox"/> 1. # of NANDs <input type="checkbox"/> 2. Timing Parameters <input type="checkbox"/> Factory Marked Bad Block Table <input type="checkbox"/> Fingerprints for identification	2 nd Search Block
Logical Drive Layout Block (LDLB1) Infrequently written data with appropriate ECC (4 bit or 8 bit) <input type="checkbox"/> Media Table (starting sectors of each drive and size of drive) <input type="checkbox"/> Pointer to the Discovered Bad Block Table (BB discovered during operation) <input type="checkbox"/> Initial Boot Applet pointer (Chip and sector)	3 rd Search Block
Alternate LDLB (LDLB2) Infrequently written data with appropriate ECC (4 bit or 8 bit) <input type="checkbox"/> Media Table (starting sectors of each drive and size of drive) <input type="checkbox"/> Pointer to the Discovered Bad Block Table (BB discovered during operation) <input type="checkbox"/> Initial Boot Applet pointer (Chip and sector)	4 th Search Block
Discovered Bad Block Table (DBBT1) Table of Bad Blocks discovered during SDK operation. Fingerprints.	5 th Search Block
Discovered Bad Block Table (DBBT2) Table of Bad Blocks discovered during SDK operation. Fingerprints.	
Boot Applet 1 Small boot image loaded from ROM	
Alternate Boot Applet 1 Small boot image loaded from ROM	
And so on...	
Boot Applet y Small boot image loaded from ROM	

Figure 148. Expected NAND Layout

Search areas are defined as 64 pages * efSearchSize. They are defined in such a way that there is at least one extra block to hedge against a Boot Control Block (BCB) going bad during operation. If a Boot Control Block goes bad during operation, the data is copied from the original BCB to the extra BCB, and the original BCB is written with zeros. Since we rely upon both the ECC being correct and the fingerprints, overwriting the BCB with zeros should invalidate all the data. The search algorithm will search an entire search area before looking for the backup or secondary Boot Control Block.

31.9.2.1. NAND Config Block

The primary NAND Config Block (NCB) resides on the NAND attached to GPMI_CE0. The NCB is the first sector in the first good block. In the single NAND configuration, a copy of the NCB also resides on the NAND—its location is immediately after NCB1 - 2nd search area. In the case of multiple NANDs, copies of the NCB, LDLB, and DBBT are located on the first two NANDs. This case is shown in more detail in [Section 31.9.2.3](#).

To determine the appropriate ECC, the boot mode contains the desired ECCSize. Since the boot mode is written to the eFuse for manufacturing, the eFuse may also contain the ECC size.

To determine a good NCB, all three fingerprints must match and the ECC must not fail.

The data held in the NCB includes:

- NAND Timing parameters.
- Which NANDs are in the system (GPMI_CE0, GPMI_CE1, GPMI_CE2, GPMI_CE3)—this is also in the eFuse.
- Geometry information (sectors per block, sectors per page, etc.).
- Additionally, the NCB is marked with three fingerprints in the sector.
- Factory Marked Bad Block Table.
- Pointer to physical sector of LDLB (on this chip) and, in a single NAND configuration, a pointer to the physical sector of LDLB2.

31.9.2.2. Logical Drive Layout Block

A copy of the Logical Drive Layout Block (LDLB) resides on each NAND. A copy of the LDLB also resides on the NAND if it is a single NAND configuration, or on the first two NANDs if it is a multi-NAND configuration (See [Figure 149](#)). Its location is determined using the search area described above. The starting page will be the beginning of the 3rd search area. The LDLB has the same ECC as the rest of the system drives. The data held in the LDLB includes:

- Start Sector and stride of boot region.
- Start Sector and stride of alternate boot region.
- Tag of boot region.
- The LDLB is marked with three fingerprints in the sector.
- Media Table (Layout of drives on the NAND—used for SDK only).
- Pointer to the Discovered Bad Block Table and Alternate Discovered Bad Block Table (if a single NAND).

31.9.2.3. Firmware Layout on the NAND

The boot image firmware is stored in the NAND as shown in [Figure 149](#).

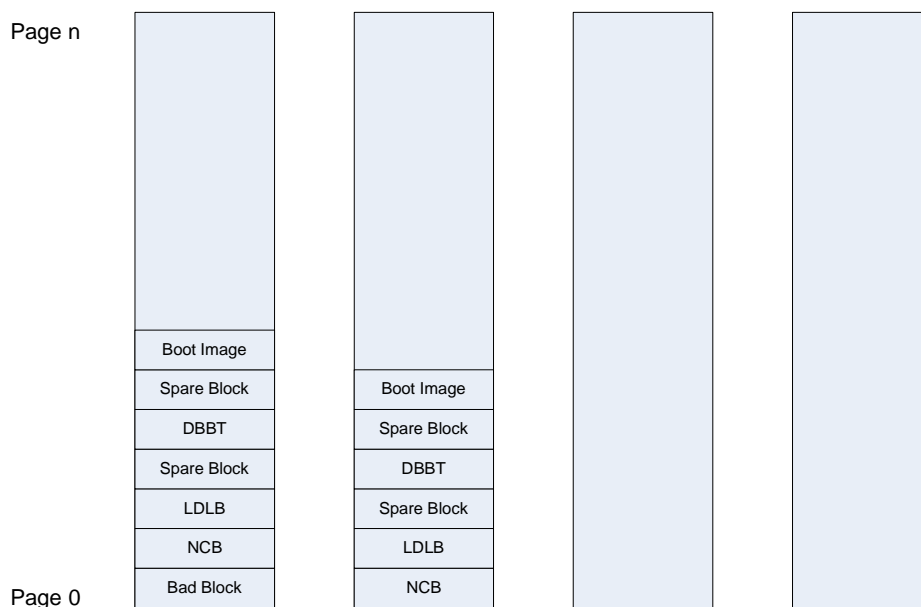


Figure 149. NAND Layout—Multiple NANDs

A pointer in the LDLB contains the physical sector address (on that NAND) of the boot image. The boot image is typically located on the first good block after the NCB, LDLB, and DBBT blocks and the additional blocks reserved for Boot Control Blocks that become Bad Blocks. The next sector will be drawn from the same block, until all sectors are drawn from the block. Then, the first sector of the subsequent block on NAND0 will be read. All boot blocks are located on the first NAND. If a block is determined to be bad (from the Discovered Bad Block Table or fails the ECC), then the algorithm will skip to the next block on NAND0, etc.

In a multiple NAND configuration, the first two NANDs will have the NCB, LDLB, DBBT, and boot images.

The DBBT will be duplicated across all the NANDs.

31.9.2.4. Recovery From a Failed Boot Firmware Image Read

The mechanism for recovering from a failed NCB or LDLB read is covered in [Section 31.9.1](#). The SDK is warned about impending errors with the NAND_SDK_BLOCK_REWRITE persistent bit and is notified of firmware boot errors with the NAND_SECONDARY_BOOT persistent bit.

In the case of a warning, the read routine will monitor the ECC threshold and set the NAND_SDK_BLOCK_REWRITE bit if the threshold is within one symbol of the maximum correctable number of symbols. The ROM continues to load from the primary boot image. At a later time, the SDK will refresh the primary boot images by copying and rewriting the primary boot image blocks.

If an error is discovered while reading the boot firmware image, the NAND driver sets the NAND_SECONDARY_BOOT persistent bit and resets the device. Booting

will proceed the second time from the secondary boot images. If booting continues uninterrupted, no “unwinding” needs to take place at the loader level.

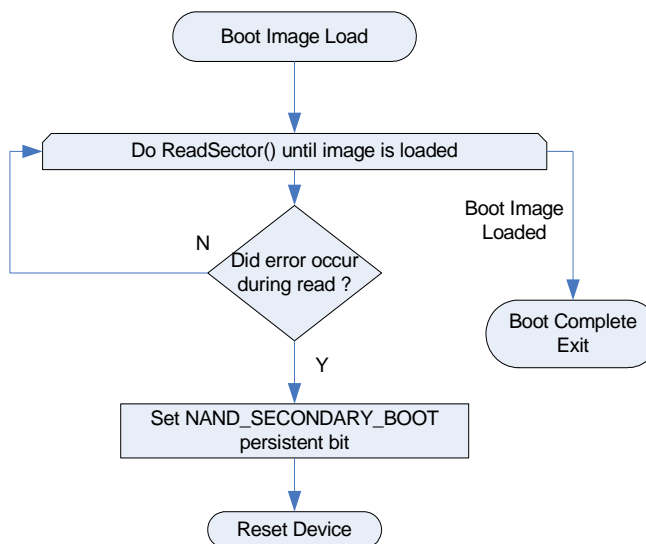


Figure 150. Boot Image Recovery

31.9.2.5. Bad Block Handling in the ROM

Bad blocks are not an issue for the Boot Control Blocks of the NAND because the NCB and LDLB Boot Control Blocks are found from a search as described in [Section 31.9.1](#). The search for the Discovered Bad Block Table Header Control Block works with a similar mechanism. The search starts where the LDLB indicates the DBBT should be and progresses for 2^n efNANDBootSearchLimit times in the same fashion as the search described in [Figure 147](#).

If the DBBT is not found, the ROM will search for the DBBT2. It will not change to boot from the secondary boot blocks even if the DBBT2 is found. Because the DBBT2 must be duplicated across all the NANDs, this is a safe assumption.

If no DBBTs are found, the ROM will assume there are no Bad Blocks and will continue to boot with no Bad Blocks.

During firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block. If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block checked.

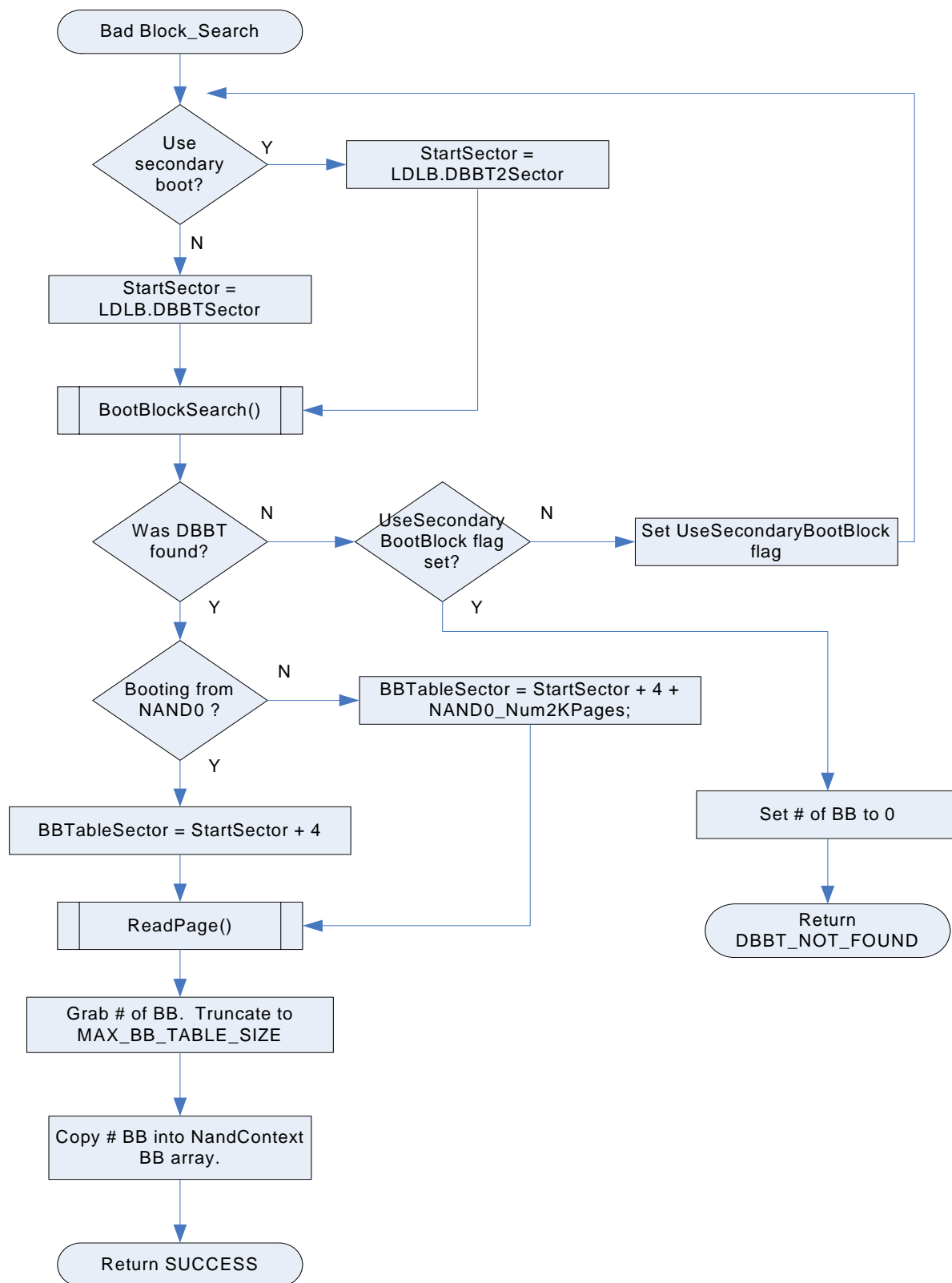


Figure 151. Bad Block Search

The DBBT structure is contained within a block and is discussed in more detail below. [Figure 152](#) depicts the layout of the Discovered Bad Block Table block. The first 8K are reserved for the DBBT Header. The following pages are used by the DBBT for each NAND.

The Bad Block table for each NAND begins on a 2KB boundary that coincides with current NAND page sizes and is a subset of future NAND page sizes. The BBT can extend beyond 2K, which is the purpose of the #2K_num, and translates to the number of 2K pages that NAND0 through NAND3 require. In this way, the ROM can quickly index to the appropriate NAND table.

Only the Bad Blocks for the current NAND are loaded. In other words, in a dual NAND configuration, if the primary boot blocks are being used, then NAND0's BBTable will be loaded. Conversely, if the secondary boot blocks are being used, then NAND1's BB table will be loaded into the ROM's NandContext Bad Block array in NAND.

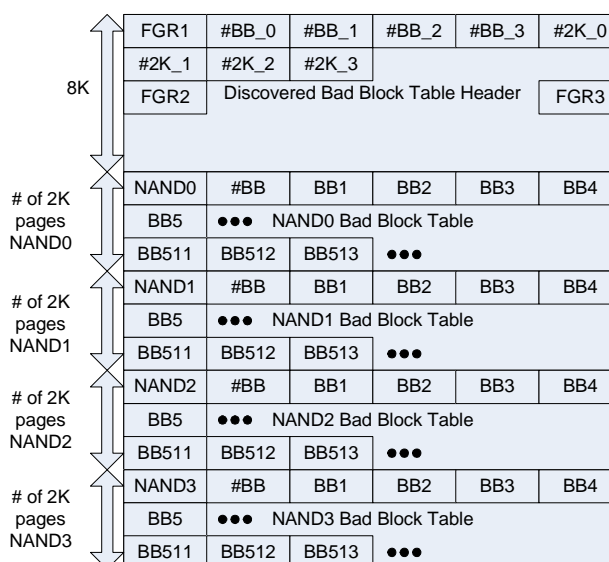


Figure 152. DBBT Layout

31.9.2.6. NAND Control Block Structure and Definitions

The NCB structure is as follows:

```
typedef struct _NAND_Control_Block
{
    uint32_t    m_u32Fingerprint1;
    struct
    {
        uint8_t m_u8DataSetup;
        uint8_t m_u8DataHold;
        uint8_t m_u8AddressSetup;
        uint8_t m_u8DSAMPLE_TIME;
    } NAND_Timing;
    uint32_t    m_u32DataPageSize;        //!< 2048 for 2K pages, 4096 for 4K pages.
    uint32_t    m_u32TotalPageSize;        //!< 2112 for 2K pages, 4314 for 4K pages.
    uint32_t    m_u32SectorsPerBlock;      //!< Number of 2K sections per block.
    uint32_t    m_u32SectorInPageMask;     //!< Mask for handling pages > 2K.
    uint32_t    m_u32SectorToPageShift;    //!< Address shift for handling pages > 2K.
    uint32_t    m_u32NumberOfNANDs;        //!< Total Number of NANDs - not used by ROM.

    uint32_t    m_u32Fingerprint2; // @ word offset 10

    uint32_t    m_u32NumRowBytes;           //!< Number of row bytes in read/write transactions.
    uint32_t    m_u32NumColumnBytes;        //!< Number of column bytes in read/write transactions.
    uint32_t    m_u32TotalInternalDie;      //!< Number of separate chips in this NAND.
    uint32_t    m_u32InternalPlanesPerDie;  //!< Number of internal planes -
                                           // !< treat like separate chips.
    uint32_t    m_u32CellType;              //!< MLC or SLC.
    uint32_t    m_u32ECCType;               //!< 4 symbol or 8 symbol ECC?
    uint32_t    m_u32Read1stCode;           //!< First value sent to initiate a NAND
                                           //!< Read sequence.
    uint32_t    m_u32Read2ndCode;           //!< Second value sent to initiate a NAND
                                           //!< Read sequence.

    uint32_t    m_u32Fingerprint3; // @ word offset 20
} NAND_Control_Block;
```

Where:

m_u32Fingerprint1—32 bit word consisting of STMP (stored as 0x504d5453)

m_u32Fingerprint2—32 bit word consisting of NCB (stored as 0x2042434E)

m_u32Fingerprint3—32 bit word consisting of RBIN (stored as 0x4E494252)

m_u32_NumberOfNANDs—32 bit word detailing the number and configuration of NANDs that are in the system.

NAND_Timing structure

m_u8DataSetup—Number of nanoseconds required for NAND Data Setup.

m_u8DataHold—Number of nanoseconds required for NAND Data Hold.

m_u8AddressSetup—Number of nanoseconds required for NAND Data Setup.

m_u8DSAMPLE_TIME—Number of ½ GPML cycles to wait before latching Read.

m_u32DataPageSize—Number of bytes in Data section of a page.

m_u32TotalPageSize—Number of bytes in a page - total.

m_u32SectorsPerBlock—Number of 2K sections per block. (4K pages = 2 x #pages

m_u32SectorInPageMask—Mask for handling pages > 2K.

m_u32SectorToPageShift—Address shift for handling pages > 2K.

m_u32NumRowBytes—Number of row bytes in read/write transactions.

m_u32NumColumnBytes—Number of column bytes in read/write transactions.

m_u32TotalInternalDie—Number of die in this chip.

m_u32InternalPlanesPerDie—Number of planes or districts per die.

m_u32CellType—Type of NAND Cell—SLC, MLC.

m_u32ECCSize—Type of ECC for this NAND—4 bit or 8 bit ECC per 512 bytes.

m_u32Read1stCode—Read command code—1st byte.

m_u32Read2ndCode—Read command code—2nd byte.

31.9.2.7. Logical Drive Layout Block Structure and Definitions

The first 512 bytes of the LDLB structure are as follows:

```
typedef struct _LogicalDriveLayoutBlock
{
    uint32_t    m_u32Fingerprint1;
    struct
    {
        uint16_t    m_ul6Major;
        uint16_t    m_ul6Minor;
        uint16_t    m_ul6Sub;
    } LDLB_Version;
    uint32_t    m_u32Fingerprint2;
    uint32_t    m_u32Firmware_startingNAND;
    uint32_t    m_u32Firmware_startingSector;
    uint32_t    m_u32Firmware_sectorStride;
    uint32_t    m_u32SectorsInFirmware;
    uint32_t    m_u32Firmware_startingNAND2;
    uint32_t    m_u32Firmware_startingSector2;
    uint32_t    m_u32Firmware_sectorStride2;
    uint32_t    m_uSectorsInFirmware2;
    struct
    {
        uint16_t    m_ul6Major;
        uint16_t    m_ul6Minor;
        uint16_t    m_ul6Sub;
    } FirmwareVersion;
    uint32_t    Rsvd[10];
    uint32_t    m_u32DiscoveredBBTableSector;
    uint32_t    m_u32DiscoveredBBTableSector2;
    uint32_t    m_u32Fingerprint3;
    uint32_t    RSVD[100]; // Region configuration used by SDK only.
} LogicalDriveLayoutBlock_t;
```

Where:

m_u32Fingerprint1—32 bit word consisting of STMP (stored as 0x504d5453)

m_u32Fingerprint2—32 bit word consisting of LDLB (stored as 0x424C444C)

m_u32Fingerprint3—32 bit word consisting of RBIL (stored as 0x4C494252)

LDLBVersion structure—must be set to Major = 1, Minor = 0; Sub = anything.

m_u32Firmware_startingNAND/m_u32Firmware_startingNAND2—Which NAND holds the firmware that will get loaded. This is zero-based, so NAND0 will be 0, NAND1 will be 1, etc.

m_u32Firmware_startingSector/m_u32Firmware_startingSector2—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). This is zero-based on the specific NAND (i.e., Sector0 of NAND3 will be 0 which may be physical sector 2048 (assuming NANDs are 1024 sectors per NAND) of the entire group of NANDs. Use 0 instead of 2048.)

m_u32Firmware_sectorStride/m_u32Firmware_sectorStride2—This 32 bit word describes how consecutive pages are read. Are the pages concatenated (read sequentially from the same block, then the next block on the same NAND), striped across NANDs on a sector basis (in NAND2 case -> sector 0 on NAND0, sector 0 on NAND1, sector 1 on NAND0, sector 1 on NAND1, etc.), striped across NANDs

on a Block basis (sector 0-63 on NAND0, then sector 0-63 on NAND1, etc.). Not used on current ROM.

m_u32SectorsInFirmware—Number of sectors (2K byte chunks) that will be read by the ROM.

m_u32DiscoveredBBTableSector—Physical sector of Discovered Bad Block Table.

m_u32DiscoveredBBTableSector—Physical sector of secondary Discovered Bad Block Table.

31.9.2.8. *Discovered Bad Block Table Header Layout Block Structure and Definitions*

The first 512 bytes of the DBBT header structure are as follows:

```
typedef struct _LogicalDriveLayoutBlock
{
    uint32_t    m_u32Fingerprint1;
    uint32_t    m_u32NumberBB_NAND0;
    uint32_t    m_u32NumberBB_NAND1;
    uint32_t    m_u32NumberBB_NAND2;
    uint32_t    m_u32NumberBB_NAND3;
    uint32_t    m_u32Number2KPagesBB_NAND0;
    uint32_t    m_u32Number2KPagesBB_NAND1;
    uint32_t    m_u32Number2KPagesBB_NAND2;
    uint32_t    m_u32Number2KPagesBB_NAND3;
    uint32_t    m_u32Fingerprint2;
    uint32_t    RSVD[20]; //
    uint32_t    m_u32Fingerprint3;
} LogicalDriveLayoutBlock_t;
```

Where:

m_u32Fingerprint1—32 bit word consisting of STMP (stored as 0x504d5453)

m_u32Fingerprint2—32 bit word consisting of DBBT (stored as 0x54424244)

m_u32Fingerprint3—32 bit word consisting of RbId (stored as 0x44494252)

m_u32NumberBB_NAND0—The number of Bad Blocks stored in this table for NAND0.

m_u32NumberBB_NAND1—The number of Bad Blocks stored in this table for NAND1.

m_u32NumberBB_NAND2—The number of Bad Blocks stored in this table for NAND2.

m_u32NumberBB_NAND3—The number of Bad Blocks stored in this table for NAND3.

m_u32Number2KPagesBB_NAND0—The number of 2K pages that the Bad Block table for NAND0 consumes.

m_u32Number2KPagesBB_NAND1—The number of 2K pages that the Bad Block table for NAND1 consumes.

m_u32Number2KPagesBB_NAND2—The number of 2K pages that the Bad Block table for NAND2 consumes.

m_u32Number2KPagesBB_NAND3—The number of 2Kpages that the Bad Block table for NAND3 consumes.

31.9.2.9. Discovered Bad Block Table Layout Block Structure and Definitions

The DBBT structure is as follows:

```
typedef struct _BadBlockTableNand_t
{
    uint32_t          uNAND;
    uint32_t          uNumberBB;
    int16_t           u16BadBlock[];
} BadBlockTableNand_t;
```

Where:

uNAND— 32 bit word denoting which NAND the table describes.

uNumberBB—32 bit word holding the number of Bad Blocks.

u16BadBlock—Array of 16 bit words holding the Bad Blocks for this particular NAND.

31.9.3. Typical NAND Page Organization

31.9.3.1. 2K Page Organization on the NAND

The ECC engine expects the following layout of data in a 2K page. Note that this configuration conflicts with the factory-marked bad block byte.

Each 512 byte group has an ECC and the metadata has its own ECC.

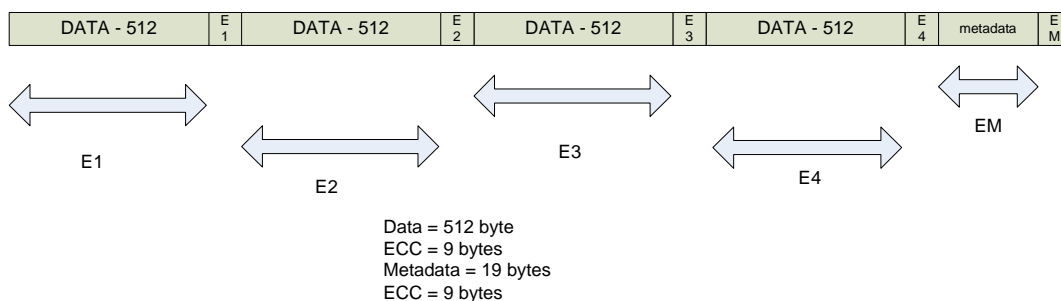


Figure 153. 2K Page Layout in NAND

31.9.3.2. In-Memory Organization

A data read or data write does not have the ECC interspersed in the data as shown in Figure 153. Instead, on a read, the data is stored as follows in the STMP3770 on-chip RAM memory.

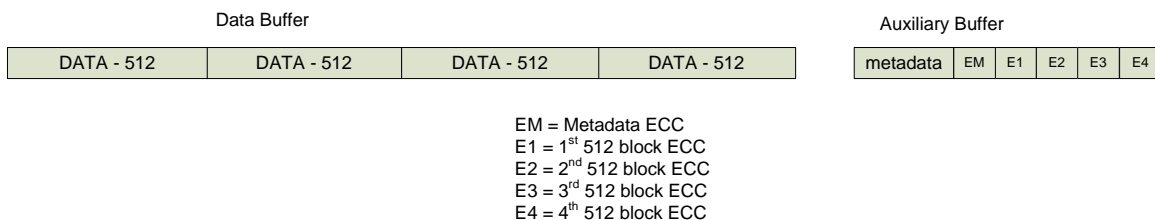


Figure 154. 2K Page Layout in On-Chip Memory

The ECC is stripped off and placed in the Auxiliary buffer. This Auxiliary buffer is 188 bytes for a 2K page NAND. In the write case, the 2048 byte buffer is sent to the GPML, but the Auxiliary buffer must be allocated for scratch pad space.

31.9.3.3. Metadata

The Metadata consists of 19 bytes of the 2112 bytes in a page. Currently unused bytes are set to 0, but they can be any value as long as they are included in the ECC calculation. Currently, the ROM does not care about the metadata. Only the ECC bytes are important.

The ROM expects the boot image sectors to have metadata as described in [Figure 155](#).

The EM shown in [Figure 154](#) is the ECC for the metadata.

512	RS_1_ECC1	1554	RS_3_ECC1
513	RS_1_ECC2	1555	RS_3_ECC2
514	RS_1_ECC3	1556	RS_3_ECC3
515	RS_1_ECC4	1557	RS_3_ECC4
516	RS_1_ECC5	1558	RS_3_ECC5
517	RS_1_ECC6	1559	RS_3_ECC6
518	RS_1_ECC7	1560	RS_3_ECC7
519	RS_1_ECC8	1561	RS_3_ECC8
520	RS_1_ECC9	1562	RS_3_ECC9

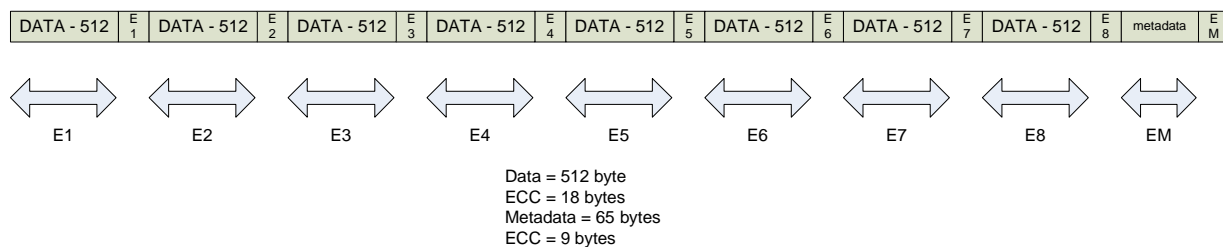
1033	RS_2_ECC1	2075	RS_4_ECC1
1034	RS_2_ECC2	2076	RS_4_ECC2
1035	RS_2_ECC3	2078	RS_4_ECC3
1036	RS_2_ECC4	2079	RS_4_ECC4
1037	RS_2_ECC5	2080	RS_4_ECC5
1038	RS_2_ECC6	2081	RS_4_ECC6
1039	RS_2_ECC7	2082	RS_4_ECC7
1040	RS_2_ECC8	2083	RS_4_ECC8
1041	RS_2_ECC9	2084	RS_4_ECC9

Figure 155. Redundant Area—2K

31.9.3.4. 4K Page Organization on the NAND

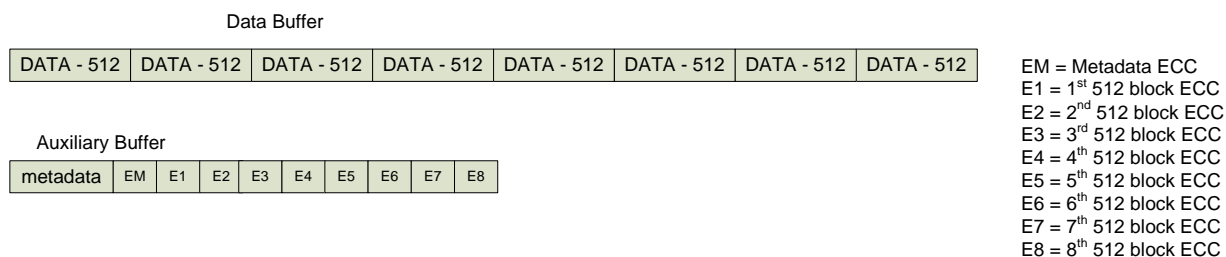
The ECC engine expects the following layout of data in a 4K page. Note that this configuration conflicts with the factory marked bad block byte.

Each 512 byte group has an ECC and the metadata has its own ECC.

**Figure 156. 4K Page in NAND**

31.9.3.5. In-Memory Organization

A data read or data write does not have the ECC interspersed in the data as shown in Figure 156. Instead, on a read, the data is stored as follows in the STMP3770 on-chip RAM memory.

**Figure 157. 4K Page Layout in On-Chip Memory**

The ECC is stripped off and placed in the Auxiliary buffer. This Auxiliary buffer is 412 bytes in the 4K page case. In the write case, the 4096 byte buffer is sent to the GPML, but the Auxiliary buffer must be allocated for scratch pad space.

31.10. USB Boot Driver

The USB Boot Driver is implemented as a USB HID class and is referred to as the Recovery HID, or RHID. The RHID serves as a fail-safe mechanism for downloading and communicating with application-specific code.

The system is based on two HID Application collections: the Boot Loader Transfer Controller (BLTC) and the Plug-in Transfer Controller (PITC). Each collection has its own set of HID reports.

31.10.1. Boot Loader Transaction Controller (BLTC)

The BLTC provides a tunnel to download application-specific PITCs to local system memory. The BLTC runs completely from ROM and interfaces directly to the ROM Loader. Typically, a PITC will be packaged on the host and downloaded through the BLTC and ROM Loader straight to OCRAM.

Four HID reports are provided for communication with the BLTC:

- BLTC Command Out (BLCO)
- BLTC Data Out (BLDO)
- BLTC Data In (BLDI)

- BLTC Status In (BLSI)

The BLTC provides a command/data protocol for downloading code to the ROM Loader.

The BLTC has no knowledge of the contents of the data passing through so it is really possible to download anything (not just PITCs).

31.10.2. *Plug-in Transaction Controller (PITC)*

The PITC is a generic command/data/status tunnel that may be used for any type of application. The implementation only specifies the HID report structure and ROM HID-stack installation for a PITC—the protocol implementation is specific to a given PITC. Typically, a PITC will be downloaded to memory via the BLTC.

Four HID reports are provided for communication with a PITC:

- PITC Command Out (PICO)
- PITC Data Out (PIDO)
- PITC Data In (PIDI)
- PITC Status In (PISI)

The command/data protocol is specific to any given PITC.

Only one PITC may be installed at any given time.

31.10.3. *USB IDs and Serial Number*

By default the USB Device Descriptor Vendor ID, Product ID, and serial number are reported as follows:

- Vendor ID—0x066F
- Product ID—0x3700
- Serial Number String—none

If any of the HW_OCOTP_ROM2 bits are blown, then the full contents of that OTP register are used to report the Vendor ID and Product ID. If the ENABLE_USB_BOOT_SERIAL_NUMBER OTP is blown, then a unique serial number will be generated from the factory-programmed SGT_L_OPS3 OTP registers.

31.10.4. *USB Recovery Mode*

USB boot mode is provided as a fail-safe mechanism for writing system firmware to the boot media. The boot mode is not usually entered by the normal methods of setting the boot pins or OTP: the other methods of entering USB boot mode are referred to generally as recovery mode.

An end user can manually start recovery mode by holding the recovery switch for several seconds while plugging in USB. Holding the recovery switch is defined as reading the STMP3770 PSWITCH input as a 0x3. There are several switch circuits that will produce this input. The loader also automatically starts recovery mode if a non-recoverable error is detected from any boot mode other than USB.

If the DISABLE_RECOVERY_MODE OTP bit is blown, then USB boot mode is disabled completely. Attempts to enter USB boot mode via boot pins, OTP, or recovery methods will result in a chip power-down.

32. PIN DESCRIPTIONS

This chapter provides various views of the pinout for the STMP3770.

- Pin definitions for the 100-Pin BGA are in [Section 32.1 on page 916](#).
- Pin definitions for the 100-Pin LQFP are in [Section 32.1 on page 916](#).

The pin tables in this chapter include columns with the headings “Description 1”, “Description 2”, and “Description 3”. These columns refer to the different functions that can be enabled for each individual pin by programming the pin control registers (HW_PINCTRL_MUXSELx). For example:

- Enable the function listed in the “Description 1” column by programming the BANKx_PINx bit field to 00.
- Enable the function listed in the “Description 2” column by programming the BANKx_PINx bit field to 01.
- Enable the function listed in the “Description 3” column by programming the BANKx_PINx bit field to 10.

See [Chapter 33, ‘Pin Control and GPIO’ on page 929](#) for more information.

[Table 1147](#) lists the abbreviations used in the pin tables in this chapter.

Table 1147. 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
ADC	ADC analog pins
CLOCK	Clock pins
DCDC	DC-DC Converter pins
EMI	External Memory Interface pins
ETM	Embedded Trace Macrocell
GPIO	General-Purpose Input/Output pins
GPPI	General-Purpose Media Interface (NAND/CMOS) pins
HP	Headphone pins
I ² C	I ² C pins
LCDIF	LCD Interface pins
LineOut	Line Out pins
LRADC	Low-Resolution ADC/Touch-Screen pins
POWER	Power pins
PWM	Pulse Width Modulator pins
RTC	Real-Time Clock pins
SSP	Synchronous Serial Port pins
SYSTEM	System pins
TIMER	Timer/Rotary Encoder pins
UART	Either debug or application UART pins
USB	USB pins

Note: Almost all digital pins are powered down (i.e., high-impedance) at reset, until reprogrammed. The only exceptions are TESTMODE and DEBUG. These two pins are always active.

32.1. Pin Definitions for 100-Pin BGA

This section includes the following pin information for the 100-pin BGA package:

- [Table 1148, “100-Pin BGA Pin Definitions by Pin Name,” on page 916](#)
- [Table 1149, “100-Pin BGA Pin Definitions by Pin Number,” on page 918](#)
- [Table 1150, “100-Pin BGA Ball Map,” on page 921](#)

Table 1148. 100-Pin BGA Pin Definitions by Pin Name

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
A8	BATT	POWER	P	Battery Input		
B8	DCDC_BATT	DCDC	A	DCDC Battery		
A10	DCDC_GND	DCDC	A	DCDC Ground		
A9	DCDC_LN1	DCDC	A	DCDC Inductor N 1		
B9	DCDC_LN2	DCDC	A	DCDC Inductor N 2		
C8	DCDC_LP	DCDC	A	DCDC Inductor P		
F7	DCDC_MODE	DCDC	A	DCDC Mode Sense		
C9	DCDC_VDDA	DCDC	A	DCDC Analog Power		
C10	DCDC_VDDD	DCDC	A	DCDC Digital Core Power		
B10	DCDC_VDDIO	DCDC	A	DCDC I/O Power		
B1	DEBUG	SYSTEM	I/O	Serial JTAG Debug Port		
J6	EMI_CE2N	EMI	I/O	EMI CE2n	GPMI_CE1n	SSP2_DATA0
H6	EMI_CE3N	EMI	I/O	EMI CE3n	GPMI_CE0n	
H7	GPMI_ALE	GPMI	I/O	NAND ALE		
G5	GPMI_CLE	GPMI	I/O	NAND CLE		
F5	GPMI_CE2N	GPMI	P	NAND2 Chip Enable	LCD_ENABLE	
F2	GPMI_CE3N	GPMI	P	NAND 3 Chip Enable	LCD_DOTCLK	UART2_RX
J1	GPMI_D00	GPMI	I/O	NAND Data 0		
J2	GPMI_D01	GPMI	I/O	NAND Data 1		SSP2_DATA1
K4	GPMI_D02	GPMI	I/O	NAND Data 2		SSP2_DATA2
J3	GPMI_D03	GPMI	I/O	NAND Data 3		SSP2_DATA3
H3	GPMI_D04	GPMI	I/O	NAND Data 4		SSP2_DATA4
H5	GPMI_D05	GPMI	I/O	NAND Data 5		SSP2_DATA5
K5	GPMI_D06	GPMI	I/O	NAND Data 6		SSP2_DATA6
K6	GPMI_D07	GPMI	I/O	NAND Data 7		SSP2_DATA7
K7	GPMI_RDN	GPMI	I/O	NAND Read Strobe		
K8	GPMI_RDY0	GPMI	I/O	NAND0 Ready/Busy#		SSP2_DETECT
J8	GPMI_RDY1	GPMI	I/O	NAND1 Ready/Busy#	UART2_TX	
J7	GPMI_RDY2	GPMI	I/O	NAND2 Ready/Busy#	UART2_RX	SSP2_CMD
E1	GPMI_RDY3	GPMI	P	NAND3 Ready/Busy #	LCD_HSYNC	UART2_TX
K9	GPMI_WP	GPMI	I/O	NAND Write Protect, Renesas Reset		JTAG_TRST
K3	GPMI_WRN	GPMI	I/O	NAND Write Strobe		SSP2_SCK
D6	HP_SENSE	HP	A	Direct-Coupled Headphone Sense		
D5	HP_VGND	HP	A	Direct-Coupled Headphone Virtual Ground		
C5	HPL	HP	A	Headphone Left		
C6	HPR	HP	A	Headphone Right		
E4	I2C_SCL	I ² C	I/O	I ² C Serial Clock	GPMI_RDY3	
D4	I2C_SDA	I ² C	I/O	I ² C Serial Data	GPMI_CE3n	
K10	LCD_BUSY	LCDIF	I/O	LCD Interface Busy	LCD_VSYNC	
J9	LCD_CS	LCDIF	I/O	LCD Interface Chip Select.		

Table 1148. 100-Pin BGA Pin Definitions by Pin Name (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
H10	LCD_D00	LCDIF	I/O	LCD Interface Data 0	ETM_DA0	
H8	LCD_D01	LCDIF	I/O	LCD Interface Data 1	ETM_DA1	
H9	LCD_D02	LCDIF	I/O	LCD Interface Data 2	ETM_DA2	
G10	LCD_D03	LCDIF	I/O	LCD Interface Data 3	ETM_DA3	
F10	LCD_D04	LCDIF	I/O	LCD Interface Data 4	ETM_DA4	
G9	LCD_D05	LCDIF	I/O	LCD Interface Data 5	ETM_DA5	
F9	LCD_D06	LCDIF	I/O	LCD Interface Data 6	ETM_DA6	
E10	LCD_D07	LCDIF	I/O	LCD Interface Data 7	ETM_DA7	
D9	LCD_RD_E	LCDIF	I/O	LCD Interface Data Read or Enable		
D10	LCD_RESET	LCDIF	I/O	LCD Interface Reset Out		
E9	LCD_RS	LCDIF	I/O	LCD Interface Register Select/LCD CCIR Clock		
D8	LCD_WR_RWN	LCDIF	I/O	LCD Interface Data Write/Read-Write Strobe		
B6	LINE_LEFT	LineOut	A	Line Out Left		
D7	LINE_RIGHT	LineOut	A	Line Out Right		
C7	LINE_VAG	LineOut	A	Line Out Reference Capacitor		
C4	LINE1L	ADC	A	Line In 1 Left / DRI_DATA for STFM1000		
D3	LINE1R	ADC	A	Line In 1 Right / DRI_CLK for STFM1000		
A7	LRADC0	LRADC	A	LRADC0 (Button 1 or Temp)		
B7	LRADC1	LRADC	A	LRADC1 (Button 2, Temp or MicBias)/Backlight		
A5	MIC	ADC	A	Microphone Input		
E3	PSWITCH	DCDC	A	Power-On/Recovery/Software-Visible		
G3	PWM0	PWM	I/O	PWM 0	ETM_TSYNC	UART1_RX (Debug)
E2	PWM1	PWM	I/O	PWM 1	ETM_PSA1	UART1_TX (Debug)
D1	PWM2	PWM	I/O	PWM 2	SPDIF	
G4	PWM3	PWM	I/O	PWM 3	ETM_PSA0	UART2_CTS
F4	PWM4	PWM	I/O	PWM 4—16mA Drive for OTG Vbus	ETM_TCLK	UART2_RTS
A3	REFP	ADC	A	ADC Positive Reference Capacitor		
G6	ROTARYA	TIMER	I/O	Rotary Encoder A	UART1_TX (Debug)	JTAG_TDO
G7	ROTARYB	TIMER	I/O	Rotary Encoder B	UART1_RX (Debug)	JTAG_TDI
B4	RTC_XTALI	RTC	A	32.768 kHz Xtal In		
C2	RTC_XTALO	RTC	A	32.768 kHz Xtal Out		
G2	SSP1_CMD	SSP	I/O	SD/MMC CMD, MS BS, SPI MOSI		
F3	SSP1_DATA0	SSP	I/O	SD/MMC/MS Data0, SPI MISO		
H2	SSP1_DATA1	SSP	I/O	SD/MMC/MS Data1	GPML_CE2n	JTAG_TCLK
H4	SSP1_DATA2	SSP	I/O	SD/MMC/MS Data2	UART1_CTS (Debug)	JTAG_RTCK
H1	SSP1_DATA3	SSP	I/O	SD/MMC/MS Data3, SPI Slave Select 0	UART1_RTS (Debug)	JTAG_TMS
G1	SSP1_DETECT	SSP	I/O	Removable Card Detect	ETM_PSA2	USB_OTG_ID
K1	SSP1_SCK	SSP	I/O	SSP Serial Clock		
B2	TESTMODE	SYSTEM	A	Test Mode Pin		
A1	USB_DM	USB	A	USB Negative Data Line		
A2	USB_DP	USB	A	USB Positive Data Line		
D2	VAG	HP	A	Analog Reference Capacitor		
A6	VDD5V	POWER	P	5V Power Input		
E7	VDDA_B1	POWER	A	Analog Power1		
E6	VDDD_B1	POWER	P	Digital Core Power 1		

STMP3770



Table 1148. 100-Pin BGA Pin Definitions by Pin Name (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
F8	VDDD_B2	POWER	P	Digital Core Power 2		
K2	VDDIO18_B1	POWER	P	Digital I/O 1.8V Power 1		
F6	VDDIO18_B2	POWER	P	Digital I/O 1.8V Power 2		
G8	VDDIO33_B1	POWER	P	Digital I/O 3.3V Power 1		
F1	VDDIO33_B3	POWER	P	Digital I/O 3.3V Power 3		
C1	VDDXTAL	CLOCK	A	Crystal Power Filter Cap		
B5	VSSA_B1	POWER	A	Analog Ground 1		
C3	VSSA_B2	POWER	A	Analog Ground 2		
J4	VSSD_B2	POWER	P	Digital Ground 2		
J5	VSSD_B3	POWER	P	Digital Ground 3		
J10	VSSD_B4	POWER	P	Digital Ground 4		
E8	VSSD_B5	POWER	P	Digital Ground 5		
E5	VSSD_B6	POWER	P	Digital Ground 6		
A4	XTALI	CLOCK	A	Crystal In—24 MHz		
B3	XTALO	CLOCK	A	Crystal Out—24 MHz		

Table 1149. 100-Pin BGA Pin Definitions by Pin Number

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
A1	USB_DM	USB	A	USB Negative Data Line		
A2	USB_DP	USB	A	USB Positive Data Line		
A3	REFP	ADC	A	ADC Positive Reference Capacitor		
A4	XTALI	CLOCK	A	Crystal In—24 MHz		
A5	MIC	ADC	A	Microphone Input		
A6	VDD5V	POWER	P	5V Power Input		
A7	LRADC0	LRADC	A	LRADC0 (Button 1 or Temp)		
A8	BATT	POWER	P	Battery Input		
A9	DCDC_LN1	DCDC	A	DCDC Inductor N 1		
A10	DCDC_GND	DCDC	A	DCDC Ground		
B1	DEBUG	SYSTEM	I/O	Serial JTAG Debug Port		
B2	TESTMODE	SYSTEM	A	Test Mode Pin		
B3	XTALO	CLOCK	A	Crystal Out—24 MHz		
B4	RTC_XTALI	RTC	A	32.768 kHz Xtal In		
B5	VSSA_B1	POWER	A	Analog Ground 1		
B6	LINE_LEFT	LineOut	A	Line Out Left		
B7	LRADC1	LRADC	A	LRADC1 (Button 2, Temp or MicBias)/Backlight		
B8	DCDC_BATT	DCDC	A	DCDC Battery		
B9	DCDC_LN2	DCDC	A	DCDC Inductor N 2		
B10	DCDC_VDDIO	DCDC	A	DCDC I/O Power		
C1	VDDXTAL	CLOCK	A	Crystal Power Filter Cap		
C2	RTC_XTALO	RTC	A	32.768 kHz Xtal Out		
C3	VSSA_B2	POWER	A	Analog Ground 2		
C4	LINE1L	ADC	A	Line In 1 Left / DRI_DATA for STFM1000		
C5	HPL	HP	A	Headphone Left		
C6	HPR	HP	A	Headphone Right		
C7	LINE_VAG	LineOut	A	Line Out Reference Capacitor		

Table 1149. 100-Pin BGA Pin Definitions by Pin Number (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
C8	DCDC_LP	DCDC	A	DCDC Inductor P		
C9	DCDC_VDDA	DCDC	A	DCDC Analog Power		
C10	DCDC_VDDD	DCDC	A	DCDC Digital Core Power		
D1	PWM2	PWM	I/O	PWM 2	SPDIF	
D2	VAG	HP	A	Analog Reference Capacitor		
D3	LINE1R	ADC	A	Line In 1 Right / DRI_CLK for STFM1000		
D4	I2C_SDA	I ² C	I/O	I ² C Serial Data	GPMI_CE3n	
D5	HP_VGND	HP	A	Direct-Coupled Headphone Virtual Ground		
D6	HP_SENSE	HP	A	Direct-Coupled Headphone Sense		
D7	LINE_RIGHT	LineOut	A	Line Out Right		
D8	LCD_WR_RWN	LCDIF	I/O	LCD Interface Data Write/Read-Write Strobe		
D9	LCD_RD_E	LCDIF	I/O	LCD Interface Data Read or Enable		
D10	LCD_RESET	LCDIF	I/O	LCD Interface Reset Out		
E1	GPMI_RDY3	GPMI	P	NAND3 Ready/Busy #	LCD_HSYNC	UART2_TX
E2	PWM1	PWM	I/O	PWM 1	ETM_PSA1	UART1_TX (Debug)
E3	PSWITCH	DCDC	A	Power-On/Recovery/Software-Visible		
E4	I2C_SCL	I ² C	I/O	I ² C Serial Clock	GPMI_RDY3	
E5	VSSD_B6	POWER	P	Digital Ground 6		
E6	VDDD_B1	POWER	P	Digital Core Power 1		
E7	VDDA_B1	POWER	A	Analog Power1		
E8	VSSD_B5	POWER	P	Digital Ground 5		
E9	LCD_RS	LCDIF	I/O	LCD Interface Register Select/LCD CCIR Clock		
E10	LCD_D07	LCDIF	I/O	LCD Interface Data 7	ETM_DA7	
F1	VDDIO33_B3	POWER	P	Digital I/O 3.3V Power 3		
F2	GPMI_CE3N	GPMI	P	NAND 3 Chip Enable	LCD_DOTCLK	UART2_RX
F3	SSP1_DATA0	SSP	I/O	SD/MMC/MS Data0, SPI MISO		
F4	PWM4	PWM	I/O	PWM 4—16mA Drive for OTG Vbus	ETM_TCLK	UART2_RTS
F5	GPMI_CE2N	GPMI	P	NAND2 Chip Enable	LCD_ENABLE	
F6	VDDIO18_B2	POWER	P	Digital I/O 1.8V Power 2		
F7	DCDC_MODE	DCDC	A	DCDC Mode Sense		
F8	VDDD_B2	POWER	P	Digital Core Power 2		
F9	LCD_D06	LCDIF	I/O	LCD Interface Data 6	ETM_DA6	
F10	LCD_D04	LCDIF	I/O	LCD Interface Data 4	ETM_DA4	
G1	SSP1_DETECT	SSP	I/O	Removable Card Detect	ETM_PSA2	USB_OTG_ID
G2	SSP1_CMD	SSP	I/O	SD/MMC CMD, MS BS, SPI MOSI		
G3	PWM0	PWM	I/O	PWM 0	ETM_TSYNC	UART1_RX (Debug)
G4	PWM3	PWM	I/O	PWM 3	ETM_PSA0	UART2_CTS
G5	GPMI_CLE	GPMI	I/O	NAND CLE		
G6	ROTARYA	TIMER	I/O	Rotary Encoder A	UART1_TX (Debug)	JTAG_TDO
G7	ROTARYB	TIMER	I/O	Rotary Encoder B	UART1_RX (Debug)	JTAG_TDI
G8	VDDIO33_B1	POWER	P	Digital I/O 3.3V Power 1		
G9	LCD_D05	LCDIF	I/O	LCD Interface Data 5	ETM_DA5	
G10	LCD_D03	LCDIF	I/O	LCD Interface Data 3	ETM_DA3	
H1	SSP1_DATA3	SSP	I/O	SD/MMC/MS Data3, SPI Slave Select 0	UART1_RTS (Debug)	JTAG_TMS
H2	SSP1_DATA1	SSP	I/O	SD/MMC/MS Data1	GPMI_CE2n	JTAG_TCLK

STMP3770

Table 1149. 100-Pin BGA Pin Definitions by Pin Number (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
H3	GPMI_D04	GPMI	I/O	NAND Data 4		SSP2_DATA4
H4	SSP1_DATA2	SSP	I/O	SD/MMC/MS Data2	UART1_CTS (Debug)	JTAG_RTCK
H5	GPMI_D05	GPMI	I/O	NAND Data 5		SSP2_DATA5
H6	EMI_CE3N	EMI	I/O	EMI CE3n	GPMI_CE0n	
H7	GPMI_ALE	GPMI	I/O	NAND ALE		
H8	LCD_D01	LCDIF	I/O	LCD Interface Data 1	ETM_DA1	
H9	LCD_D02	LCDIF	I/O	LCD Interface Data 2	ETM_DA2	
H10	LCD_D00	LCDIF	I/O	LCD Interface Data 0	ETM_DA0	
J1	GPMI_D00	GPMI	I/O	NAND Data 0		
J2	GPMI_D01	GPMI	I/O	NAND Data 1		SSP2_DATA1
J3	GPMI_D03	GPMI	I/O	NAND Data 3		SSP2_DATA3
J4	VSSD_B2	POWER	P	Digital Ground 2		
J5	VSSD_B3	POWER	P	Digital Ground 3		
J6	EMI_CE2N	EMI	I/O	EMI CE2n	GPMI_CE1n	SSP2_DATA0
J7	GPMI_RDY2	GPMI	I/O	NAND2 Ready/Busy#	UART2_RX	SSP2_CMD
J8	GPMI_RDY1	GPMI	I/O	NAND1 Ready/Busy#	UART2_TX	
J9	LCD_CS	LCDIF	I/O	LCD Interface Chip Select.		
J10	VSSD_B4	POWER	P	Digital Ground 4		
K1	SSP1_SCK	SSP	I/O	SSP Serial Clock		
K2	VDDIO18_B1	POWER	P	Digital I/O 1.8V Power 1		
K3	GPMI_WRN	GPMI	I/O	NAND Write Strobe		
K4	GPMI_D02	GPMI	I/O	NAND Data 2		SSP2_DATA2
K5	GPMI_D06	GPMI	I/O	NAND Data 6		SSP2_DATA6
K6	GPMI_D07	GPMI	I/O	NAND Data 7		SSP2_DATA7
K7	GPMI_RDN	GPMI	I/O	NAND Read Strobe		
K8	GPMI_RDY0	GPMI	I/O	NAND0 Ready/Busy#		SSP2_DETECT
K9	GPMI_WP	GPMI	I/O	NAND Write Protect, Renesas Reset		JTAG_TRST
K10	LCD_BUSY	LCDIF	I/O	LCD Interface Busy	LCD_VSYNC	

Table 1150. 100-Pin BGA Ball Map

	1	2	3	4	5	6	7	8	9	10	
A	USB_DM	USB_DP	REFP	XTALI	MIC	VDD5V	LRADC0	BATT	DCDC_LN1	DCDC_GND	A
B	DEBUG	TESTMODE	XTALO	RTC_XTALI	VSSA_B1	LINE_LEFT	LRADC1	DCDC_BATT	DCDC_LN2	DCDC_VDDIO	B
C	VDDXTAL	RTC_XTALO	VSSA_B2	LINE1L	HPL	HPR	LINE_VAG	DCDC_LP	DCDC_VDDA	DCDC_VDDD	C
D	PWM2	VAG	LINE1R	I2C_SDA	HP_VGND	HP_SENSE	LINE_RIGHT	LCD_WR_RWN	LDC_RD_E	LCD_RESET	D
E	GPML_RDY3	PWM1	PSWITCH	I2C_SCL	VSSD_B6	VDDD_B1	VDDA_B1	VSSD_B5	LCD_RS	LCD_D07	E
F	VDDIO33_B3	GPML_CE3N	SSP1_DATA0	PWM4	GPML_CE2N	VDDIO18_B2	DCDC_MODE	VDDD_B2	LCD_D06	LCD_D04	F
G	SSP1_DETECT	SSP1_CMD	PWM0	PWM3	GPML_CLE	ROTARYA	ROTARYB	VDDIO33_B1	LCD_D05	LCD_D03	G
H	SSP1_DATA3	SSP1_DATA1	GPML_D04	SSP1_DATA2	GPML_D05	EMI_CE3N	GPML_ALE	LCD_D01	LCD_D02	LCD_D00	H
J	GPML_D00	GPML_D01	GPML_D03	VSSD_B2	VSSD_B3	EMI_CE2N	GPML_RDY2	GPML_RDY1	LCD_CS	VSSD_B4	J
K	SSP1_SCK	VDDIO18_B1	GPML_WRN	GPML_D02	GPML_D06	GPML_D07	GPML_RDN	GPML_RDY0	GPML_WP	LCD_BUSY	K
	1	2	3	4	5	6	7	8	9	10	

32.2. Pin Definitions for 100-Pin LQFP

This section includes the following pin information for the 100-pin LQFP package:

- [Table 1151, “100-Pin LQFP Pin Definitions by Pin Name,” on page 922](#)
- [Table 1152, “100-Pin LQFP Pin Definitions by Pin Number,” on page 924](#)
- [Table 1153, “100-Pin LQFP Connection Diagram—Top View,” on page 927](#)

Table 1151. 100-Pin LQFP Pin Definitions by Pin Name

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
54	BATT	POWER	P	Battery Input		
52	DCDC_BATT	DCDC	A	DCDC Battery		
50	DCDC_GND	DCDC	A	DCDC Ground		
49	DCDC_LN1	DCDC	A	DCDC Inductor N 1		
46	DCDC_LN2	DCDC	A	DCDC Inductor N 2		
51	DCDC_LP	DCDC	A	DCDC Inductor P		
63	DCDC_MODE	DCDC	A	DCDC Mode Sense		
48	DCDC_VDDA	DCDC	A	DCDC Analog Power		
45	DCDC_VDDD	DCDC	A	DCDC Digital Core Power		
47	DCDC_VDDIO	DCDC	A	DCDC I/O Power		
79	DEBUG	SYSTEM	I/O	Serial JTAG Debug Port		
14	EMI_CE2N	EMI	I/O	EMI CE2n	GPMI_CE1n	SSP2_DATA0
15	EMI_CE3N	EMI	I/O	EMI CE3n	GPMI_CE0n	
17	GPMI_ALE	GPMI	I/O	NAND CLE		
1	GPMI_CE2N	GPMI	I/O	NAND2 Chip Enable		
89	GPMI_CE3N	GPMI	I/O	NAND3 Chip Enable		
18	GPMI_CLE	GPMI	I/O	NAND ALE		
4	GPMI_D00	GPMI	I/O	NAND Data 0		
5	GPMI_D01	GPMI	I/O	NAND Data 1		SSP2_DATA1
6	GPMI_D02	GPMI	I/O	NAND Data 2		SSP2_DATA2
7	GPMI_D03	GPMI	I/O	NAND Data 3		SSP2_DATA3
8	GPMI_D04	GPMI	I/O	NAND Data 4		SSP2_DATA4
9	GPMI_D05	GPMI	I/O	NAND Data 5		SSP2_DATA5
10	GPMI_D06	GPMI	I/O	NAND Data 6		SSP2_DATA6
12	GPMI_D07	GPMI	I/O	NAND Data 7		SSP2_DATA7
16	GPMI_RDN	GPMI	I/O	NAND Read Strobe		
22	GPMI_RDY0	GPMI	I/O	NAND0 Ready/Busy#		SSP2_DETECT
23	GPMI_RDY1	GPMI	I/O	NAND1 Ready/Busy#	UART2_TX	SSP2_SCK
21	GPMI_RDY2	GPMI	I/O	NAND2 Ready/Busy#	UART2_RX	SSP2_CMD
86	GPMI_RDY3	GPMI	I/O	NAND3 Ready/Busy #	LCD_HSYNC	UART2_TX
26	GPMI_WP	GPMI	I/O	NAND Write Protect, Renesas Reset		JTAG_TRST
3	GPMI_WRN	GPMI	I/O	NAND Write Strobe		
62	HP_SENSE	HP	A	Direct-Coupled Headphone Sense		
59	HP_VGND	HP	A	Direct-Coupled Headphone Virtual Ground		
61	HPL	HP	A	Headphone Left		

Table 1151. 100-Pin LQFP Pin Definitions by Pin Name (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
57	HPR	HP	A	Headphone Right		
81	I2C_SCL	I ² C	I/O	I ² C Serial Clock	GPMI_RDY3	
80	I2C_SDA	I ² C	I/O	I ² C Serial Data	GPMI_CE3n	
27	LCD_BUSY	LCDIF	I/O	LCD Interface Busy	LCD_VSYNC	
28	LCD_CS	LCDIF	I/O	LCD Interface Chip Select.		
29	LCD_D00	LCDIF	I/O	LCD Interface Data 0	ETM_DA0	
30	LCD_D01	LCDIF	I/O	LCD Interface Data 1	ETM_DA1	
31	LCD_D02	LCDIF	I/O	LCD Interface Data 2	ETM_DA2	
32	LCD_D03	LCDIF	I/O	LCD Interface Data 3	ETM_DA3	
35	LCD_D04	LCDIF	I/O	LCD Interface Data 4	ETM_DA4	
36	LCD_D05	LCDIF	I/O	LCD Interface Data 5	ETM_DA5	
37	LCD_D06	LCDIF	I/O	LCD Interface Data 6	ETM_DA6	
38	LCD_D07	LCDIF	I/O	LCD Interface Data 7	ETM_DA7	
44	LCD_RD_E	LCDIF	I/O	LCD Interface Data Read or Enable		
42	LCD_RESET	LCDIF	I/O	LCD Interface Reset Out		
41	LCD_RS	LCDIF	I/O	LCD Interface Register Select/LCD CCIR Clock		
43	LCD_WR_RWN	LCDIF	I/O	LCD Interface Data Write/Read-Write Strobe		
65	LINE1L	ADC	A	Line In 1 Left / DRI_DATA for STFM1000		
64	LINE1R	ADC	A	Line In 1 Right / DRI_CLK for STMP1000		
56	LRADC0	LRADC	A	LRADC0 (Button 1 or Temp)		
55	LRADC1	LRADC	A	LRADC1 (Button 2, Temp or MicBias)/Backlight		
66	MIC	ADC	A	Microphone Input		
70	PSWITCH	DCDC	A	Power-On/Recovery/Software-Visible		
91	PWM0	PWM	I/O	PWM 0	ETM_TSYNC	UART1_RX (Debug)
87	PWM1	PWM	I/O	PWM 1	ETM_PSA1	UART1_TX (Debug)
83	PWM2	PWM	I/O	PWM 2	SPDIF	
84	PWM3	PWM	I/O	PWM 3	ETM_PSA0	UART2_CTS
82	PWM4	PWM	I/O	PWM 4—16mA Drive for OTG Vbus	ETM_TCLK	UART2_RTS
68	REFP	ADC	A	ADC Positive Reference Capacitor		
24	ROTARYA	TIMER	I/O	Rotary Encoder A	UART1_TX (Debug)	JTAG_TDO
25	ROTARYB	TIMER	I/O	Rotary Encoder B	UART1_RX (Debug)	JTAG_TDI
72	RTC_XTALI	RTC	A	32.768 kHz Xtal In		
71	RTC_XTALO	RTC	A	32.768 kHz Xtal Out		
94	SSP1_CMD	SSP	I/O	SD/MMC CMD, MS BS, SPI MOSI		
92	SSP1_DATA0	SSP	I/O	SD/MMC/MS Data0, SPI MISO		
98	SSP1_DATA1	SSP	I/O	SD/MMC/MS Data1	GPMI_CE2n	JTAG_TCLK
95	SSP1_DATA2	SSP	I/O	SD/MMC/MS Data2	UART1_CTS (Debug)	JTAG_RTCK
96	SSP1_DATA3	SSP	I/O	SD/MMC/MS Data3, SPI Slave Select 0	UART1_RTS (Debug)	JTAG_TMS
93	SSP1_DETECT	SSP	I/O	Removable Card Detect	ETM_PSA2	USB_OTG_ID
100	SSP1_SCK	SSP	I/O	SSP Serial Clock		
76	TESTMODE	SYSTEM	A	Test Mode Pin		
78	USB_DM	USB	A	USB Negative Data Line		

STMP3770

Table 1151. 100-Pin LQFP Pin Definitions by Pin Name (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
77	USB_DP	USB	A	USB Positive Data Line		
67	VAG	HP	A	Analog Reference Capacitor		
53	VDD5V	POWER	P	5V Power Input		
58	VDDA1	POWER	A	Analog Power1		
11	VDDD1	POWER	P	Digital Core Power 1		
40	VDDD2	POWER	P	Digital Core Power 2		
88	VDDD3	POWER	P	Digital Core Power 3		
2	VDDIO18_1	POWER	P	Digital I/O 1.8V Power 1		
19	VDDIO18_2	POWER	P	Digital I/O 1.8V Power 2		
34	VDDIO33_1	POWER	P	Digital I/O 3.3V Power 1		
97	VDDIO33_2	POWER	P	Digital I/O 3.3 Power 2		
99	VDDIO33_2_S	POWER	P	Shared Digital I/O 3.3V Power 2		
73	VDDXTAL	CLOCK	A	Crystal Power Filter Cap		
60	VSSA1	POWER	A	Analog Ground 1		
69	VSSA2	POWER	A	Analog Ground 2		
13	VSSD2	POWER	P	Digital Ground 2		
20	VSSD3	POWER	P	Digital Ground 3		
33	VSSD4	POWER	P	Digital Ground 4		
39	VSSD5	POWER	P	Digital Ground 5		
90	VSSD6	POWER	P	Digital Ground 6		
85	VSSD8	POWER	P	Digital Ground		
75	XTALI	CLOCK	A	Crystal In—24 MHz		
74	XTALO	CLOCK	A	Crystal Out—24 MHz		

Table 1152. 100-Pin LQFP Pin Definitions by Pin Number

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
1	GPMI_CE2N	GPMI	I/O	NAND2 Chip Enable		
2	VDDIO18_1	POWER	P	Digital I/O 1.8V Power 1		
3	GPMI_WRN	GPMI	I/O	NAND Write Strobe		
4	GPMI_D00	GPMI	I/O	NAND Data 0		
5	GPMI_D01	GPMI	I/O	NAND Data 1		SSP2_DATA1
6	GPMI_D02	GPMI	I/O	NAND Data 2		SSP2_DATA2
7	GPMI_D03	GPMI	I/O	NAND Data 3		SSP2_DATA3
8	GPMI_D04	GPMI	I/O	NAND Data 4		SSP2_DATA4
9	GPMI_D05	GPMI	I/O	NAND Data 5		SSP2_DATA5
10	GPMI_D06	GPMI	I/O	NAND Data 6		SSP2_DATA6
11	VDDD1	POWER	P	Digital Core Power 1		
12	GPMI_D07	GPMI	I/O	NAND Data 7		SSP2_DATA7
13	VSSD2	POWER	P	Digital Ground 2		
14	EMI_CE2N	EMI	I/O	EMI CE2n	GPMI_CE1n	SSP2_DATA0
15	EMI_CE3N	EMI	I/O	EMI CE3n	GPMI_CE0n	
16	GPMI_RDN	GPMI	I/O	NAND Read Strobe		

Table 1152. 100-Pin LQFP Pin Definitions by Pin Number (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
17	GPML_ALE	GPML	I/O	NAND CLE		
18	GPML_CLE	GPML	I/O	NAND ALE		
19	VDDIO18_2	POWER	P	Digital I/O 1.8V Power 2		
20	VSSD3	POWER	P	Digital Ground 3		
21	GPML_RDY2	GPML	I/O	NAND2 Ready/Busy#	UART2_RX	SSP2_CMD
22	GPML_RDY0	GPML	I/O	NAND0 Ready/Busy#		SSP2_DETECT
23	GPML_RDY1	GPML	I/O	NAND1 Ready/Busy#	UART2_TX	SSP2_SCK
24	ROTARYA	TIMER	I/O	Rotary Encoder A	UART1_TX (Debug)	JTAG_TDO
25	ROTARYB	TIMER	I/O	Rotary Encoder B	UART1_RX (Debug)	JTAG_TDI
26	GPML_WP	GPML	I/O	NAND Write Protect, Renesas Reset		JTAG_TRST
27	LCD_BUSY	LCDIF	I/O	LCD Interface Busy	LCD_VSYNC	
28	LCD_CS	LCDIF	I/O	LCD Interface Chip Select.		
29	LCD_D00	LCDIF	I/O	LCD Interface Data 0	ETM_DA0	
30	LCD_D01	LCDIF	I/O	LCD Interface Data 1	ETM_DA1	
31	LCD_D02	LCDIF	I/O	LCD Interface Data 2	ETM_DA2	
32	LCD_D03	LCDIF	I/O	LCD Interface Data 3	ETM_DA3	
33	VSSD4	POWER	P	Digital Ground 4/5		
34	VDDIO33_1	POWER	P	Digital I/O 3.3V Power 1		
35	LCD_D04	LCDIF	I/O	LCD Interface Data 4	ETM_DA4	
36	LCD_D05	LCDIF	I/O	LCD Interface Data 5	ETM_DA5	
37	LCD_D06	LCDIF	I/O	LCD Interface Data 6	ETM_DA6	
38	LCD_D07	LCDIF	I/O	LCD Interface Data 7	ETM_DA7	
39	VSSD5	POWER	P	Digital Ground 5		
40	VDDD2	POWER	P	Digital Core Power 2		
41	LCD_RS	LCDIF	I/O	LCD Interface Register Select/LCD CCIR Clock		
42	LCD_RESET	LCDIF	I/O	LCD Interface Reset Out		
43	LCD_WR_RWN	LCDIF	I/O	LCD Interface Data Write/Read-Write Strobe		
44	LCD_RD_E	LCDIF	I/O	LCD Interface Data Read or Enable		
45	DCDC_VDDD	DCDC	A	DCDC Digital Core Power		
46	DCDC_LN2	DCDC	A	DCDC Inductor N 2		
47	DCDC_VDDIO	DCDC	A	DCDC I/O Power		
48	DCDC_VDDA	DCDC	A	DCDC Analog Power		
49	DCDC_LN1	DCDC	A	DCDC Inductor N 1		
50	DCDC_GND	DCDC	A	DCDC Ground		
51	DCDC_LP	DCDC	A	DCDC Inductor P		
52	DCDC_BATT	DCDC	A	DCDC Battery		
53	VDD5V	POWER	P	5V Power Input		
54	BATT	POWER	P	Battery Input		
55	LRADC1	LRADC	A	LRADC1 (Button 2, Temp or MicBias)/Backlight		
56	LRADC0	LRADC	A	LRADC0 (Button 1 or Temp)		
57	HPR	HP	A	Headphone Right		
58	VDDA1	POWER	A	Analog Power1		

STMP3770



Table 1152. 100-Pin LQFP Pin Definitions by Pin Number (Continued)

Number	Pin Name	Group	Type	Description 1	Description 2	Description 3
59	HP_VGND	HP	A	Direct-Coupled Headphone Virtual Ground		
60	VSSA1	POWER	A	Analog Ground 1		
61	HPL	HP	A	Headphone Left		
62	HP_SENSE	HP	A	Direct-Coupled Headphone Sense		
63	DCDC_MODE	DCDC	A	DCDC Mode Sense		
64	LINE1R	ADC	A	Line In 1 Right / DRI_CLK for STFM1000		
65	LINE1L	ADC	A	Line In 1 Left / DRI_DATA for STFM1000		
66	MIC	ADC	A	Microphone Input		
67	VAG	HP	A	Analog Reference Capacitor		
68	REFP	ADC	A	ADC Positive Reference Capacitor		
69	VSSA2	POWER	A	Analog Ground 2		
70	PSWITCH	DCDC	A	Power-On/Recovery/Software-Visible		
71	RTC_XTALO	RTC	A	32.768 kHz Xtal Out		
72	RTC_XTALI	RTC	A	32.768 kHz Xtal In		
73	VDDXTAL	CLOCK	A	Crystal Power Filter Cap		
74	XTALO	CLOCK	A	Crystal Out—24 MHz		
75	XTALI	CLOCK	A	Crystal In—24 MHz		
76	TESTMODE	SYSTEM	A	Test Mode Pin		
77	USB_DP	USB	A	USB Positive Data Line		
78	USB_DM	USB	A	USB Negative Data Line		
79	DEBUG	SYSTEM	I/O	Serial JTAG Debug Port		
80	I2C_SDA	I ² C	I/O	I ² C Serial Data	GPMI_CE3n	
81	I2C_SCL	I ² C	I/O	I ² C Serial Clock	GPMI_RDY3	
82	PWM4	PWM	I/O	PWM 4—16mA Drive for OTG Vbus	ETM_TCLK	UART2_RTS
83	PWM2	PWM	I/O	PWM 2	SPDIF	
84	PWM3	PWM	I/O	PWM 3	ETM_PSA0	UART2_CTS
85	VSSD8	POWER	P	Digital Ground 8		
86	GPMI_RDY3	GPMI	I/O	NAND3 Ready/Busy#	LCD_HSYNC	UART2_TX
87	PWM1	PWM	I/O	PWM 1	ETM_PSA1	UART1_TX (Debug)
88	VDDD3	POWER	P	Digital Core Power 3		
89	GPMI_CE3N	GPMI	I/O	NAND3 Chip Enable		
90	VSSD6	POWER	P	Digital Ground 6		
91	PWM0	PWM	I/O	PWM 0	ETM_TSYNC	UART1_RX (Debug)
92	SSP1_DATA0	SSP	I/O	SD/MMC/MS Data0, SPI MISO		
93	SSP1_DETECT	SSP	I/O	Removable Card Detect	ETM_PSA2	USB_OTG_ID
94	SSP1_CMD	SSP	I/O	SD/MMC CMD, MS BS, SPI MOSI		
95	SSP1_DATA2	SSP	I/O	SD/MMC/MS Data2	UART1_CTS (Debug)	JTAG_RTCK
96	SSP1_DATA3	SSP	I/O	SD/MMC/MS Data3, SPI Slave Select 0	UART1_RTS (Debug)	JTAG_TMS
97	VDDIO33_2	POWER	P	Digital I/O 3.3 Power 2		
98	SSP1_DATA1	SSP	I/O	SD/MMC/MS Data1	GPMI_CE2n	JTAG_TCLK
99	VDDIO33_2_S	POWER	P	Shared Digital I/O 3.3V Power 2		
100	SSP1_SCK	SSP	I/O	SSP Serial Clock		

Table 1153. 100-Pin LQFP Connection Diagram—Top View

		SSP1_SCK	VDDIO33_2_S	SSP1_DATA1	VDDIO33_2	SSP1_DATA3	SSP1_DATA2	SSP1_CMD	SSP1_DETECT	SSP1_DATA0	PWM0	VSSD6	GPMI_CE3N	VDDD3	PWM1	GPMI_RDY3	VSSD8	PWM3	PWM2	PWM4	I2C_SCL	I2C_SDA	DEBUG	USB_DM	USB_DP	TESTMODE	
		100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	
GPMI_CE2N	1																									75	XTALI
VDDIO18_1	2																									74	XTALO
GPMI_WRN	3																									73	VDDXTAL
GPMI_D00	4																									72	RTC_XTALI
GPMI_D01	5																									71	RTC_XTALO
GPMI_D02	6																									70	PSWITCH
GPMI_D03	7																									69	VSSA2
GPMI_D04	8																									68	REFP
GPMI_D05	9																									67	VAG
GPMI_D06	10																									66	MIC
VDDD1	11																									65	LINE1L
GPMI_D07	12																									64	LINE1R
VSSD2	13																									63	DCDC_MODE
EMI_CE2N	14																									62	HP_SENSE
EMI_CE3N	15																									61	HPL
GPMI_RDN	16																									60	VSSA1
GPMI_ALE	17																									59	HP_VGND
GPMI_CLE	18																									58	VDDA1
VDDIO18_2	19																									57	HPR
VSSD3	20																									56	LRADC0
GPMI_RDY2	21																									55	LRADC1
GPMI_RDY0	22																									54	BATT
GPMI_RDY1	23																									53	VDD5V
ROTARYA	24																									52	DCDC_BATT
ROTARYB	25																									51	DCDC_LP
		26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	
		GPMI_WP	LCD_BUSY	LCD_CS	LCD_D00	LCD_D01	LCD_D02	LCD_D03	VSSD4	VDDIO33_1	LCD_D04	LCD_D05	LCD_D06	LCD_D07	VSSD5	VDDD2	LCD_RS	LCD_RESET	LCD_WR_RWN	LCD_RD_E	DCDC_VDDD	DCDC_LN2	DCDC_VDDIO	DCDC_VDDA	DCDC_LN1	DCDC_GND	

STMP3770



33. PIN CONTROL AND GPIO

This chapter describes the pin control and general-purpose input/output (GPIO) pin interface implemented on the STMP3770. It includes sections on pin multiplexing and configuration, including color-coded pin multiplexing tables (see Table 1155), followed by a description of the GPIO interface operation. Programmable registers are described in [Section 33.4](#).

33.1. Overview

The STMP3770 has 48 digital interface pins available on the 100-pin packages. (In the context of this chapter, “digital interface pin” means the 1.8/3.3-V standard digital interface pins. This does *not* include JTAG, TESTMODE, or digital radio interface pins.) The pin control interface on the STMP3770 has the following features:

- All digital pins have selectable output drive strengths as described in [Section 33.2.2.1](#).
- All digital pins have individual 1.8-V and 3.3-V selects as described in [Section 33.2.2.2](#).
- All SSP data pins have selectable 47-K Ω pullup resistors, and SSP command pins have 10-K Ω pullups.
- All primary GPMI RDY/BUSY pins have selectable 10-K Ω pullup resistors.
- All primary GPMI CE pins have selectable 47-K Ω pullup resistors.

33.2. Operation

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 33.2.3](#).

Each digital pin can be independently programmed for 1.8 V or 3.3 V.

Selected pins have pullups that can be configured using register settings.

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 as described in [Section 33.2.2.1](#).

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 113 digital interface 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.

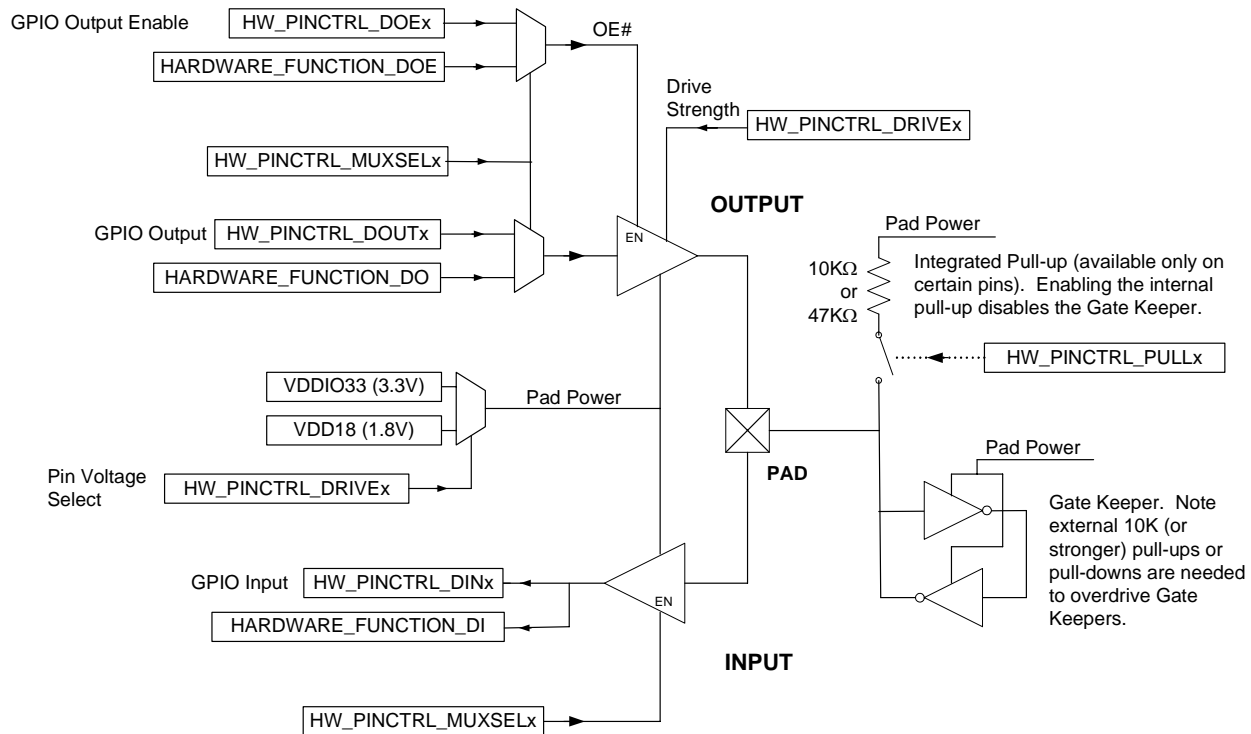


Figure 158. Pad Diagram

33.2.1. Reset Configuration

Out of reset, all pins (with the exception of those required for boot) will be configured as 3.3 V GPIO inputs with gate keepers enabled.

33.2.2. Pin Interface Multiplexing

The STMP3770 is somewhat pin-limited. It contains a rich set of specialized hardware interfaces (SSP, NAND flash, 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 STMP3770 allows many digital pins to be used as GPIOs. 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 STMP3770 specialized hardware interfaces.

Each pin is connected to one, two, or three specialized hardware interfaces, in addition to the GPIO function available on banks 0, 1, and 2. The description of each pin in [Chapter 32, “Pin Descriptions” on page 915](#) contains full details on which specialized hardware interfaces are attached to that pin. For example, pin PWM0 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.

Table 1155 illustrates the pin multiplexing on the STMP3770.

- Table 1154 shows the color mapping used in the table.
- shows the multiplexing used in both 100-pin packages.

Table 1154. Color Mapping for Pin Control Bank Tables

	GPIO	I ² C	JTAG	PWM	SPDIF	Timers and Rotary	Application UART
ETM	GPMI		LCD		SSP	Debug UART	USB

Table 1155. Pin Multiplexing for 100-Pin BGA and 100-Pin LQFP Packages

Bank 0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mux Reg 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
select = 00	gpmi_ce3n	gpmi_ce2n							gpmi_d7	gpmi_d6	gpmi_d5	gpmi_d4	gpmi_d3	gpmi_d2	gpmi_d1	gpmi_d0
select = 01	lcd_dotclk	lcd_enable														
select = 10	uart2_rx								ssp2_d7	ssp2_d6	ssp2_d5	ssp2_d4	ssp2_d3	ssp2_d2	ssp2_d1	
select = 11	GPIO	GPIO							GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
Bank 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mux Reg 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
select = 00									gpmi_rdn	gpmi_wrn	gpmi_irq	gpmi_rstn	gpmi_rb3	gpmi_rb2	gpmi_rb0	
select = 01									uart2_tx		lcd_hsync	uart2_rx				
select = 10									ssp2_sck	jtag_trst_n	uart2_rx	ssp2_cmd	ssp2_det			
select = 11								GPIO	GPIO	GPIO	GPIO	GPIO	GPIO		GPIO	GPIO
Bank 1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mux Reg 2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
select = 00									lcd_d7	lcd_d6	lcd_d5	lcd_d4	lcd_d3	lcd_d2	lcd_d1	lcd_d0
select = 01									etm_da7	etm_da6	etm_da5	etm_da4	etm_da3	etm_da2	etm_da1	etm_da0
select = 10																
select = 11									GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
Bank 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mux Reg 3	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
select = 00							ssp1_det	ssp1_d3	ssp1_d2	ssp1_d1	ssp1_d0	ssp1_sck	ssp1_cmd	lcd_busy	lcd_cs	lcd_rd_e
select = 01							etm_psa2	uart1_rts	uart1_cts	gpmi_ce2n				lcd_vsync		
select = 10							usb_otg_id	jtag_tms	jtag_rtc	jtag_tck						
select = 11							GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
Bank 2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mux Reg 4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
select = 00	emi_ce3n	emi_ce2n							timrot2	timrot1	i2c_sd	i2c_clk	pwm4	pwm3	pwm2	pwm1
select = 01	gpmi_ce0n	gpmi_ce1n							uart1_rx	uart1_tx	gpmi_ce3n	gpmi_rb3	etm_tclk	etm_psa0	spdif	etm_psa1
select = 10			ssp2_d0						jtag_tdi	jtag_tdo			uart2_rts	uart2_cts		uart1_tx
select = 11	GPIO	GPIO							GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO

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, many pins not required for a given active interface can be reused as GPIO pins. For example, the EMI_CE0N pin can be configured and controlled as a GPIO pin.

33.2.2.1. Pin Drive Strength Selection

The drive strength for each digital pin can be programmed by setting the bit corresponding to that pin in one of the HW_PINCTRL_DRIVEx registers. All digital pins have selectable output drive strengths of 4, 8, and 12 mA, with the following exceptions:

- PWM4 has 4, 16, and 20 mA drive strengths.
- SSP1_SCK, GPML_WRN, GPML_RDN, and GPML_IRQ/SSP2_SCK have 8 and 16 mA drive strengths.

See [Table 8, “Drive Strengths Per Pad Type,” on page 46](#) for more information.

Note: The HW_PINCTRL_DRIVEx registers must be configured prior to operation of the pins and cannot be changed mid-course during active operation. Drive-strength options are provided to optimize simultaneous switching output (SSO) noise. The majority of GPIO pins must be programmed in 4-mA mode.

Note: It is recommended that the drive strength of GPML_RDn and GPML_WRn output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND interface read and write cycle times are below 30 ns. The other GPML pins may remain at 4 mA, since their frequency is only up to half that of GPML_RDn and GPML_WRn.

33.2.2.2. Pin Voltage Selection

Each digital pin can be programmed to operate at either 1.8 V or 3.3 V by setting the bit corresponding to that pin in one of the HW_PINCTRL_DRIVEx registers.

Note: The I/O pad driver has two PMOS pullup drivers directly connected to a 1.8-V or 3.3-V power supply. Drive voltage selection for STMP3770 pins is handled in the chip by switching N-well circuits. When the I/O driver is configured in 1.8-V mode, it is not 3.3-V signal-compatible and will cause a DC current flow if the input is driven by a 3.3-V signal.

33.2.2.3. Pullup/Pulldown Selection

Several digital pins can be programmed to enable pullups by setting the appropriate bit in one of the HW_PINCTRL_PULLx registers. Note that enabling the pullup will also disable the internal gate keeper on that pin.

The pullups are tied to the physical pin pad and not the function. So, for example, if the GPML_RDY2 pullup is enabled, that pullup will be present on the GPML_RDY2 pin regardless of what function (GPML_RDY2, UART2_RX, or SSP2_Command) is pin multiplexed out of that pin.

Table 1156. STMP3770 Pins with Internal Pullup Resistors

PIN NAME	TYPE	VALUE
GPMI_RDY2	Pullup	10K
GPMI_D01	Pullup	47K
GPMI_D02	Pullup	47K
GPMI_D03	Pullup	47K
GPMI_D04	Pullup	47K
GPMI_D05	Pullup	47K
GPMI_D06	Pullup	47K
GPMI_D07	Pullup	47K
SSP1_DATA0	Pullup	47K
SSP1_DATA1	Pullup	47K
SSP1_DATA2	Pullup	47K
SSP1_DATA3	Pullup	47K
SSP1_CMD	Pullup	10K
EMI_CE2N	Pullup	47K
GPMI_RDY1	Pullup	10K
GPMI_RDY3	Pullup	10K
GPMI_RDY0	Pullup	10K
GPMI_CE2N	Pullup	47K
GPMI_CE3N	Pullup	47K
EMI_CE3N	Pullup	47K

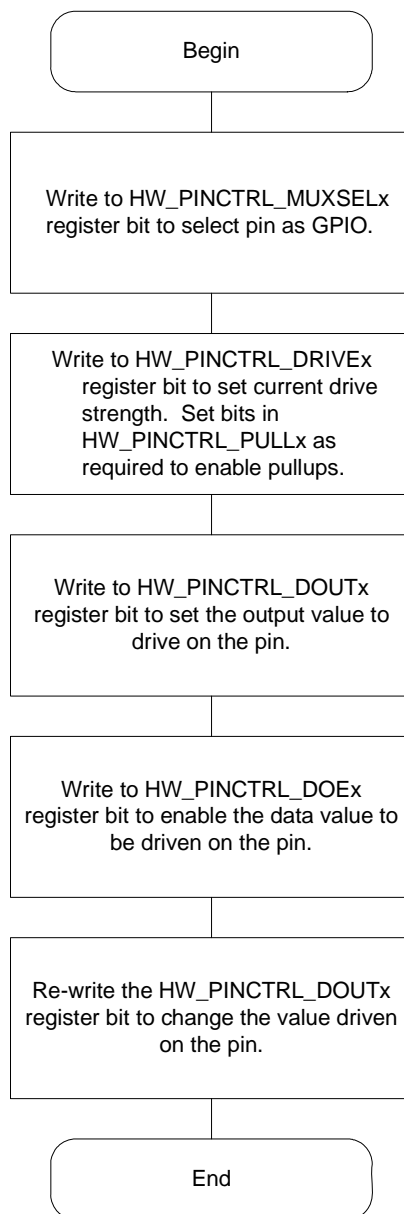
33.2.3. GPIO Interface

The registers discussed in the following sections exist within each of the three GPIO banks to configure the chip's digital pins. Some pins exist in the 169-pin package only. The registers that control those pins exist but perform no useful function when in a 100-pin package.

33.2.3.1. 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 159](#).

- 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 and pin voltage. Set bits in HW_PINCTRL_PULLx as required to enable pullups.
- 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 159. GPIO Output Setup Flowchart**

33.2.3.2. 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 160](#). 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.

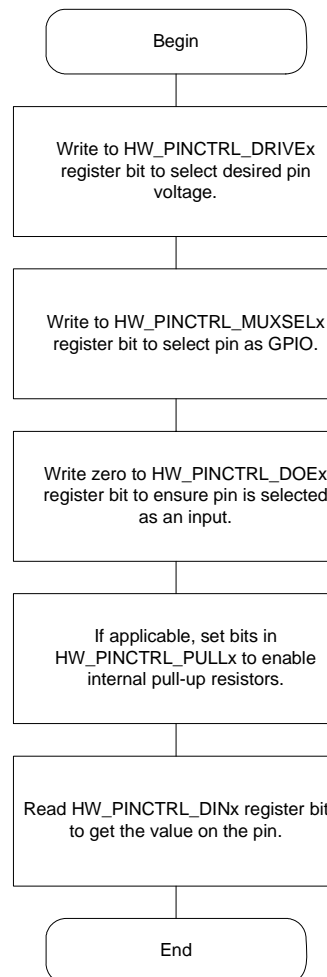


Figure 160. GPIO Input Setup Flowchart

33.2.3.3. 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 161](#).

- 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.

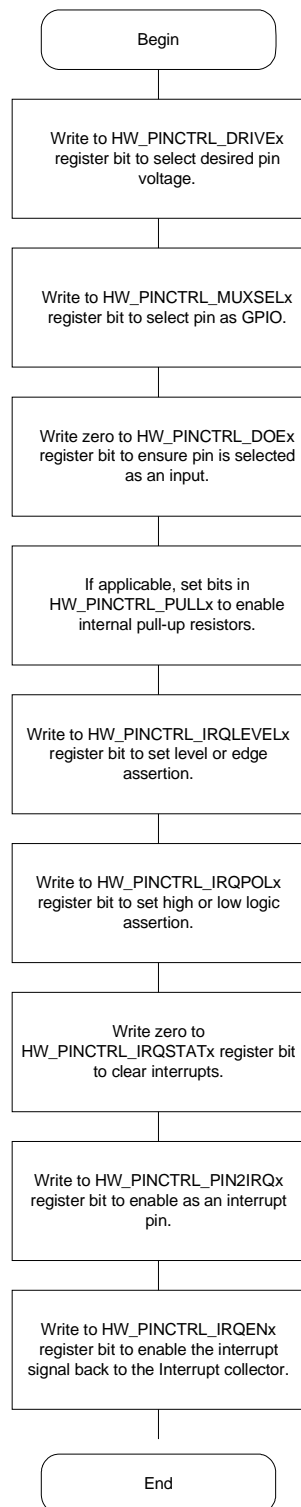
**Figure 161. GPIO Interrupt Flowchart**

Figure 162 shows the logic diagram for the interrupt-generation circuit.

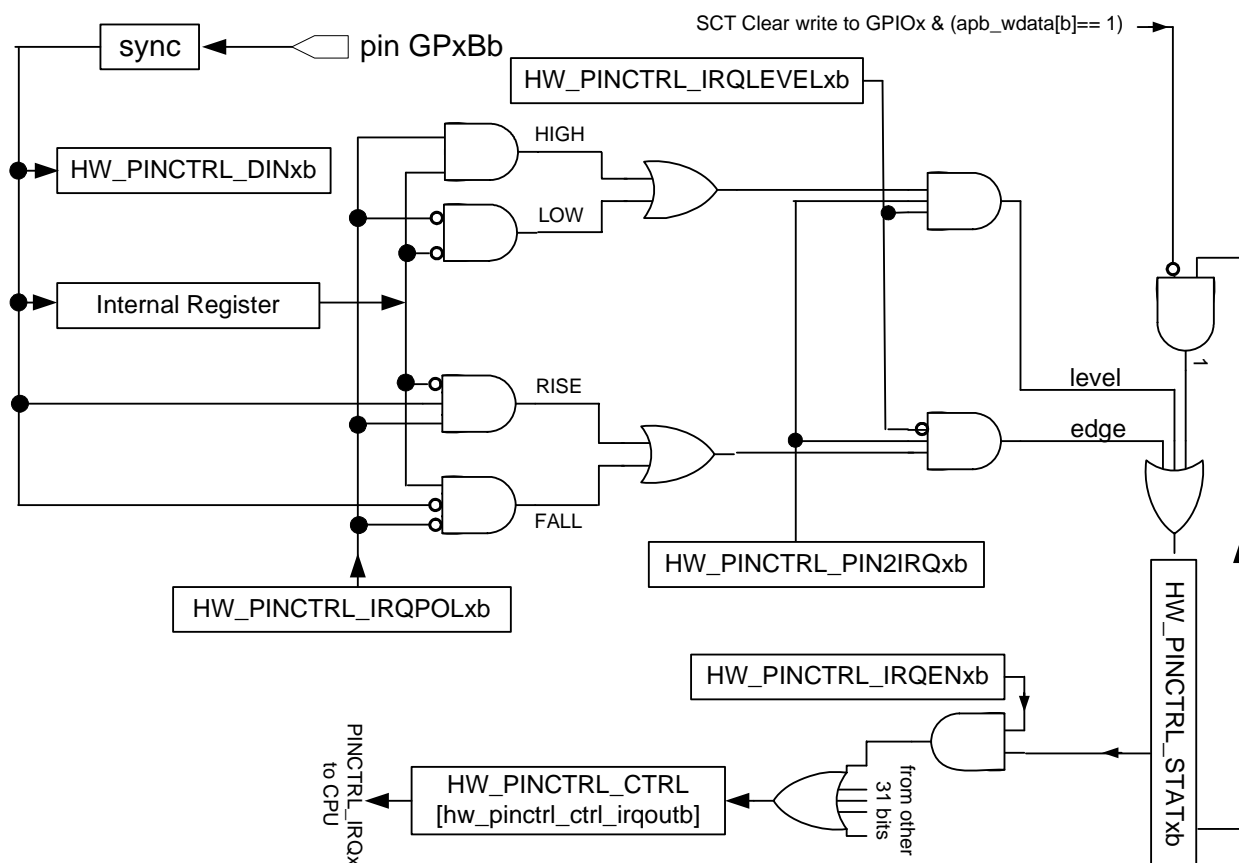


Figure 162. GPIO Interrupt Generation

33.3. 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 34.4.10. “Correct Way to Soft Reset a Block” on page 1013](#), for additional information on using the SFTRST and CLKGATE bit fields.

Table 1158. HW_PINCTRL_CTRL Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	IRQOUT2	RO	0x0	Read-only view of the interrupt collector GPIO2 signal, sourced from the combined IRQ outputs from Bank 2.
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.

33.4.2. PINCTRL Pin Mux Select Register 0 Description

The PINCTRL Pin Mux Select Register 0 provides pin function selection for 16 pins in Bank 0.

HW_PINCTRL_MUXSEL0 0x80018100
 HW_PINCTRL_MUXSEL0_SET 0x80018104
 HW_PINCTRL_MUXSEL0_CLR 0x80018108
 HW_PINCTRL_MUXSEL0_TOG 0x8001810C

Table 1159. HW_PINCTRL_MUXSEL0

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
BANK0_PIN15		BANK0_PIN14		BANK0_PIN13		BANK0_PIN12		BANK0_PIN11		BANK0_PIN10		BANK0_PIN09		BANK0_PIN08		BANK0_PIN07		BANK0_PIN06		BANK0_PIN05		BANK0_PIN04		BANK0_PIN03		BANK0_PIN02		BANK0_PIN01		BANK0_PIN00	

Table 1160. HW_PINCTRL_MUXSEL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	BANK0_PIN15	RW	0x3	GPMI_D15 Pin Function Selection. 00 = GPMI_CE3N. 01 = LCD_DOTCLK. 10 = UART2_RX. 11 = GPIO.
29:28	BANK0_PIN14	RW	0x3	GPMI_D14 Pin Function Selection. 00 = GPMI_CE2N. 01 = LCD_ENABLE. 10 = UART2_RX. 11 = GPIO.
27:26	BANK0_PIN13	RW	0x3	GPMI_D13 Pin Function Selection. 00 = GPMI_D13. 01 = EMI_ADDR20. 10 = GPMI_CE1N. 11 = GPIO.

Table 1160. HW_PINCTRL_MUXSEL0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
25:24	BANK0_PIN12	RW	0x3	GPMI_D12 Pin Function Selection. 00 = GPMI_D12. 01 = EMI_ADDR19. 10 = GPMI_CE0N. 11 = GPIO.
23:22	BANK0_PIN11	RW	0x3	GPMI_D11 Pin Function Selection. 00 = GPMI_D11. 01 = EMI_ADDR18. 10 = Reserved. 11 = GPIO.
21:20	BANK0_PIN10	RW	0x3	GPMI_D10 Pin Function Selection. 00 = GPMI_D10. 01 = EMI_ADDR17. 10 = Reserved. 11 = GPIO.
19:18	BANK0_PIN09	RW	0x3	GPMI_D09 Pin Function Selection. 00 = GPMI_D9. 01 = EMI_ADDR16. 10 = Reserved. 11 = GPIO.
17:16	BANK0_PIN08	RW	0x3	GPMI_D08 Pin Function Selection. 00 = GPMI_D8. 01 = EMI_ADDR15. 10 = Reserved. 11 = GPIO.
15:14	BANK0_PIN07	RW	0x3	GPMI_D07 Pin Function Selection. 00 = GPMI_D7. 01 = Reserved. 10 = SSP2_D7. 11 = GPIO.
13:12	BANK0_PIN06	RW	0x3	GPMI_D06 Pin Function Selection. 00 = GPMI_D6. 01 = Reserved. 10 = SSP2_D6. 11 = GPIO.
11:10	BANK0_PIN05	RW	0x3	GPMI_D05 Pin Function Selection. 00 = GPMI_D5. 01 = Reserved. 10 = SSP2_D5. 11 = GPIO.
9:8	BANK0_PIN04	RW	0x3	GPMI_D04 Pin Function Selection. 00 = GPMI_D4. 01 = Reserved. 10 = SSP2_D4. 11 = GPIO.

STMP3770**Table 1162. HW_PINCTRL_MUXSEL1 Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
31:28	RSVD	RO	0x0	Reserved.
27:26	BANK0_PIN29	RW	0x3	UART2_TX Pin Function Selection. 00 = UART2_TX. 01 = IR_XCVR_DATA_OUT. 10 = SSP1_D7. 11 = GPIO.
25:24	BANK0_PIN28	RW	0x3	UART2_RX Pin Function Selection. 00 = UART2_RX. 01 = IR_XCVR_DATA_IN. 10 = SSP1_D6. 11 = GPIO.
23:22	BANK0_PIN27	RW	0x3	UART2_RTS Pin Function Selection. 00 = UART2_RTS. 01 = IR_XCVR_CLK. 10 = SSP1_D5. 11 = GPIO.
21:20	BANK0_PIN26	RW	0x3	UART2_CTS Pin Function Selection. 00 = UART2_CTS. 01 = RESERVED. 10 = SSP1_D4. 11 = GPIO.
19:18	BANK0_PIN25	RW	0x3	GPML_RDN Pin Function Selection. 00 = GPML_RDN. 01 = Reserved. 10 = Reserved. 11 = GPIO.
17:16	BANK0_PIN24	RW	0x3	GPML_WRN Pin Function Selection. 00 = GPML_WRN. 01 = Reserved. 10 = Reserved. 11 = GPIO.
15:14	BANK0_PIN23	RW	0x3	GPML_IRQ Pin Function Selection. 00 = GPML_IRQ. 01 = UART2_TX. 10 = SSP2_SCK. 11 = GPIO.
13:12	BANK0_PIN22	RW	0x3	GPML_RESETN Pin Function Selection. 00 = GPML_RESETN. 01 = EMI_RESETN. 10 = ALT_JTAG_TRST_N. 11 = GPIO.
11:10	BANK0_PIN21	RW	0x3	GPML_RDY3 Pin Function Selection. 00 = GPML_READY3. 01 = EMI_OEN. 10 = IR_XCVR_DATA_IN. 11 = GPIO.

Table 1162. HW_PINCTRL_MUXSEL1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9:8	BANK0_PIN20	RW	0x3	GPML_RDY2 Pin Function Selection. 00 = GPML_DMACK/GPML_READY2. 01 = UART2_RX. 10 = SSP2_CMD. 11 = GPIO.
7:6	BANK0_PIN19	RW	0x3	GPML_RDY0 Pin Function Selection. 00 = GPML_READY0. 01 = Reserved. 10 = SSP2_DETECT. 11 = GPIO.
5:4	BANK0_PIN18	RW	0x3	GPML_A2 Pin Function Selection. 00 = GPML_A2. 01 = EMI_ADDR25. 10 = IR_XCVR_DATA_OUT. 11 = GPIO.
3:2	BANK0_PIN17	RW	0x3	GPML_A1 Pin Function Selection. 00 = GPML_A1. 01 = EMI_ADDR24. 10 = Reserved. 11 = GPIO.
1:0	BANK0_PIN16	RW	0x3	GPML_A0 Pin Function Selection. 00 = GPML_A0. 01 = EMI_ADDR23. 10 = Reserved. 11 = GPIO.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 14 pins shown above.

33.4.4. PINCTRL Pin Mux Select Register 2 Description

The PINCTRL Pin Mux Select Register 2 provides pin function selection for 16 pins in Bank 1.

HW_PINCTRL_MUXSEL2 0x80018120
 HW_PINCTRL_MUXSEL2_SET 0x80018124
 HW_PINCTRL_MUXSEL2_CLR 0x80018128
 HW_PINCTRL_MUXSEL2_TOG 0x8001812C

Table 1163. HW_PINCTRL_MUXSEL2

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
BANK1_PIN15	BANK1_PIN14	BANK1_PIN13	BANK1_PIN12	BANK1_PIN11	BANK1_PIN10	BANK1_PIN09	BANK1_PIN08	BANK1_PIN07	BANK1_PIN06	BANK1_PIN05	BANK1_PIN04	BANK1_PIN03	BANK1_PIN02	BANK1_PIN01	BANK1_PIN00																

Table 1164. HW_PINCTRL_MUXSEL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	BANK1_PIN15	RW	0x3	LCD_D15 Pin Function Selection. 00 = LCDIF_D15. 01 = ETM_DA7. 10 = LCDIF_VSYNC. 11 = GPIO.
29:28	BANK1_PIN14	RW	0x3	LCD_D14 Pin Function Selection. 00 = LCDIF_D14. 01 = ETM_DA5. 10 = SAIF1_SDATA2. 11 = GPIO.
27:26	BANK1_PIN13	RW	0x3	LCD_D13 Pin Function Selection. 00 = LCDIF_D13. 01 = ETM_DA6. 10 = SAIF2_SDATA2. 11 = GPIO.
25:24	BANK1_PIN12	RW	0x3	LCD_D12 Pin Function Selection. 00 = LCDIF_D12. 01 = ETM_DA4. 10 = SAIF2_SDATA1. 11 = GPIO.
23:22	BANK1_PIN11	RW	0x3	LCD_D11 Pin Function Selection. 00 = LCDIF_D11. 01 = ETM_DA3. 10 = SAIF_LRCLK. 11 = GPIO.
21:20	BANK1_PIN10	RW	0x3	LCD_D10 Pin Function Selection. 00 = LCDIF_D10. 01 = ETM_DA2. 10 = SAIF_BITCLK. 11 = GPIO.
19:18	BANK1_PIN09	RW	0x3	LCD_D09 Pin Function Selection. 00 = LCDIF_D9. 01 = ETM_DA1. 10 = SAIF1_SDATA0. 11 = GPIO.
17:16	BANK1_PIN08	RW	0x3	LCD_D08 Pin Function Selection. 00 = LCDIF_D8. 01 = ETM_DA0. 10 = SAIF2_SDATA0. 11 = GPIO.
15:14	BANK1_PIN07	RW	0x3	LCD_D07 Pin Function Selection. 00 = LCDIF_D7. 01 = ETM_DA7. 10 = Reserved. 11 = GPIO.

Table 1164. HW_PINCTRL_MUXSEL2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
13:12	BANK1_PIN06	RW	0x3	LCD_D06 Pin Function Selection. 00 = LCDIF_D6. 01 = ETM_DA6. 10 = Reserved. 11 = GPIO.
11:10	BANK1_PIN05	RW	0x3	LCD_D05 Pin Function Selection. 00 = LCDIF_D5. 01 = ETM_DA5. 10 = Reserved. 11 = GPIO.
9:8	BANK1_PIN04	RW	0x3	LCD_D04 Pin Function Selection. 00 = LCDIF_D4. 01 = ETM_DA4. 10 = Reserved. 11 = GPIO.
7:6	BANK1_PIN03	RW	0x3	LCD_D03 Pin Function Selection. 00 = LCDIF_D3. 01 = ETM_DA3. 10 = Reserved. 11 = GPIO.
5:4	BANK1_PIN02	RW	0x3	LCD_D02 Pin Function Selection. 00 = LCDIF_D2. 01 = ETM_DA2. 10 = Reserved. 11 = GPIO.
3:2	BANK1_PIN01	RW	0x3	LCD_D01 Pin Function Selection. 00 = LCDIF_D1. 01 = ETM_DA1. 10 = Reserved. 11 = GPIO.
1:0	BANK1_PIN00	RW	0x3	LCD_D00 Pin Function Selection. 00 = LCDIF_D0. 01 = ETM_DA0. 10 = Reserved. 11 = GPIO.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

33.4.5. PINCTRL Pin Mux Select Register 3 Description

The PINCTRL Pin Mux Select Register 3 provides pin function selection for 13 pins in Bank 1.

HW_PINCTRL_MUXSEL3	0x80018130
HW_PINCTRL_MUXSEL3_SET	0x80018134
HW_PINCTRL_MUXSEL3_CLR	0x80018138
HW_PINCTRL_MUXSEL3_TOG	0x8001813C

STMP3770

Table 1165. HW_PINCTRL_MUXSEL3

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0																								
RSVD				BANK1_PIN28				BANK1_PIN27				BANK1_PIN26				BANK1_PIN25				BANK1_PIN24				BANK1_PIN23				BANK1_PIN22				BANK1_PIN21				BANK1_PIN20				BANK1_PIN19				BANK1_PIN18				BANK1_PIN17				BANK1_PIN16			

Table 1166. HW_PINCTRL_MUXSEL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:26	RSVD	RO	0x0	Reserved.
25:24	BANK1_PIN28	RW	0x3	SSP1_DETECT Pin Function Selection. 00 = SSP1_DETECT. 01 = ETM_PSA2. 10 = USB_OTG_ID. 11 = GPIO.
23:22	BANK1_PIN27	RW	0x3	SSP1_DATA3 Pin Function Selection. 00 = SSP1_D3. 01 = UART1_RTS. 10 = ALT_JTAG_TMS. 11 = GPIO.
21:20	BANK1_PIN26	RW	0x3	SSP1_DATA2 Pin Function Selection. 00 = SSP1_D2. 01 = UART1_CTS. 10 = ALT_JTAG_RTCK. 11 = GPIO.
19:18	BANK1_PIN25	RW	0x3	SSP1_DATA1 Pin Function Selection. 00 = SSP1_D1. 01 = GPMI_CE2N. 10 = ALT_JTAG_TCK. 11 = GPIO.
17:16	BANK1_PIN24	RW	0x3	SSP1_DATA0 Pin Function Selection. 00 = SSP1_D0. 01 = Reserved. 10 = Reserved. 11 = GPIO.
15:14	BANK1_PIN23	RW	0x3	SSP1_SCK Pin Function Selection. 00 = SSP1_SCK. 01 = Reserved. 10 = Reserved. 11 = GPIO.
13:12	BANK1_PIN22	RW	0x3	SSP1_CMD Pin Function Selection. 00 = SSP1_CMD. 01 = Reserved. 10 = Reserved. 11 = GPIO.

Table 1166. HW_PINCTRL_MUXSEL3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
11:10	BANK1_PIN21	RW	0x3	LCD_BUSY Pin Function Selection. 00 = LCDIF_BUSY. 01 = LCDIF_VSYNC. 10 = SAIF1_SDATA1. 11 = GPIO.
9:8	BANK1_PIN20	RW	0x3	LCD_CS Pin Function Selection. 00 = LCDIF_CS. 01 = Reserved. 10 = Reserved. 11 = GPIO.
7:6	BANK1_PIN19	RW	0x3	LCD_RD_E Pin Function Selection. 00 = LCDIF_RD_E. 01 = Reserved. 10 = Reserved. 11 = GPIO.
5:4	BANK1_PIN18	RW	0x3	LCD_WR_RWN Pin Function Selection. 00 = LCDIF_WR_RWN. 01 = Reserved. 10 = Reserved. 11 = GPIO.
3:2	BANK1_PIN17	RW	0x3	LCD_RS Pin Function Selection. 00 = LCDIF_REG. 01 = Reserved. 10 = Reserved. 11 = GPIO.
1:0	BANK1_PIN16	RW	0x3	LCD_RESET Pin Function Selection. 00 = LCDIF_RESET. 01 = Reserved. 10 = Reserved. 11 = GPIO.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 13 pins shown above.

33.4.6. PINCTRL Pin Mux Select Register 4 Description

The PINCTRL Pin Mux Select Register 4 provides pin function selection for 16 pins in Bank 2.

HW_PINCTRL_MUXSEL4	0x80018140
HW_PINCTRL_MUXSEL4_SET	0x80018144
HW_PINCTRL_MUXSEL4_CLR	0x80018148
HW_PINCTRL_MUXSEL4_TOG	0x8001814C

Table 1168. HW_PINCTRL_MUXSEL4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:16	BANK2_PIN08	RW	0x3	ROTARYB Pin Function Selection. 00 = TIMROT2. 01 = UART1_RX. 10 = ALT_JTAG_TDI. 11 = GPIO.
15:14	BANK2_PIN07	RW	0x3	ROTARYA Pin Function Selection. 00 = TIMROT1. 01 = UART1_TX. 10 = ALT_JTAG_TDO. 11 = GPIO.
13:12	BANK2_PIN06	RW	0x3	I2C_SDA Pin Function Selection. 00 = I2C_SD. 01 = GPMI_CE3N. 10 = Reserved. 11 = GPIO.
11:10	BANK2_PIN05	RW	0x3	I2C_SCL Pin Function Selection. 00 = I2C_CLK. 01 = GPMI_READY3. 10 = Reserved. 11 = GPIO.
9:8	BANK2_PIN04	RW	0x3	PWM4 Pin Function Selection. 00 = PWM4. 01 = ETM_TCLK. 10 = UART2_RTS. 11 = GPIO.
7:6	BANK2_PIN03	RW	0x3	PWM3 Pin Function Selection. 00 = PWM3. 01 = ETM_PSA0. 10 = UART2_CTS. 11 = GPIO.
5:4	BANK2_PIN02	RW	0x3	PWM2 Pin Function Selection. 00 = PWM2. 01 = SPDIF. 10 = SAIF_ALT_BITCLK. 11 = GPIO.
3:2	BANK2_PIN01	RW	0x3	PWM1 Pin Function Selection. 00 = PWM1. 01 = ETM_PSA1. 10 = UART1_TX. 11 = GPIO.
1:0	BANK2_PIN00	RW	0x3	PWM0 Pin Function Selection. 00 = PWM0. 01 = ETM_TSYNCA. 10 = UART1_RX. 11 = GPIO.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

STMP3770

33.4.7. PINCTRL Pin Mux Select Register 5 Description

The PINCTRL Pin Mux Select Register 5 provides pin function selection for 16 pins in Bank 2.

HW_PINCTRL_MUXSEL5 0x80018150
 HW_PINCTRL_MUXSEL5_SET 0x80018154
 HW_PINCTRL_MUXSEL5_CLR 0x80018158
 HW_PINCTRL_MUXSEL5_TOG 0x8001815C

Table 1169. HW_PINCTRL_MUXSEL5

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
BANK2_PIN31	BANK2_PIN30	BANK2_PIN29	BANK2_PIN28	BANK2_PIN27	BANK2_PIN26	BANK2_PIN25	BANK2_PIN24	BANK2_PIN23	BANK2_PIN22	BANK2_PIN21	BANK2_PIN20	BANK2_PIN19	BANK2_PIN18	BANK2_PIN17	BANK2_PIN16																

Table 1170. HW_PINCTRL_MUXSEL5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	BANK2_PIN31	RW	0x3	EMI_WEN Pin Function Selection. 00 = EMI_WEN. 01 = Reserved. 10 = Reserved. 11 = GPIO.
29:28	BANK2_PIN30	RW	0x3	EMI_A14 Pin Function Selection. 00 = EMI_ADDR14. 01 = Reserved. 10 = Reserved. 11 = GPIO.
27:26	BANK2_PIN29	RW	0x3	EMI_A13 Pin Function Selection. 00 = EMI_ADDR13. 01 = Reserved. 10 = Reserved. 11 = GPIO.
25:24	BANK2_PIN28	RW	0x3	EMI_A12 Pin Function Selection. 00 = EMI_ADDR12. 01 = Reserved. 10 = Reserved. 11 = GPIO.
23:22	BANK2_PIN27	RW	0x3	EMI_A11 Pin Function Selection. 00 = EMI_ADDR11. 01 = Reserved. 10 = Reserved. 11 = GPIO.
21:20	BANK2_PIN26	RW	0x3	EMI_A10 Pin Function Selection. 00 = EMI_ADDR10. 01 = Reserved. 10 = Reserved. 11 = GPIO.

Table 1170. HW_PINCTRL_MUXSEL5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19:18	BANK2_PIN25	RW	0x3	EMI_A09 Pin Function Selection. 00 = EMI_ADDR9. 01 = Reserved. 10 = Reserved. 11 = GPIO.
17:16	BANK2_PIN24	RW	0x3	EMI_A08 Pin Function Selection. 00 = EMI_ADDR8. 01 = Reserved. 10 = Reserved. 11 = GPIO.
15:14	BANK2_PIN23	RW	0x3	EMI_A07 Pin Function Selection. 00 = EMI_ADDR7. 01 = Reserved. 10 = Reserved. 11 = GPIO.
13:12	BANK2_PIN22	RW	0x3	EMI_A06 Pin Function Selection. 00 = EMI_ADDR6. 01 = Reserved. 10 = Reserved. 11 = GPIO.
11:10	BANK2_PIN21	RW	0x3	EMI_A05 Pin Function Selection. 00 = EMI_ADDR5. 01 = Reserved. 10 = Reserved. 11 = GPIO.
9:8	BANK2_PIN20	RW	0x3	EMI_A04 Pin Function Selection. 00 = EMI_ADDR4. 01 = Reserved. 10 = Reserved. 11 = GPIO.
7:6	BANK2_PIN19	RW	0x3	EMI_A03 Pin Function Selection. 00 = EMI_ADDR3. 01 = Reserved. 10 = Reserved. 11 = GPIO.
5:4	BANK2_PIN18	RW	0x3	EMI_A02 Pin Function Selection. 00 = EMI_ADDR2. 01 = Reserved. 10 = Reserved. 11 = GPIO.

STMP3770

Table 1170. HW_PINCTRL_MUXSEL5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
3:2	BANK2_PIN17	RW	0x3	EMI_A01 Pin Function Selection. 00 = EMI_ADDR1. 01 = Reserved. 10 = Reserved. 11 = GPIO.
1:0	BANK2_PIN16	RW	0x3	EMI_A00 Pin Function Selection. 00 = EMI_ADDR0. 01 = Reserved. 10 = Reserved. 11 = GPIO.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

33.4.8. PINCTRL Pin Mux Select Register 6 Description

The PINCTRL Pin Mux Select Register 6 provides pin function selection for 16 pins in Bank 3.

HW_PINCTRL_MUXSEL6 0x80018160
 HW_PINCTRL_MUXSEL6_SET 0x80018164
 HW_PINCTRL_MUXSEL6_CLR 0x80018168
 HW_PINCTRL_MUXSEL6_TOG 0x8001816C

Table 1171. HW_PINCTRL_MUXSEL6

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
BANK3_PIN15		BANK3_PIN14		BANK3_PIN13		BANK3_PIN12		BANK3_PIN11		BANK3_PIN10		BANK3_PIN09		BANK3_PIN08		BANK3_PIN07		BANK3_PIN06		BANK3_PIN05		BANK3_PIN04		BANK3_PIN03		BANK3_PIN02		BANK3_PIN01		BANK3_PIN00	

Table 1172. HW_PINCTRL_MUXSEL6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	BANK3_PIN15	RW	0x3	EMI_D15 Pin Function Selection. 00 = EMI_DATA15. 01 = Reserved. 10 = Reserved. 11 = Disabled.
29:28	BANK3_PIN14	RW	0x3	EMI_D14 Pin Function Selection. 00 = EMI_DATA14. 01 = Reserved. 10 = Reserved. 11 = Disabled.

Table 1172. HW_PINCTRL_MUXSEL6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
27:26	BANK3_PIN13	RW	0x3	EMI_D13 Pin Function Selection. 00 = EMI_DATA13. 01 = Reserved. 10 = Reserved. 11 = Disabled.
25:24	BANK3_PIN12	RW	0x3	EMI_D12 Pin Function Selection. 00 = EMI_DATA12. 01 = Reserved. 10 = Reserved. 11 = Disabled.
23:22	BANK3_PIN11	RW	0x3	EMI_D11 Pin Function Selection. 00 = EMI_DATA11. 01 = Reserved. 10 = Reserved. 11 = Disabled.
21:20	BANK3_PIN10	RW	0x3	EMI_D10 Pin Function Selection. 00 = EMI_DATA10. 01 = Reserved. 10 = Reserved. 11 = Disabled.
19:18	BANK3_PIN09	RW	0x3	EMI_D09 Pin Function Selection. 00 = EMI_DATA9. 01 = Reserved. 10 = Reserved. 11 = Disabled.
17:16	BANK3_PIN08	RW	0x3	EMI_D08 Pin Function Selection. 00 = EMI_DATA8. 01 = Reserved. 10 = Reserved. 11 = Disabled.
15:14	BANK3_PIN07	RW	0x3	EMI_D07 Pin Function Selection. 00 = EMI_DATA7. 01 = Reserved. 10 = Reserved. 11 = Disabled.
13:12	BANK3_PIN06	RW	0x3	EMI_D06 Pin Function Selection. 00 = EMI_DATA6. 01 = Reserved. 10 = Reserved. 11 = Disabled.
11:10	BANK3_PIN05	RW	0x3	EMI_D05 Pin Function Selection. 00 = EMI_DATA5. 01 = Reserved. 10 = Reserved. 11 = Disabled.

Table 1172. HW_PINCTRL_MUXSEL6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
9:8	BANK3_PIN04	RW	0x3	EMI_D04 Pin Function Selection. 00 = EMI_DATA4. 01 = Reserved. 10 = Reserved. 11 = Disabled.
7:6	BANK3_PIN03	RW	0x3	EMI_D03 Pin Function Selection. 00 = EMI_DATA3. 01 = Reserved. 10 = Reserved. 11 = Disabled.
5:4	BANK3_PIN02	RW	0x3	EMI_D02 Pin Function Selection. 00 = EMI_DATA2. 01 = Reserved. 10 = Reserved. 11 = Disabled.
3:2	BANK3_PIN01	RW	0x3	EMI_D01 Pin Function Selection. 00 = EMI_DATA1. 01 = Reserved. 10 = Reserved. 11 = Disabled.
1:0	BANK3_PIN00	RW	0x3	EMI_D00 Pin Function Selection. 00 = EMI_DATA0. 01 = Reserved. 10 = Reserved. 11 = Disabled.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

33.4.9. PINCTRL Pin Mux Select Register 7 Description

The PINCTRL Pin Mux Select Register 7 provides pin function selection for 6 pins in Bank 3.

HW_PINCTRL_MUXSEL7	0x80018170
HW_PINCTRL_MUXSEL7_SET	0x80018174
HW_PINCTRL_MUXSEL7_CLR	0x80018178
HW_PINCTRL_MUXSEL7_TOG	0x8001817C

Table 1173. HW_PINCTRL_MUXSEL7

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD																				BANK3_PIN21		BANK3_PIN20		BANK3_PIN19		BANK3_PIN18		BANK3_PIN17		BANK3_PIN16	

Table 1174. HW_PINCTRL_MUXSEL7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:12	RSVD	RO	0x0	Reserved.
11:10	BANK3_PIN21	RW	0x3	EMI_CLKN Pin Function Selection. 00 = EMI_CLKN. 01 = Reserved. 10 = Reserved. 11 = Disabled.
9:8	BANK3_PIN20	RW	0x3	EMI_CLK Pin Function Selection. 00 = EMI_CLK. 01 = Reserved. 10 = Reserved. 11 = Disabled.
7:6	BANK3_PIN19	RW	0x3	EMI_DQM1 Pin Function Selection. 00 = EMI_DQM1. 01 = Reserved. 10 = Reserved. 11 = Disabled.
5:4	BANK3_PIN18	RW	0x3	EMI_DQM0 Pin Function Selection. 00 = EMI_DQM0. 01 = Reserved. 10 = Reserved. 11 = Disabled.
3:2	BANK3_PIN17	RW	0x3	EMI_DQS1 Pin Function Selection. 00 = EMI_DQS1. 01 = Reserved. 10 = Reserved. 11 = Disabled.
1:0	BANK3_PIN16	RW	0x3	EMI_DQS0 Pin Function Selection. 00 = EMI_DQS0. 01 = Reserved. 10 = Reserved. 11 = Disabled.

DESCRIPTION:

This register allows the programmer to select which hardware interface blocks drive the 6 pins shown above.

33.4.10. PINCTRL Drive Strength and Voltage Register 0 Description

The PINCTRL Drive Strength and Voltage Register 0 selects the current drive strength for eight pins of Bank 0.

HW_PINCTRL_DRIVE0	0x80018200
HW_PINCTRL_DRIVE0_SET	0x80018204
HW_PINCTRL_DRIVE0_CLR	0x80018208
HW_PINCTRL_DRIVE0_TOG	0x8001820C

RSVD	3 1
BANK0_PIN07_V	3 0
BANK0_PIN07_MA	2 9
RSVD	2 8
BANK0_PIN06_V	2 7
BANK0_PIN06_MA	2 6
RSVD	2 5
BANK0_PIN05_V	2 4
BANK0_PIN05_MA	2 3
RSVD	2 2
BANK0_PIN04_V	2 1
BANK0_PIN04_MA	2 0
RSVD	1 9
BANK0_PIN03_V	1 8
BANK0_PIN03_MA	1 7
RSVD	1 6
BANK0_PIN02_V	1 5
BANK0_PIN02_MA	1 4
RSVD	1 3
BANK0_PIN01_V	1 2
BANK0_PIN01_MA	1 1
RSVD	0 0
BANK0_PIN00_V	0 9
BANK0_PIN00_MA	0 8
RSVD	0 7
BANK0_PIN00_V	0 6
BANK0_PIN00_MA	0 5
RSVD	0 4
BANK0_PIN00_V	0 3
BANK0_PIN00_MA	0 2
RSVD	0 1
BANK0_PIN00_V	0 0
BANK0_PIN00_MA	0 0

BITS	LABEL	RW	RESET	DEFINITION
31	RSVD	RO	0x0	Reserved.
30	BANK0_PIN07_V	RW	0x1	GPMI_D07 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
29:28	BANK0_PIN07_MA	RW	0x0	GPMI_D07 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
27	RSVD	RO	0x0	Reserved.
26	BANK0_PIN06_V	RW	0x1	GPMI_D06 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK0_PIN06_MA	RW	0x0	GPMI_D06 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK0_PIN05_V	RW	0x1	GPMI_D05 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK0_PIN05_MA	RW	0x0	GPMI_D05 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK0_PIN04_V	RW	0x1	GPMI_D04 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.

Table 1176. HW_PINCTRL_DRIVE0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:16	BANK0_PIN04_MA	RW	0x0	GPMI_D04 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK0_PIN03_V	RW	0x1	GPMI_D03 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK0_PIN03_MA	RW	0x0	GPMI_D03 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK0_PIN02_V	RW	0x1	GPMI_D02 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK0_PIN02_MA	RW	0x0	GPMI_D02 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK0_PIN01_V	RW	0x1	GPMI_D01 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK0_PIN01_MA	RW	0x0	GPMI_D01 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.
2	BANK0_PIN00_V	RW	0x1	GPMI_D00 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK0_PIN00_MA	RW	0x0	GPMI_D00 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 0 selects the drive strength and voltage for pins that are configured for output.

STMP3770

33.4.11. PINCTRL Drive Strength and Voltage Register 1 Description

The PINCTRL Drive Strength and Voltage Register 1 selects the current drive strength for eight pins of Bank 0.

HW_PINCTRL_DRIVE1 0x80018210
 HW_PINCTRL_DRIVE1_SET 0x80018214
 HW_PINCTRL_DRIVE1_CLR 0x80018218
 HW_PINCTRL_DRIVE1_TOG 0x8001821C

Table 1177. HW_PINCTRL_DRIVE1

3	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3
RSVD	BANK0_PIN15_V	BANK0_PIN15_MA	RSVD	BANK0_PIN14_V	BANK0_PIN14_MA	RSVD	BANK0_PIN13_V	BANK0_PIN13_MA	RSVD	BANK0_PIN12_V	BANK0_PIN12_MA	RSVD	BANK0_PIN11_V	BANK0_PIN11_MA	RSVD	BANK0_PIN10_V	BANK0_PIN10_MA	RSVD	BANK0_PIN09_V	BANK0_PIN09_MA	RSVD	BANK0_PIN08_V	BANK0_PIN08_MA					

Table 1178. HW_PINCTRL_DRIVE1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	RSVD	RO	0x0	Reserved.
30	BANK0_PIN15_V	RW	0x1	GPML_D15 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
29:28	BANK0_PIN15_MA	RW	0x0	GPML_D15 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
27	RSVD	RO	0x0	Reserved.
26	BANK0_PIN14_V	RW	0x1	GPML_D14 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK0_PIN14_MA	RW	0x0	GPML_D14 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK0_PIN13_V	RW	0x1	GPML_D13 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK0_PIN13_MA	RW	0x0	GPML_D13 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

Table 1178. HW_PINCTRL_DRIVE1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19	RSVD	RO	0x0	Reserved.
18	BANK0_PIN12_V	RW	0x1	GPMI_D12 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK0_PIN12_MA	RW	0x0	GPMI_D12 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK0_PIN11_V	RW	0x1	GPMI_D11 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK0_PIN11_MA	RW	0x0	GPMI_D11 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK0_PIN10_V	RW	0x1	GPMI_D10 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK0_PIN10_MA	RW	0x0	GPMI_D10 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK0_PIN09_V	RW	0x1	GPMI_D09 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK0_PIN09_MA	RW	0x0	GPMI_D09 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.
2	BANK0_PIN08_V	RW	0x1	GPMI_D08 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK0_PIN08_MA	RW	0x0	GPMI_D08 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 1 selects the drive strength and voltage for pins that are configured for output.

33.4.12. PINCTRL Drive Strength and Voltage Register 2 Description

The PINCTRL Drive Strength and Voltage Register 2 selects the current drive strength for eight pins of Bank 0.

```
HW_PINCTRL_DRIVE2      0x80018220
```

```
HW_PINCTRL_DRIVE2_SET      0x80018224
```

HW_PINCTRL_DRIVE2_CLR	0x80018228
-----------------------	------------

HW_PINCTRL_DRIVE2_TOG	0x8001822C
-----------------------	------------

Table 1179. HW_PINCTRL_DRIVE2

RSVD	3	1
BANK0_PIN23_V	3	0
BANK0_PIN23_MA	2	9
	2	8
RSVD	2	7
BANK0_PIN22_V	2	6
BANK0_PIN22_MA	2	5
	2	4
RSVD	2	3
BANK0_PIN21_V	2	2
BANK0_PIN21_MA	2	1
	2	0
RSVD	1	9
BANK0_PIN20_V	1	8
BANK0_PIN20_MA	1	7
	1	6
RSVD	1	5
BANK0_PIN19_V	1	4
BANK0_PIN19_MA	1	3
	1	2
RSVD	1	1
BANK0_PIN18_V	1	0
BANK0_PIN18_MA	0	9
	0	8
RSVD	0	7
BANK0_PIN17_V	0	6
BANK0_PIN17_MA	0	5
	0	4
RSVD	0	3
BANK0_PIN16_V	0	2
BANK0_PIN16_MA	0	1
	0	0

Table 1180. HW_PINCTRL_DRIVE2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	RSVD	RO	0x0	Reserved.
30	BANK0_PIN23_V	RW	0x1	GPMI_IRQ Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
29:28	BANK0_PIN23_MA	RW	0x0	GPMI_IRQ Pin Output Drive Strength Selection. 00 = Reserved. 01 = 8 mA. 10 = Reserved. 11 = 16 mA.
27	RSVD	RO	0x0	Reserved.
26	BANK0_PIN22_V	RW	0x1	GPMI_RESETN Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK0_PIN22_MA	RW	0x0	GPMI_RESETN Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK0_PIN21_V	RW	0x1	GPMI_RDY3 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.

Table 1180. HW_PINCTRL_DRIVE2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:20	BANK0_PIN21_MA	RW	0x0	GPML_RDY3 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK0_PIN20_V	RW	0x1	GPML_RDY2 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK0_PIN20_MA	RW	0x0	GPML_RDY2 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK0_PIN19_V	RW	0x1	GPML_RDY0 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK0_PIN19_MA	RW	0x0	GPML_RDY0 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK0_PIN18_V	RW	0x1	GPML_A2 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK0_PIN18_MA	RW	0x0	GPML_A2 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK0_PIN17_V	RW	0x1	GPML_A1 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK0_PIN17_MA	RW	0x0	GPML_A1 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.

Table 1182. HW_PINCTRL_DRIVE3 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
18	BANK0_PIN28_V	RW	0x1	UART2_RX Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK0_PIN28_MA	RW	0x0	UART2_RX Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK0_PIN27_V	RW	0x1	UART2_RTS Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK0_PIN27_MA	RW	0x0	UART2_RTS Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK0_PIN26_V	RW	0x1	UART2_CTS Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK0_PIN26_MA	RW	0x0	UART2_CTS Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK0_PIN25_V	RW	0x1	GPML_RDN Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK0_PIN25_MA	RW	0x0	GPML_RDN Pin Output Drive Strength Selection. 00 = Reserved. 01 = 8 mA. 10 = Reserved. 11 = 16 mA.
3	RSVD	RO	0x0	Reserved.
2	BANK0_PIN24_V	RW	0x1	GPML_WRN Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK0_PIN24_MA	RW	0x0	GPML_WRN Pin Output Drive Strength Selection. 00 = Reserved. 01 = 8 mA. 10 = Reserved. 11 = 16 mA.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 3 selects the drive strength and voltage for pins that are configured for output.

33.4.14. PINCTRL Drive Strength and Voltage Register 4 Description

The PINCTRL Drive Strength and Voltage Register 4 selects the current drive strength for eight pins of Bank 1.

```
HW_PINCTRL_DRIVE4      0x80018240
```

```
HW_PINCTRL_DRIVE4_SET      0x80018244
```

HW_PINCTRL_DRIVE4_CLR	0x80018248
-----------------------	------------

HW_PINCTRL_DRIVE4_TOG	0x8001824C
-----------------------	------------

Table 1183. HW_PINCTRL_DRIVE4

RSVD	3	1
BANK1_PIN07_V	3	0
BANK1_PIN07_MA	2	9
	2	8
RSVD	2	7
BANK1_PIN06_V	2	6
BANK1_PIN06_MA	2	5
	2	4
RSVD	2	3
BANK1_PIN05_V	2	2
BANK1_PIN05_MA	2	1
	2	0
RSVD	1	9
BANK1_PIN04_V	1	8
BANK1_PIN04_MA	1	7
	1	6
RSVD	1	5
BANK1_PIN03_V	1	4
BANK1_PIN03_MA	1	3
	1	2
RSVD	1	1
BANK1_PIN02_V	1	0
BANK1_PIN02_MA	0	9
	0	8
RSVD	0	7
BANK1_PIN01_V	0	6
BANK1_PIN01_MA	0	5
	0	4
RSVD	0	3
BANK1_PIN00_V	0	2
BANK1_PIN00_MA	0	1
	0	0

Table 1184. HW_PINCTRL_DRIVE4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	RSVD	RO	0x0	Reserved.
30	BANK1_PIN07_V	RW	0x1	LCD_D07 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
29:28	BANK1_PIN07_MA	RW	0x0	LCD_D07 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
27	RSVD	RO	0x0	Reserved.
26	BANK1_PIN06_V	RW	0x1	LCD_D06 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK1_PIN06_MA	RW	0x0	LCD_D06 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK1_PIN05_V	RW	0x1	LCD_D05 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.

Table 1184. HW_PINCTRL_DRIVE4 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
21:20	BANK1_PIN05_MA	RW	0x0	LCD_D05 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK1_PIN04_V	RW	0x1	LCD_D04 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK1_PIN04_MA	RW	0x0	LCD_D04 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK1_PIN03_V	RW	0x1	LCD_D03 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK1_PIN03_MA	RW	0x0	LCD_D03 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK1_PIN02_V	RW	0x1	LCD_D02 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK1_PIN02_MA	RW	0x0	LCD_D02 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK1_PIN01_V	RW	0x1	LCD_D01 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK1_PIN01_MA	RW	0x0	LCD_D01 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.

Table 1186. HW_PINCTRL_DRIVE5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26	BANK1_PIN14_V	RW	0x1	LCD_D14 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK1_PIN14_MA	RW	0x0	LCD_D14 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK1_PIN13_V	RW	0x1	LCD_D13 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK1_PIN13_MA	RW	0x0	LCD_D13 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK1_PIN12_V	RW	0x1	LCD_D12 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK1_PIN12_MA	RW	0x0	LCD_D12 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK1_PIN11_V	RW	0x1	LCD_D11 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK1_PIN11_MA	RW	0x0	LCD_D11 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK1_PIN10_V	RW	0x1	LCD_D10 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK1_PIN10_MA	RW	0x0	LCD_D10 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.

Table 1186. HW_PINCTRL_DRIVE5 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
6	BANK1_PIN09_V	RW	0x1	LCD_D09 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK1_PIN09_MA	RW	0x0	LCD_D09 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.
2	BANK1_PIN08_V	RW	0x1	LCD_D08 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK1_PIN08_MA	RW	0x0	LCD_D08 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 5 selects the drive strength and voltage for pins that are configured for output.

33.4.16. PINCTRL Drive Strength and Voltage Register 6 Description

The PINCTRL Drive Strength and Voltage Register 6 selects the current drive strength for eight pins of Bank 1.

HW_PINCTRL_DRIVE6 0x80018260
 HW_PINCTRL_DRIVE6_SET 0x80018264
 HW_PINCTRL_DRIVE6_CLR 0x80018268
 HW_PINCTRL_DRIVE6_TOG 0x8001826C

Table 1187. HW_PINCTRL_DRIVE6

RSVD	3 1
BANK1_PIN23_V	3 0
BANK1_PIN23_MA	2 9
	2 8
RSVD	2 7
BANK1_PIN22_V	2 6
BANK1_PIN22_MA	2 5
	2 4
RSVD	2 3
BANK1_PIN21_V	2 2
BANK1_PIN21_MA	2 1
	2 0
RSVD	1 9
BANK1_PIN20_V	1 8
BANK1_PIN20_MA	1 7
	1 6
RSVD	1 5
BANK1_PIN19_V	1 4
BANK1_PIN19_MA	1 3
	1 2
RSVD	1 1
BANK1_PIN18_V	1 0
BANK1_PIN18_MA	0 9
	0 8
RSVD	0 7
BANK1_PIN17_V	0 6
BANK1_PIN17_MA	0 5
	0 4
RSVD	0 3
BANK1_PIN16_V	0 2
BANK1_PIN16_MA	0 1
	0 0

Table 1188. HW_PINCTRL_DRIVE6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	RSVD	RO	0x0	Reserved.
30	BANK1_PIN23_V	RW	0x1	SSP1_SCK Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
29:28	BANK1_PIN23_MA	RW	0x0	SSP1_SCK Pin Output Drive Strength Selection. 00 = Reserved. 01 = 8 mA. 10 = Reserved. 11 = 16 mA.
27	RSVD	RO	0x0	Reserved.
26	BANK1_PIN22_V	RW	0x1	SSP1_CMD Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK1_PIN22_MA	RW	0x0	SSP1_CMD Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK1_PIN21_V	RW	0x1	LCD_BUSY Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK1_PIN21_MA	RW	0x0	LCD_BUSY Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK1_PIN20_V	RW	0x1	LCD_CS Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK1_PIN20_MA	RW	0x0	LCD_CS Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK1_PIN19_V	RW	0x1	LCD_RD_E Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK1_PIN19_MA	RW	0x0	LCD_RD_E Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.

Table 1188. HW_PINCTRL_DRIVE6 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
10	BANK1_PIN18_V	RW	0x1	LCD_WR_RWN Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK1_PIN18_MA	RW	0x0	LCD_WR_RWN Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK1_PIN17_V	RW	0x1	LCD_RS Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK1_PIN17_MA	RW	0x0	LCD_RS Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.
2	BANK1_PIN16_V	RW	0x1	LCD_RESET Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK1_PIN16_MA	RW	0x0	LCD_RESET Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 6 selects the drive strength and voltage for pins that are configured for output.

33.4.17. PINCTRL Drive Strength and Voltage Register 7 Description

The PINCTRL Drive Strength and Voltage Register 7 selects the current drive strength for five pins of Bank 1.

HW_PINCTRL_DRIVE7	0x80018270
HW_PINCTRL_DRIVE7_SET	0x80018274
HW_PINCTRL_DRIVE7_CLR	0x80018278
HW_PINCTRL_DRIVE7_TOG	0x8001827C

Table 1189. HW_PINCTRL_DRIVE7

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8
RSVD																							
BANK1_PIN28_V																							
BANK1_PIN28_MA																							
RSVD																							
BANK1_PIN27_V																							
BANK1_PIN27_MA																							
RSVD																							
BANK1_PIN26_V																							
BANK1_PIN26_MA																							
RSVD																							
BANK1_PIN25_V																							
BANK1_PIN25_MA																							
RSVD																							
BANK1_PIN24_V																							
BANK1_PIN24_MA																							

Table 1190. HW_PINCTRL_DRIVE7 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:20	RSVD	RO	0x0	Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK1_PIN28_V	RW	0x1	SSP1_DETECT Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK1_PIN28_MA	RW	0x0	SSP1_DETECT Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK1_PIN27_V	RW	0x1	SSP1_DATA3 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK1_PIN27_MA	RW	0x0	SSP1_DATA3 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK1_PIN26_V	RW	0x1	SSP1_DATA2 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK1_PIN26_MA	RW	0x0	SSP1_DATA2 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK1_PIN25_V	RW	0x1	SSP1_DATA1 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.

Table 1192. HW_PINCTRL_DRIVE8 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
29:28	BANK2_PIN07_MA	RW	0x0	ROTARYA Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
27	RSVD	RO	0x0	Reserved.
26	BANK2_PIN06_V	RW	0x1	I2C_SDA Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK2_PIN06_MA	RW	0x0	I2C_SDA Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK2_PIN05_V	RW	0x1	I2C_SCL Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK2_PIN05_MA	RW	0x0	I2C_SCL Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK2_PIN04_V	RW	0x1	PWM4 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK2_PIN04_MA	RW	0x0	PWM4 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 16 mA. 10 = 20 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK2_PIN03_V	RW	0x1	PWM3 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK2_PIN03_MA	RW	0x0	PWM3 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK2_PIN02_V	RW	0x1	PWM2 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.

Table 1194. HW_PINCTRL_DRIVE9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	RSVD	RO	0x0	Reserved.
30	BANK2_PIN15_V	RW	0x1	EMI_CE3N Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
29:28	BANK2_PIN15_MA	RW	0x0	EMI_CE3N Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
27	RSVD	RO	0x0	Reserved.
26	BANK2_PIN14_V	RW	0x1	EMI_CE2N Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK2_PIN14_MA	RW	0x0	EMI_CE2N Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
23	RSVD	RO	0x0	Reserved.
22	BANK2_PIN13_V	RW	0x1	EMI_CE1N Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK2_PIN13_MA	RW	0x0	EMI_CE1N Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
19	RSVD	RO	0x0	Reserved.
18	BANK2_PIN12_V	RW	0x1	EMI_CE0N Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK2_PIN12_MA	RW	0x0	EMI_CE0N Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK2_PIN11_V	RW	0x1	EMI_CASN Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK2_PIN11_MA	RW	0x0	EMI_CASN Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.

Table 1194. HW_PINCTRL_DRIVE9 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
10	BANK2_PIN10_V	RW	0x1	EMI_RASN Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK2_PIN10_MA	RW	0x0	EMI_RASN Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK2_PIN09_V	RW	0x1	EMI_CKE Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK2_PIN09_MA	RW	0x0	EMI_CKE Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.
2	BANK2_PIN08_V	RW	0x1	ROTARYB Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK2_PIN08_MA	RW	0x0	ROTARYB Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 9 selects the drive strength and voltage for pins that are configured for output.

33.4.20. PINCTRL Drive Strength and Voltage Register 10 Description

The PINCTRL Drive Strength and Voltage Register 10 selects the current drive strength for eight pins of Bank 2.

HW_PINCTRL_DRIVE10	0x800182A0
HW_PINCTRL_DRIVE10_SET	0x800182A4
HW_PINCTRL_DRIVE10_CLR	0x800182A8
HW_PINCTRL_DRIVE10_TOG	0x800182AC

STMP3770

Table 1196. HW_PINCTRL_DRIVE10 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
17:16	BANK2_PIN20_MA	RW	0x0	EMI_A04 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK2_PIN19_V	RW	0x1	EMI_A03 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK2_PIN19_MA	RW	0x0	EMI_A03 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK2_PIN18_V	RW	0x1	EMI_A02 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK2_PIN18_MA	RW	0x0	EMI_A02 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK2_PIN17_V	RW	0x1	EMI_A01 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK2_PIN17_MA	RW	0x0	EMI_A01 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.
2	BANK2_PIN16_V	RW	0x1	EMI_A00 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK2_PIN16_MA	RW	0x0	EMI_A00 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 10 selects the drive strength and voltage for pins that are configured for output.

STMP3770



Table 1198. HW_PINCTRL_DRIVE11 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
19	RSVD	RO	0x0	Reserved.
18	BANK2_PIN28_V	RW	0x1	EMI_A12 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK2_PIN28_MA	RW	0x0	EMI_A12 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
15	RSVD	RO	0x0	Reserved.
14	BANK2_PIN27_V	RW	0x1	EMI_A11 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK2_PIN27_MA	RW	0x0	EMI_A11 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
11	RSVD	RO	0x0	Reserved.
10	BANK2_PIN26_V	RW	0x1	EMI_A10 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK2_PIN26_MA	RW	0x0	EMI_A10 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
7	RSVD	RO	0x0	Reserved.
6	BANK2_PIN25_V	RW	0x1	EMI_A09 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK2_PIN25_MA	RW	0x0	EMI_A09 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.
3	RSVD	RO	0x0	Reserved.
2	BANK2_PIN24_V	RW	0x1	EMI_A08 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK2_PIN24_MA	RW	0x0	EMI_A08 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = Reserved.

STMP3770**Table 1200. HW_PINCTRL_DRIVE12 Bit Field Descriptions**

BITS	LABEL	RW	RESET	DEFINITION
21:20	BANK3_PIN05_MA	RW	0x0	EMI_D05 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
19	RSVD	RO	0x0	Reserved.
18	BANK3_PIN04_V	RW	0x1	EMI_D04 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK3_PIN04_MA	RW	0x0	EMI_D04 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
15	RSVD	RO	0x0	Reserved.
14	BANK3_PIN03_V	RW	0x1	EMI_D03 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK3_PIN03_MA	RW	0x0	EMI_D03 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
11	RSVD	RO	0x0	Reserved.
10	BANK3_PIN02_V	RW	0x1	EMI_D02 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK3_PIN02_MA	RW	0x0	EMI_D02 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
7	RSVD	RO	0x0	Reserved.
6	BANK3_PIN01_V	RW	0x1	EMI_D01 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK3_PIN01_MA	RW	0x0	EMI_D01 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
3	RSVD	RO	0x0	Reserved.

Table 1200. HW_PINCTRL_DRIVE12 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
2	BANK3_PIN00_V	RW	0x1	EMI_D00 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
1:0	BANK3_PIN00_MA	RW	0x0	EMI_D00 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.

DESCRIPTION:

The PINCTRL Drive Strength and Voltage Register 12 selects the drive strength and voltage for pins that are configured for output.

33.4.23. PINCTRL Drive Strength and Voltage Register 13 Description

The PINCTRL Drive Strength and Voltage Register 13 selects the current drive strength for eight pins of Bank 3.

HW_PINCTRL_DRIVE13	0x800182D0
HW_PINCTRL_DRIVE13_SET	0x800182D4
HW_PINCTRL_DRIVE13_CLR	0x800182D8
HW_PINCTRL_DRIVE13_TOG	0x800182DC

Table 1201. HW_PINCTRL_DRIVE13

RSVD	3	1
BANK3_PIN15_V	3	0
BANK3_PIN15_MA	2	9
	2	8
RSVD	2	7
BANK3_PIN14_V	2	6
BANK3_PIN14_MA	2	5
	2	4
RSVD	2	3
BANK3_PIN13_V	2	2
BANK3_PIN13_MA	2	1
	2	0
RSVD	1	9
BANK3_PIN12_V	1	8
BANK3_PIN12_MA	1	7
	1	6
RSVD	1	5
BANK3_PIN11_V	1	4
BANK3_PIN11_MA	1	3
	1	2
RSVD	1	1
BANK3_PIN10_V	1	0
BANK3_PIN10_MA	0	9
	0	8
RSVD	0	7
BANK3_PIN09_V	0	6
BANK3_PIN09_MA	0	5
	0	4
RSVD	0	3
BANK3_PIN08_V	0	2
BANK3_PIN08_MA	0	1
	0	0

Table 1202. HW_PINCTRL_DRIVE13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31	RSVD	RO	0x0	Reserved.
30	BANK3_PIN15_V	RW	0x1	EMI_D15 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
29:28	BANK3_PIN15_MA	RW	0x0	EMI_D15 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
27	RSVD	RO	0x0	Reserved.

STMP3770

Table 1202. HW_PINCTRL_DRIVE13 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
26	BANK3_PIN14_V	RW	0x1	EMI_D14 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
25:24	BANK3_PIN14_MA	RW	0x0	EMI_D14 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
23	RSVD	RO	0x0	Reserved.
22	BANK3_PIN13_V	RW	0x1	EMI_D13 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK3_PIN13_MA	RW	0x0	EMI_D13 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
19	RSVD	RO	0x0	Reserved.
18	BANK3_PIN12_V	RW	0x1	EMI_D12 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK3_PIN12_MA	RW	0x0	EMI_D12 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
15	RSVD	RO	0x0	Reserved.
14	BANK3_PIN11_V	RW	0x1	EMI_D11 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK3_PIN11_MA	RW	0x0	EMI_D11 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
11	RSVD	RO	0x0	Reserved.
10	BANK3_PIN10_V	RW	0x1	EMI_D10 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK3_PIN10_MA	RW	0x0	EMI_D10 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
7	RSVD	RO	0x0	Reserved.

STMP3770

Table 1204. HW_PINCTRL_DRIVE14 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
22	BANK3_PIN21_V	RW	0x1	EMI_CLKN Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
21:20	BANK3_PIN21_MA	RW	0x0	EMI_CLKN Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
19	RSVD	RO	0x0	Reserved.
18	BANK3_PIN20_V	RW	0x1	EMI_CLK Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
17:16	BANK3_PIN20_MA	RW	0x0	EMI_CLK Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
15	RSVD	RO	0x0	Reserved.
14	BANK3_PIN19_V	RW	0x1	EMI_DQM1 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
13:12	BANK3_PIN19_MA	RW	0x0	EMI_DQM1 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
11	RSVD	RO	0x0	Reserved.
10	BANK3_PIN18_V	RW	0x1	EMI_DQM0 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
9:8	BANK3_PIN18_MA	RW	0x0	EMI_DQM0 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
7	RSVD	RO	0x0	Reserved.
6	BANK3_PIN17_V	RW	0x1	EMI_DQS1 Pin Voltage Selection. 0 = 1.8 V. 1 = 3.3 V.
5:4	BANK3_PIN17_MA	RW	0x0	EMI_DQS1 Pin Output Drive Strength Selection. 00 = 4 mA. 01 = 8 mA. 10 = 12 mA. 11 = 16 mA.
3	RSVD	RO	0x0	Reserved.

HW_PINCTRL_DOUT1	0x80018410
HW_PINCTRL_DOUT1_SET	0x80018414
HW_PINCTRL_DOUT1_CLR	0x80018418
HW_PINCTRL_DOUT1_TOG	0x8001841C

Table 1215. HW PINCTRL DOUT1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD			DOUT																												

Table 1216. HW_PINCTRL_DOUT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x0	Reserved.
28:0	DOUT	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 29 pins in Bank 1.

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.

33.4.31. PINCTRL Bank 2 Data Output Register Description

HW_PINCTRL_DOUT2	0x80018420
HW_PINCTRL_DOUT2_SET	0x80018424
HW_PINCTRL_DOUT2_CLR	0x80018428
HW_PINCTRL_DOUT2_TOG	0x8001842C

Table 1217. HW_PINCTRL_DOUT2

3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0			
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
DOUT																															

Table 1218. HW_PINCTRL_DOUT2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	DOUT	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 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.

33.4.32. 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	0x80018500
HW_PINCTRL_DIN0_SET	0x80018504
HW_PINCTRL_DIN0_CLR	0x80018508
HW_PINCTRL_DIN0_TOG	0x8001850C

Table 1219. HW_PINCTRL_DIN0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	1 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD		DIN																													

Table 1220. HW_PINCTRL_DIN0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29:0	DIN	RO	0x00000000	Each bit in this read-only register corresponds to one of the 30 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.

33.4.33. 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.

33.4.38. 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	0x80018700
HW_PINCTRL_PIN2IRQ0_SET	0x80018704
HW_PINCTRL_PIN2IRQ0_CLR	0x80018708
HW_PINCTRL_PIN2IRQ0_TOG	0x8001870C

Table 1231. HW_PINCTRL_PIN2IRQ0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD		PIN2IRQ																													

Table 1232. HW_PINCTRL_PIN2IRQ0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29:0	PIN2IRQ	RW	0x00000000	Each bit in this register corresponds to one of the 30 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_IRQPOLO 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.

33.4.39. PINCTRL Bank 1 Interrupt Select Register Description

The Bank 1 Interrupt Select register selects which of the Bank 1 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ1	0x80018710
HW_PINCTRL_PIN2IRQ1_SET	0x80018714
HW_PINCTRL_PIN2IRQ1_CLR	0x80018718
HW_PINCTRL_PIN2IRQ1_TOG	0x8001871C

Table 1236. HW_PINCTRL_PIN2IRQ2 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:0	PIN2IRQ	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_IRQEN0 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 0 will cause bits to be set in the HW_PINCTRL_IRQSTAT2 register.

33.4.41. 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	0x80018800
HW_PINCTRL_IRQEN0_SET	0x80018804
HW_PINCTRL_IRQEN0_CLR	0x80018808
HW_PINCTRL_IRQEN0_TOG	0x8001880C

Table 1237. HW_PINCTRL_IRQEN0

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD		IRQEN																													

Table 1238. HW_PINCTRL_IRQEN0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29:0	IRQEN	RW	0x00000000	Each bit in this register corresponds to one of the 30 pins in Bank 0: 0 = Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0. 1 = Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0.

STMP3770

Table 1256. HW_PINCTRL_IRQSTAT0 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:30	RSVD	RO	0x0	Reserved.
29:0	IRQSTAT	RW	0x00000000	Each bit in this register corresponds to one of the 30 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_PINCTRL_IRQEN0 mask register, then the GPIO0 interrupt will be asserted to the interrupt collector.

33.4.51. 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	0x80018b10
HW_PINCTRL_IRQSTAT1_SET	0x80018b14
HW_PINCTRL_IRQSTAT1_CLR	0x80018b18
HW_PINCTRL_IRQSTAT1_TOG	0x80018b1C

Table 1257. HW_PINCTRL_IRQSTAT1

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	0 9	0 8	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0
RSVD				IRQSTAT																											

Table 1258. HW_PINCTRL_IRQSTAT1 Bit Field Descriptions

BITS	LABEL	RW	RESET	DEFINITION
31:29	RSVD	RO	0x0	Reserved.
28:0	IRQSTAT	RW	0x00000000	Each bit in this register corresponds to one of the 29 pins in Bank 1: 0 = No interrupt pending. 1 = Interrupt pending.

STMP3770



34. REGISTER MACRO USAGE

This chapter provides background on the STMP3770 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 chip database.

34.1. Definitions

```

////////////////////////////////////
// These macros will be generated from the chip data base 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;
}

////////////////////////////////////

```

34.2. Background

The STMP3770 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 a 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 register descriptions. These include files provide a consistent set of C defines and macros that should be used to access the hardware registers.

The STMP3770 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 0, 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 0 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.

34.3. 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 bit field 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(<valuenam>)
// - macros for generating a bit field value. The parameter is masked
// and shifted to the field position.
//
// BW_<module>_<regname>_<field>()
// - macro for writing a bit field. Usually expands to a CS operation.
// Not generated for read-only fields.
//
// BV_<module>_<regname>_<field>__<valuenam>
// - 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.
```

34.4. 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.

34.4.1. 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
```

34.4.2. 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
```

34.4.3. 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
```

34.4.4. 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;
```

34.4.5. 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;
```

34.4.6. 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;
```

34.4.7. 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;
```

34.4.8. Reading Entire Register

```
0 // Preferred
  i = HW_GPML_CTRL0_RD();

  // Alternate (same as above, just different syntax)
  i = HW_GPML_CTRL0.U;
```

34.4.9. 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
    BW_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);
}
```

34.4.10. 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. Poll until this has happened before subsequently
// preparing soft-reset and clock gate
HW_GPML_CTRL0_CLR(BM_GPML_CTRL0_SFTRST);
HW_GPML_CTRL0_CLR(BM_GPML_CTRL0_CLKGATE);

// asserting soft-reset
HW_GPML_CTRL0_SET(BM_GPML_CTRL0_SFTRST);

// waiting for confirmation of soft-reset
while (!HW_GPML_CTRL0.B_CLKGATE)
{
    // busy wait
}

// Done.
HW_GPML_CTRL0_CLR(BM_GPML_CTRL0_SFTRST);
HW_GPML_CTRL0_CLR(BM_GPML_CTRL0_CLKGATE);
```

34.5. Summary Preferred

```
// Setting, clearing, toggling 1-bit wide field
HW_GPML_CTRL0_SET(BM_GPML_CTRL0_UDMA);
HW_GPML_CTRL0_CLR(BM_GPML_CTRL0_DEV_IRQ_EN);
HW_GPML_CTRL0_TOG(BM_GPML_CTRL0_RUN);

// Modifying n-bit wide field
BW_GPML_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 bit field
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();
```

34.6. 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 bit field
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;
```

34.7. 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(<valuenam>)
// - macros for generating a bit field value. The parameter is masked
// and shifted to the field position.
//
// BV_<module>_<regname>_<field>__<valuenam>
// - 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
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//! }@
```

STMP3770



35. MEMORY MAP**Figure 163. STMP3770 Detailed 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											512 Kbyte On-Chip SRAM																		
01		AHB Default Slave ^a																													
100	XXXXXXXX	000	APBH 0x0: Interrupt Collector		XXXX	Register Selects	SET CLEAR TOGGLE	Byte																							
			APBH 0x1: APBH DMA																												
			APBH 0x2: ECC8																												
			APBH 0x3: GPMI																												
			APBH 0x4: SSP1																												
			APBH 0x5: Reserved		Decode 5 Unused																										
			APBH 0x6: PINCTRL		XXXX	Register Selects	SET CLEAR TOGGLE	Byte																							
			APBH 0x7: DIGCTL																												
			APBH 0x8: NOR Flash																												
			APBH 0x9: APBX DMA																												
			APBH 0xA: DCP																												
			APBH 0xB: OCOTP																												
			APBH 0xC: LCDIF																												
			APBH 0xD: SSP2																												
			APBH 0xE: Reserved		Decode E Unused																										
			APBH 0xF		Simulation termination and printf registers. Default APBH Slave on real chip.																										

STMP3770

Figure 163. STMP3770 Detailed 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	XXXXXXXXXX											01	APBX 0x0: CLKCTRL				XXX	Register Selects								SET CLEAR TOGGLE				Byte						
													APBX 0x1: SAIF1																							
													APBX 0x2: Power																							
													APBX 0x3: SAIF2																							
													APBX 0x4: AUDIOOUT																							
													APBX 0x6: AUDIOIN																							
													APBX 0x8: LRADC																							
													APBX 0xA: SPDIF																							
													APBX 0xC: I ² C																							
													APBX 0xE: RTC																							
													APBX 0x10: Reserved				Decode 10 Unused																			
													APBX 0x12: PWM				XXX	Register Selects								SET CLEAR TOGGLE				Byte						
													APBX 0x14: TIMROT																							
													APBX 0x16: APPUART																							
													APBX 0x18: DBGUART																							
													APBX 0x1A: DRI																							
													APBX 0x1C: IrDA																							
													APBX 0x1E: USB PHY																							
													10				XXXXXXXXXX								USB Controller Registers											
													110				XXX				Default First-Level Page Table 16KB															
													111								AHB Default Slave ^b															
101			AHB Default Slave ^a																																	
11		XXXXXXXXXXXXXXXXXX														On-Chip ROM or Interrupt Vector Register																				

Notes:

- If the ARM core accesses an address space described as “default slave”, it will receive an instruction or data abort. For all masters other than the ARM core, accesses to these same address spaces will trigger the master’s error interrupt.
- If the ARM core accesses an address space described as “default slave”, it will receive an instruction or data abort. For all masters other than the ARM core, accesses to these same address spaces will trigger the master’s error interrupt.

36. PACKAGE DRAWINGS

The STMP3770 is offered in two packages, which are illustrated in this chapter.

- 100-Pin Low-Profile Quad Flat Pack (LQFP)
- 100-Pin Ball Grid Array (BGA)

36.1. 100-Pin Low-Profile Quad Flat Pack (LQFP)

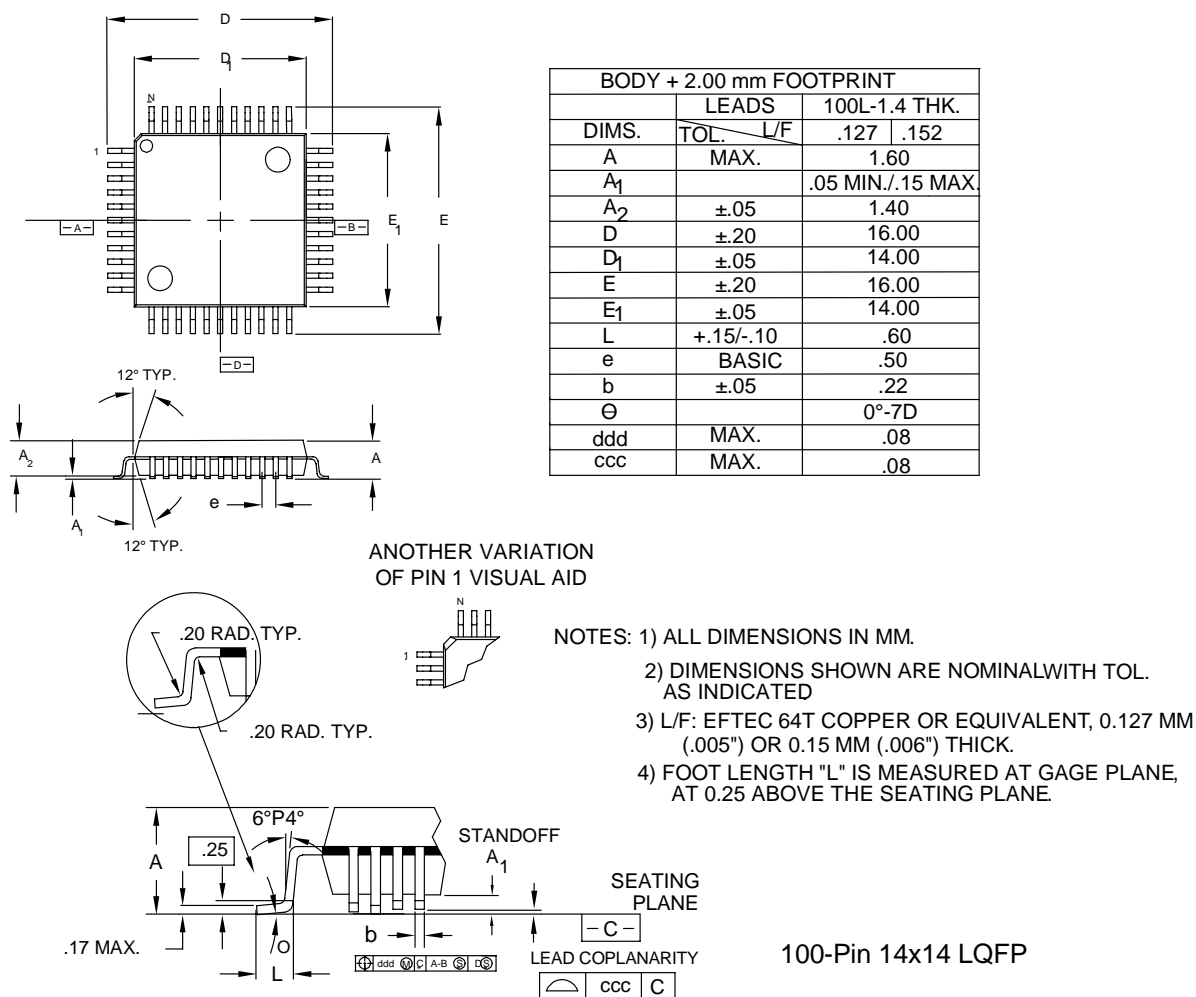


Figure 164. 100-Pin Low-Profile Quad Flat Pack (LQFP) Package Drawing

STMP3770

36.2. 100-Pin Ball Grid Array (BGA)

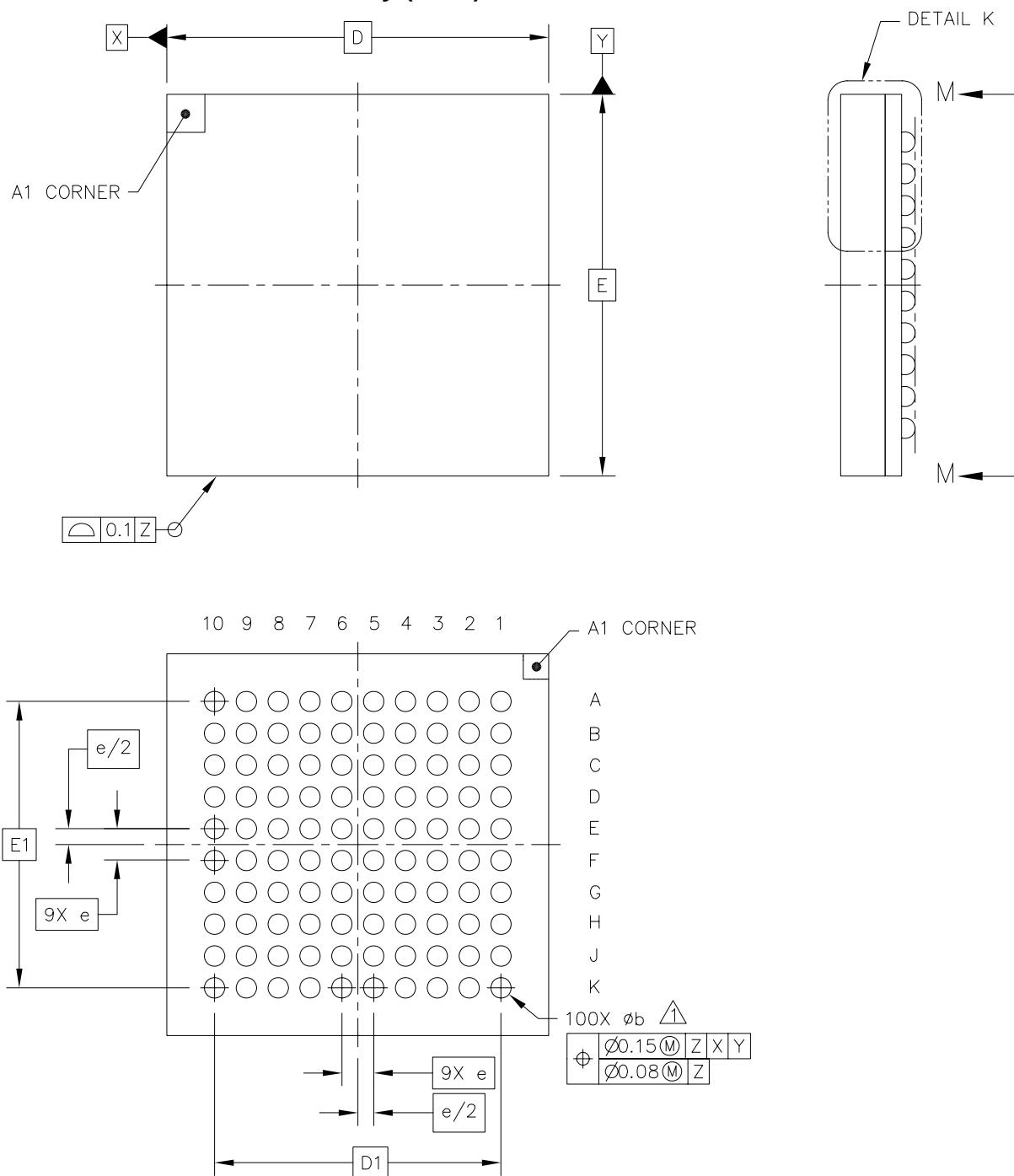
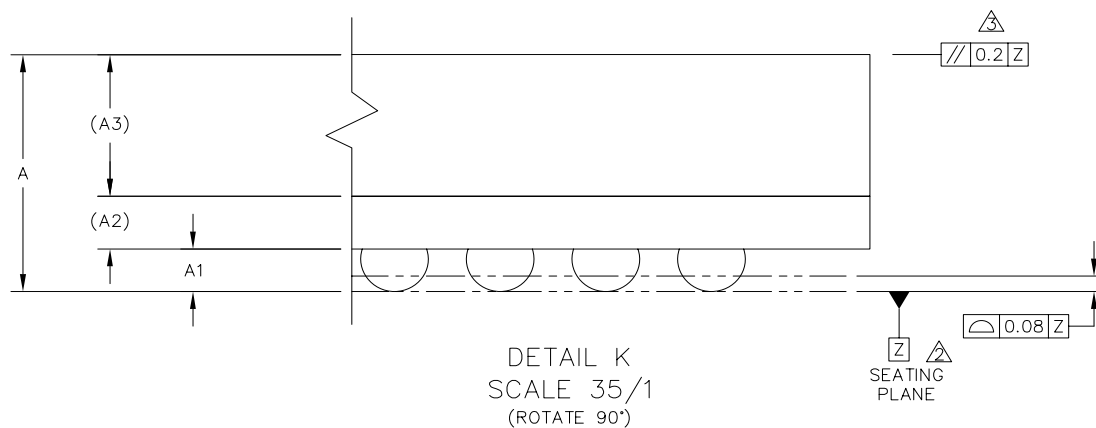


Figure 165. 100-Pin Ball Grid Array (BGA) Package Drawing, Part 1



DIM	MIN.	NOR.	MAX.	NOTES		
A	---		1	DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO DATUM PLANE Z. DATUM Z (SEATING PLANE) IS DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS. PARALLELISM MEASUREMENT SHALL EXCLUDE ANY EFFECT OF MARK ON TOP SURFACE OF PACKAGE.	UNIT	DIMENSION AND TOLERANCES
A1	0.16		0.26			
A2		0.21 REF				
A3		0.45 REF				
b	0.27		0.37			
D		6 BSC			MM	ASME Y14.5M
E		6 BSC				
e		0.5 BSC				
D1		4.5 BSC				
E1		4.5 BSC				
						MO-225

Figure 166. 100-Pin BGA Package Drawing, Part 2

STMP3770



37. STMP3770 PART NUMBERS AND ORDERING INFORMATION

Table 1261 shows the part number and features 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 1261. Part Numbers for STMP3770 Family Members

Part Number	Package	Description
STMP3770XXBJEA1x	100-pin BGA	No MP3 Encode Support No USB Host/OTG No IrDA
STMP3770XXLAEA1x	100-pin LQFP	No MP3 Encode Support No USB Host/OTG No IrDA

Note:

x = M for MP3 decode license included

x = N for No MP3 decode license

STMP3770



38. ACRONYMS AND ABBREVIATIONS

This appendix includes definitions for many of the acronyms and abbreviations found in this product data sheet.

AAC:	Advanced Audio Coding
AC:	Audio Coding
ADC:	Analog-to-Digital Converter
ADC:	Adaptive Differential Pulse-Code Modulation
AES:	Advanced Encryption Standard
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
BCB:	Boot Control Block
BGA:	Ball Grid Array
BIST:	Built-In Self-Test
BKPT:	Breakpoint
BLTC:	Boot Loader Transaction Controller
CBC	Cipher Block Chaining
CCS:	Command Completion Signaling
CE:	Consumer Electronics
CLKCTRL:	Clock Control
CP:	Charge Pump
CPUCLK:	Processor (ARM CPU) Clock
CRC:	Cyclic Redundancy Check
CSC:	Color-Space Conversion
CTS:	Clear To Send
DABT:	Data Abort
DAC:	Digital-to-Analog Converter
dB:	Decibel
DCP:	Data Co-Processor
DBBT:	Discovered Bad Block Table
DES:	Data Encryption Standard

STMP3770

DFD:	Digital Fractional Divider
DFLPT:	Default First-Level Page Table
DIGCTL:	Digital Control
DIO:	Digital Input/Output
DiVX:	Digital video codec created by DivXNetworks, Inc.
DMA:	Direct Memory Access
DRI:	Digital Radio Interface
DVI:	Digital Video Interface (ITU-R BT.656 mode)
ECB:	Electronic Book Code
ECC:	Error Correction Code
EL:	Electroluminescent
EMI:	External Memory Interface
EMICLK:	EMI Clock
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
HCLK:	Main and HBUS Peripherals Clock
HID:	Human Interface Device
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)
IRCLK:	IR Clock (source from IROVCLK)
IRQ:	Normal Peripheral Interrupt
ISR:	Interrupt Service Register
JEDEC:	Joint Electron Device Engineering Council
JPEG:	Joint Photographic Experts Group (computer image format)
JTAG:	Joint Test Action Group
LDLB:	Logical Drive Layout Block

LFE:	Low Frequency Effects
Li-Ion:	Lithium Ion (battery type)
LJ:	Left-Justified
LQFP:	Low-Profile Quad Flat Pack
LRADC:	Low Resolution ADC
LSB:	Least Significant Bit
MATT:	Multi-chip Attachment mode
MBR:	Master Boot Record
MIR:	Mid Infrared
MMC:	Multi-Media Card
MPEG4:	Motion Picture Experts Group 4 (standard for compressed video at 64 kbps)
MP3:	Moving Picture Experts Group Layer-3 Audio
MS:	Memory Stick
MSB:	Most Significant Bit
Mux:	Multiplexer
NCB:	NAND Control Block
NiMH:	Nickel Metal Hydride
NRZI:	Non-Return to Zero Inverted
NTSC:	National Television Systems Committee
OTG:	On the Go
OTP:	One Time Programmable
PABT:	Instruction Pre-Fetch Abort
PAL:	Phase Alternating Line
PCM:	Pulse Code Modulation
PDA:	Personal Digital Assistant
PDDRM:	Portable Device Digital Rights Management (DRM9)
PFD:	Phase Fractional Divider
PFM:	Pulse Frequency Modulation
PHY:	Physical Layer Protocol
PLL:	Phase-Locked Loop
PITC:	Plug-In Transfer Controller
PWM:	Pulse Width Modulation
RHID:	Recovery Human Interface Device
RJ:	Right-Justified
RTC:	Real-Time Clock
RTS:	Request To Send
RMW:	Read-Modify-Write
SAIF:	Serial Audio Interface
SB:	Safe Boot

STMP3770

SDIO:	Secure Digital Input/Output
SDK:	Software Development Kit
SHA:	Secure Hash Algorithm
SIR:	Serial Infrared
SJTAG:	Serial JTAG
SNR:	Signal-to-Noise Ratio
SOC:	System-on-a-Chip
SPDIF:	Sony-Philips Digital Interface Format
SPDIFCLK:	SPDIF Clock
SPI:	Serial Peripheral Interface
SSI:	Synchronous Serial Interface
SSP:	Synchronous Serial Port
SWI:	Software Interrupt
TAP:	Test Access Port
TBD:	To Be Determined
TDEA	Triple Data Encryption Algorithm
THD:	Total Harmonic Distortion
TPC:	Transfer Protocol Commands
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
VMI:	Virtual Memory Interface
WMDRM10:	Windows Media® Digital Rights Management 10 (Janus)
WMA:	Windows Media® Audio
XCLK:	XBUS Peripherals Clock

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_PTE_2048	0x800C2000	128
HW_APBH_CH0_BAR	0x80004070	282
HW_APBH_CH0_CMD	0x80004060	280
HW_APBH_CH0_CURCMDAR	0x80004040	279
HW_APBH_CH0_DEBUG1	0x80004090	283
HW_APBH_CH0_DEBUG2	0x800040A0	286
HW_APBH_CH0_NXTCMDAR	0x80004050	279
HW_APBH_CH0_SEMA	0x80004080	282
HW_APBH_CH1_BAR	0x800040E0	289
HW_APBH_CH1_CMD	0x800040D0	287
HW_APBH_CH1_CURCMDAR	0x800040B0	286
HW_APBH_CH1_DEBUG1	0x80004100	291
HW_APBH_CH1_DEBUG2	0x80004110	293
HW_APBH_CH1_NXTCMDAR	0x800040C0	287
HW_APBH_CH1_SEMA	0x800040F0	290
HW_APBH_CH2_BAR	0x80004150	296
HW_APBH_CH2_CMD	0x80004140	294
HW_APBH_CH2_CURCMDAR	0x80004120	293
HW_APBH_CH2_DEBUG1	0x80004170	298
HW_APBH_CH2_DEBUG2	0x80004180	299
HW_APBH_CH2_NXTCMDAR	0x80004130	294
HW_APBH_CH2_SEMA	0x80004160	297
HW_APBH_CH3_BAR	0x800041C0	302
HW_APBH_CH3_CMD	0x800041B0	301
HW_APBH_CH3_CURCMDAR	0x80004190	300
HW_APBH_CH3_DEBUG1	0x800041E0	303
HW_APBH_CH3_DEBUG2	0x800041F0	305
HW_APBH_CH3_NXTCMDAR	0x800041A0	300
HW_APBH_CH3_SEMA	0x800041D0	302
HW_APBH_CH4_BAR	0x80004230	309
HW_APBH_CH4_CMD	0x80004220	307
HW_APBH_CH4_CURCMDAR	0x80004200	306
HW_APBH_CH4_DEBUG1	0x80004250	310
HW_APBH_CH4_DEBUG2	0x80004260	312
HW_APBH_CH4_NXTCMDAR	0x80004210	306
HW_APBH_CH4_SEMA	0x80004240	309
HW_APBH_CH5_BAR	0x800042A0	316
HW_APBH_CH5_CMD	0x80004290	314
HW_APBH_CH5_CURCMDAR	0x80004270	313
HW_APBH_CH5_DEBUG1	0x800042C0	318
HW_APBH_CH5_DEBUG2	0x800042D0	319
HW_APBH_CH5_NXTCMDAR	0x80004280	314
HW_APBH_CH5_SEMA	0x800042B0	317
HW_APBH_CH6_BAR	0x80004310	323
HW_APBH_CH6_CMD	0x80004300	321
HW_APBH_CH6_CURCMDAR	0x800042E0	320
HW_APBH_CH6_DEBUG1	0x80004330	325
HW_APBH_CH6_DEBUG2	0x80004340	326
HW_APBH_CH6_NXTCMDAR	0x800042F0	321
HW_APBH_CH6_SEMA	0x80004320	324
HW_APBH_CH7_BAR	0x80004380	330
HW_APBH_CH7_CMD	0x80004370	328
HW_APBH_CH7_CURCMDAR	0x80004350	327
HW_APBH_CH7_DEBUG1	0x800043A0	332
HW_APBH_CH7_DEBUG2	0x800043B0	333
HW_APBH_CH7_NXTCMDAR	0x80004360	328

STMP3770

HW_APBH_CH7_SEMA	0x80004390	331
HW_APBH_CTRL0	0x80004000	274
HW_APBH_CTRL0_CLR	0x80004008	274
HW_APBH_CTRL0_SET	0x80004004	274
HW_APBH_CTRL0_TOG	0x8000400C	274
HW_APBH_CTRL1	0x80004010	275
HW_APBH_CTRL1_CLR	0x80004018	275
HW_APBH_CTRL1_SET	0x80004014	275
HW_APBH_CTRL1_TOG	0x8000401C	275
HW_APBH_DEVSEL	0x80004020	278
HW_APBH_VERSION	0x800043F0	334
HW_APBX_CH0_BAR	0x80024070	348
HW_APBX_CH0_CMD	0x80024060	346
HW_APBX_CH0_CURCMDAR	0x80024040	345
HW_APBX_CH0_DEBUG1	0x80024090	350
HW_APBX_CH0_DEBUG2	0x800240A0	351
HW_APBX_CH0_NXTCMDAR	0x80024050	346
HW_APBX_CH0_SEMA	0x80024080	349
HW_APBX_CH1_BAR	0x800240E0	355
HW_APBX_CH1_CMD	0x800240D0	353
HW_APBX_CH1_CURCMDAR	0x800240B0	352
HW_APBX_CH1_DEBUG1	0x80024100	356
HW_APBX_CH1_DEBUG2	0x80024110	358
HW_APBX_CH1_NXTCMDAR	0x800240C0	353
HW_APBX_CH1_SEMA	0x800240F0	355
HW_APBX_CH2_BAR	0x80024150	361
HW_APBX_CH2_CMD	0x80024140	360
HW_APBX_CH2_CURCMDAR	0x80024120	359
HW_APBX_CH2_DEBUG1	0x80024170	363
HW_APBX_CH2_DEBUG2	0x80024180	364
HW_APBX_CH2_NXTCMDAR	0x80024130	359
HW_APBX_CH2_SEMA	0x80024160	362
HW_APBX_CH3_BAR	0x800241C0	367
HW_APBX_CH3_CMD	0x800241B0	366
HW_APBX_CH3_CURCMDAR	0x80024190	365
HW_APBX_CH3_DEBUG1	0x800241E0	368
HW_APBX_CH3_DEBUG2	0x800241F0	370
HW_APBX_CH3_NXTCMDAR	0x800241A0	365
HW_APBX_CH3_SEMA	0x800241D0	368
HW_APBX_CH4_BAR	0x80024230	373
HW_APBX_CH4_CMD	0x80024220	372
HW_APBX_CH4_CURCMDAR	0x80024200	371
HW_APBX_CH4_DEBUG1	0x80024250	374
HW_APBX_CH4_DEBUG2	0x80024260	376
HW_APBX_CH4_NXTCMDAR	0x80024210	371
HW_APBX_CH4_SEMA	0x80024240	374
HW_APBX_CH5_BAR	0x800242A0	379
HW_APBX_CH5_CMD	0x80024290	378
HW_APBX_CH5_CURCMDAR	0x80024270	377
HW_APBX_CH5_DEBUG1	0x800242C0	380
HW_APBX_CH5_DEBUG2	0x800242D0	382
HW_APBX_CH5_NXTCMDAR	0x80024280	377
HW_APBX_CH5_SEMA	0x800242B0	380
HW_APBX_CH6_BAR	0x80024310	385
HW_APBX_CH6_CMD	0x80024300	384
HW_APBX_CH6_CURCMDAR	0x800242E0	383
HW_APBX_CH6_DEBUG1	0x80024330	386
HW_APBX_CH6_DEBUG2	0x80024340	388
HW_APBX_CH6_NXTCMDAR	0x800242F0	383
HW_APBX_CH6_SEMA	0x80024320	386
HW_APBX_CH7_BAR	0x80024380	391
HW_APBX_CH7_CMD	0x80024370	390
HW_APBX_CH7_CURCMDAR	0x80024350	389

HW_APBX_CH7_DEBUG1	0x800243A0	392
HW_APBX_CH7_DEBUG2	0x800243B0	394
HW_APBX_CH7_NXTCMDAR	0x80024360	389
HW_APBX_CH7_SEMA	0x80024390	392
HW_APBX_CTRL0	0x80024000	341
HW_APBX_CTRL0_CLR	0x80024008	341
HW_APBX_CTRL0_SET	0x80024004	341
HW_APBX_CTRL0_TOG	0x8002400C	341
HW_APBX_CTRL1	0x80024010	342
HW_APBX_CTRL1_CLR	0x80024018	342
HW_APBX_CTRL1_SET	0x80024014	342
HW_APBX_CTRL1_TOG	0x8002401C	342
HW_APBX_DEVSEL	0x80024020	344
HW_APBX_VERSION	0x800243F0	395
HW_ARC_GENERAL	0x80080004	194
HW_AUDIOIN_ADCDEBUG	0x8004C040	733
HW_AUDIOIN_ADCDEBUG_CLR	0x8004C048	733
HW_AUDIOIN_ADCDEBUG_SET	0x8004C044	733
HW_AUDIOIN_ADCDEBUG_TOG	0x8004C04C	733
HW_AUDIOIN_ADCSRR	0x8004C020	730
HW_AUDIOIN_ADCSRR_CLR	0x8004C028	730
HW_AUDIOIN_ADCSRR_SET	0x8004C024	730
HW_AUDIOIN_ADCSRR_TOG	0x8004C02C	730
HW_AUDIOIN_ADCVOL	0x8004C050	735
HW_AUDIOIN_ADCVOL_CLR	0x8004C058	735
HW_AUDIOIN_ADCVOL_SET	0x8004C054	735
HW_AUDIOIN_ADCVOL_TOG	0x8004C05C	735
HW_AUDIOIN_ADCVOLUME	0x8004C030	731
HW_AUDIOIN_ADCVOLUME_CLR	0x8004C038	731
HW_AUDIOIN_ADCVOLUME_SET	0x8004C034	731
HW_AUDIOIN_ADCVOLUME_TOG	0x8004C03C	731
HW_AUDIOIN_ANACLKCTRL	0x8004C070	738
HW_AUDIOIN_ANACLKCTRL_CLR	0x8004C078	738
HW_AUDIOIN_ANACLKCTRL_SET	0x8004C074	738
HW_AUDIOIN_ANACLKCTRL_TOG	0x8004C07C	738
HW_AUDIOIN_CTRL	0x8004C000	726
HW_AUDIOIN_CTRL_CLR	0x8004C008	726
HW_AUDIOIN_CTRL_SET	0x8004C004	726
HW_AUDIOIN_CTRL_TOG	0x8004C00C	726
HW_AUDIOIN_DATA	0x8004C080	739
HW_AUDIOIN_DATA_CLR	0x8004C088	739
HW_AUDIOIN_DATA_SET	0x8004C084	739
HW_AUDIOIN_DATA_TOG	0x8004C08C	739
HW_AUDIOIN_MICLINE	0x8004C060	737
HW_AUDIOIN_MICLINE_CLR	0x8004C068	737
HW_AUDIOIN_MICLINE_SET	0x8004C064	737
HW_AUDIOIN_MICLINE_TOG	0x8004C06C	737
HW_AUDIOIN_STAT	0x8004C010	729
HW_AUDIOIN_STAT_CLR	0x8004C018	729
HW_AUDIOIN_STAT_SET	0x8004C014	729
HW_AUDIOIN_STAT_TOG	0x8004C01C	729
HW_AUDIOOUT_ANACLKCTRL	0x800480e0	770
HW_AUDIOOUT_ANACLKCTRL_CLR	0x800480e8	770
HW_AUDIOOUT_ANACLKCTRL_SET	0x800480e4	770
HW_AUDIOOUT_ANACLKCTRL_TOG	0x800480eC	770
HW_AUDIOOUT_ANACTRL	0x80048090	765
HW_AUDIOOUT_ANACTRL_CLR	0x80048098	765
HW_AUDIOOUT_ANACTRL_SET	0x80048094	765
HW_AUDIOOUT_ANACTRL_TOG	0x8004809C	765
HW_AUDIOOUT_BISTCTRL	0x800480b0	769
HW_AUDIOOUT_BISTCTRL_CLR	0x800480b8	769
HW_AUDIOOUT_BISTCTRL_SET	0x800480b4	769
HW_AUDIOOUT_BISTCTRL_TOG	0x800480bC	769

STMP3770

HW_AUDIOOUT_BISTSTAT0	0x800480c0	769
HW_AUDIOOUT_BISTSTAT0_CLR	0x800480c8	769
HW_AUDIOOUT_BISTSTAT0_SET	0x800480c4	769
HW_AUDIOOUT_BISTSTAT0_TOG	0x800480cC	769
HW_AUDIOOUT_BISTSTAT1	0x800480d0	770
HW_AUDIOOUT_BISTSTAT1_CLR	0x800480d8	770
HW_AUDIOOUT_BISTSTAT1_SET	0x800480d4	770
HW_AUDIOOUT_BISTSTAT1_TOG	0x800480dC	770
HW_AUDIOOUT_CTRL	0x80048000	751
HW_AUDIOOUT_CTRL_CLR	0x80048008	751
HW_AUDIOOUT_CTRL_SET	0x80048004	751
HW_AUDIOOUT_CTRL_TOG	0x8004800C	751
HW_AUDIOOUT_DACDEBUG	0x80048040	758
HW_AUDIOOUT_DACDEBUG_CLR	0x80048048	758
HW_AUDIOOUT_DACDEBUG_SET	0x80048044	758
HW_AUDIOOUT_DACDEBUG_TOG	0x8004804C	758
HW_AUDIOOUT_DACSRR	0x80048020	754
HW_AUDIOOUT_DACSRR_CLR	0x80048028	754
HW_AUDIOOUT_DACSRR_SET	0x80048024	754
HW_AUDIOOUT_DACSRR_TOG	0x8004802C	754
HW_AUDIOOUT_DACVOLUME	0x80048030	756
HW_AUDIOOUT_DACVOLUME_CLR	0x80048038	756
HW_AUDIOOUT_DACVOLUME_SET	0x80048034	756
HW_AUDIOOUT_DACVOLUME_TOG	0x8004803C	756
HW_AUDIOOUT_DATA	0x800480f0	771
HW_AUDIOOUT_DATA_CLR	0x800480f8	771
HW_AUDIOOUT_DATA_SET	0x800480f4	771
HW_AUDIOOUT_DATA_TOG	0x800480fC	771
HW_AUDIOOUT_HPVOL	0x80048050	759
HW_AUDIOOUT_HPVOL_CLR	0x80048058	759
HW_AUDIOOUT_HPVOL_SET	0x80048054	759
HW_AUDIOOUT_HPVOL_TOG	0x8004805C	759
HW_AUDIOOUT_LINEOUTCTRL	0x80048100	772
HW_AUDIOOUT_LINEOUTCTRL_CLR	0x80048108	772
HW_AUDIOOUT_LINEOUTCTRL_SET	0x80048104	772
HW_AUDIOOUT_LINEOUTCTRL_TOG	0x8004810C	772
HW_AUDIOOUT_PWRDN	0x80048070	761
HW_AUDIOOUT_PWRDN_CLR	0x80048078	761
HW_AUDIOOUT_PWRDN_SET	0x80048074	761
HW_AUDIOOUT_PWRDN_TOG	0x8004807C	761
HW_AUDIOOUT_REFCTRL	0x80048080	762
HW_AUDIOOUT_REFCTRL_CLR	0x80048088	762
HW_AUDIOOUT_REFCTRL_SET	0x80048084	762
HW_AUDIOOUT_REFCTRL_TOG	0x8004808C	762
HW_AUDIOOUT_RESERVED	0x80048060	761
HW_AUDIOOUT_RESERVED_CLR	0x80048068	761
HW_AUDIOOUT_RESERVED_SET	0x80048064	761
HW_AUDIOOUT_RESERVED_TOG	0x8004806C	761
HW_AUDIOOUT_STAT	0x80048010	753
HW_AUDIOOUT_STAT_CLR	0x80048018	753
HW_AUDIOOUT_STAT_SET	0x80048014	753
HW_AUDIOOUT_STAT_TOG	0x8004801C	754
HW_AUDIOOUT_TEST	0x800480a0	767
HW_AUDIOOUT_TEST_CLR	0x800480a8	767
HW_AUDIOOUT_TEST_SET	0x800480a4	767
HW_AUDIOOUT_TEST_TOG	0x800480aC	767
HW_AUDIOOUT_VERSION	0x80048200	774
HW_CLKCTRL_CLKSEQ	0x800400E0	72
HW_CLKCTRL_CLKSEQ_CLR	0x800400E8	72
HW_CLKCTRL_CLKSEQ_SET	0x800400E4	72
HW_CLKCTRL_CLKSEQ_TOG	0x800400EC	72
HW_CLKCTRL_CPU	0x80040020	61
HW_CLKCTRL_CPU_CLR	0x80040028	61

HW_CLKCTRL_CPU_SET	0x80040024	61
HW_CLKCTRL_CPU_TOG	0x8004002c	61
HW_CLKCTRL_FRAC	0x800400D0	70
HW_CLKCTRL_FRAC_CLR	0x800400D8	70
HW_CLKCTRL_FRAC_SET	0x800400D4	70
HW_CLKCTRL_FRAC_TOG	0x800400DC	70
HW_CLKCTRL_GPMI	0x80040080	68
HW_CLKCTRL_HBUS	0x80040030	62
HW_CLKCTRL_HBUS_CLR	0x80040038	62
HW_CLKCTRL_HBUS_SET	0x80040034	62
HW_CLKCTRL_HBUS_TOG	0x8004003C	62
HW_CLKCTRL_PIX	0x80040060	66
HW_CLKCTRL_PLLCTRL0	0x80040000	59
HW_CLKCTRL_PLLCTRL0_CLR	0x80040008	59
HW_CLKCTRL_PLLCTRL0_SET	0x80040004	59
HW_CLKCTRL_PLLCTRL0_TOG	0x8004000C	59
HW_CLKCTRL_PLLCTRL1	0x80040010	61
HW_CLKCTRL_RESET	0x800400F0	73
HW_CLKCTRL_SPDIF	0x80040090	69
HW_CLKCTRL_SSP	0x80040070	67
HW_CLKCTRL_VERSION	0x80040100	74
HW_CLKCTRL_XBUS	0x80040040	64
HW_CLKCTRL_XTAL	0x80040050	65
HW_CLKCTRL_XTAL_CLR	0x80040058	65
HW_CLKCTRL_XTAL_SET	0x80040054	65
HW_CLKCTRL_XTAL_TOG	0x8004005C	65
HW_DCP_CAPABILITY0	0x80028030	488
HW_DCP_CAPABILITY1	0x80028040	489
HW_DCP_CH0CMDPTR	0x80028100	496
HW_DCP_CH0OPTS	0x80028130	499
HW_DCP_CH0OPTS_CLR	0x80028138	499
HW_DCP_CH0OPTS_SET	0x80028134	499
HW_DCP_CH0OPTS_TOG	0x8002813C	499
HW_DCP_CH0SEMA	0x80028110	497
HW_DCP_CH0STAT	0x80028120	498
HW_DCP_CH0STAT_CLR	0x80028128	498
HW_DCP_CH0STAT_SET	0x80028124	498
HW_DCP_CH0STAT_TOG	0x8002812C	498
HW_DCP_CH1CMDPTR	0x80028140	500
HW_DCP_CH1OPTS	0x80028170	504
HW_DCP_CH1OPTS_CLR	0x80028178	504
HW_DCP_CH1OPTS_SET	0x80028174	504
HW_DCP_CH1OPTS_TOG	0x8002817C	504
HW_DCP_CH1SEMA	0x80028150	501
HW_DCP_CH1STAT	0x80028160	502
HW_DCP_CH1STAT_CLR	0x80028168	502
HW_DCP_CH1STAT_SET	0x80028164	502
HW_DCP_CH1STAT_TOG	0x8002816C	502
HW_DCP_CH2CMDPTR	0x80028180	504
HW_DCP_CH2OPTS	0x800281B0	508
HW_DCP_CH2OPTS_CLR	0x800281B8	508
HW_DCP_CH2OPTS_SET	0x800281B4	508
HW_DCP_CH2OPTS_TOG	0x800281BC	508
HW_DCP_CH2SEMA	0x80028190	505
HW_DCP_CH2STAT	0x800281A0	506
HW_DCP_CH2STAT_CLR	0x800281A8	506
HW_DCP_CH2STAT_SET	0x800281A4	506
HW_DCP_CH2STAT_TOG	0x800281AC	506
HW_DCP_CH3CMDPTR	0x800281C0	508
HW_DCP_CH3OPTS	0x800281F0	512
HW_DCP_CH3OPTS_CLR	0x800281F8	512
HW_DCP_CH3OPTS_SET	0x800281F4	512
HW_DCP_CH3OPTS_TOG	0x800281FC	512

STMP3770

HW_DCP_CH3SEMA	0x800281D0	509
HW_DCP_CH3STAT	0x800281E0	510
HW_DCP_CH3STAT_CLR	0x800281E8	510
HW_DCP_CH3STAT_SET	0x800281E4	510
HW_DCP_CH3STAT_TOG	0x800281EC	510
HW_DCP_CHANNELCTRL	0x80028020	487
HW_DCP_CHANNELCTRL_CLR	0x80028028	487
HW_DCP_CHANNELCTRL_SET	0x80028024	487
HW_DCP_CHANNELCTRL_TOG	0x8002802C	487
HW_DCP_CONTEXT	0x80028050	489
HW_DCP_CSCCHROMAU	0x80028360	517
HW_DCP_CSCCHROMAV	0x80028370	518
HW_DCP_CSCCOEFF0	0x80028380	518
HW_DCP_CSCCOEFF1	0x80028390	519
HW_DCP_CSCCOEFF2	0x800283A0	520
HW_DCP_CSCCTRL0	0x80028300	512
HW_DCP_CSCCTRL0_CLR	0x80028308	512
HW_DCP_CSCCTRL0_SET	0x80028304	512
HW_DCP_CSCCTRL0_TOG	0x8002830C	512
HW_DCP_CSCINBUFPARAM	0x80028330	515
HW_DCP_CSCLUMA	0x80028350	517
HW_DCP_CSCOUTBUFPARAM	0x80028320	515
HW_DCP_CSCRGB	0x80028340	516
HW_DCP_CSCSTAT	0x80028310	514
HW_DCP_CSCSTAT_CLR	0x80028318	514
HW_DCP_CSCSTAT_SET	0x80028314	514
HW_DCP_CSCSTAT_TOG	0x8002831C	514
HW_DCP_CSCXSCALE	0x800283E0	520
HW_DCP_CSCYSCALE	0x800283F0	521
HW_DCP_CTRL	0x80028000	484
HW_DCP_CTRL_CLR	0x80028008	484
HW_DCP_CTRL_SET	0x80028004	484
HW_DCP_CTRL_TOG	0x8002800C	484
HW_DCP_DBGDATA	0x80028410	522
HW_DCP_DBGSELECT	0x80028400	522
HW_DCP_KEY	0x80028060	490
HW_DCP_KEYDATA	0x80028070	491
HW_DCP_PACKET0	0x80028080	491
HW_DCP_PACKET1	0x80028090	492
HW_DCP_PACKET2	0x800280A0	493
HW_DCP_PACKET3	0x800280B0	494
HW_DCP_PACKET4	0x800280C0	495
HW_DCP_PACKET5	0x800280D0	495
HW_DCP_PACKET6	0x800280E0	496
HW_DCP_STAT	0x80028010	485
HW_DCP_STAT_CLR	0x80028018	485
HW_DCP_STAT_SET	0x80028014	485
HW_DCP_STAT_TOG	0x8002801C	485
HW_DCP_VERSION	0x80028420	523
HW_DIGCTL_AHB_STATS_SELECT	0x8001C330	156
HW_DIGCTL_ARMCACHE	0x8001C2B0	153
HW_DIGCTL_CHIPID	0x8001C310	155
HW_DIGCTL_CTRL	0x8001C000	131
HW_DIGCTL_CTRL_CLR	0x8001C008	131
HW_DIGCTL_CTRL_SET	0x8001C004	131
HW_DIGCTL_CTRL_TOG	0x8001C00C	131
HW_DIGCTL_DBG	0x8001C0E0	143
HW_DIGCTL_DBGDRD	0x8001C0D0	142
HW_DIGCTL_DEBUG_TRAP_ADDR_HIGH	0x8001C2D0	155
HW_DIGCTL_DEBUG_TRAP_ADDR_LOW	0x8001C2C0	154
HW_DIGCTL_ENTROPY	0x8001C090	139
HW_DIGCTL_ENTROPY_LATCHED	0x8001C0A0	140
HW_DIGCTL_HCLKCOUNT	0x8001C020	137

HW_DIGCTL_HCLKCOUNT_CLR	0x8001C028	137
HW_DIGCTL_HCLKCOUNT_SET	0x8001C024	137
HW_DIGCTL_HCLKCOUNT_TOG	0x8001C02C	137
HW_DIGCTL_L0_AHB_ACTIVE_CYCLES	0x8001C340	157
HW_DIGCTL_L0_AHB_DATA_CYCLES	0x8001C360	158
HW_DIGCTL_L0_AHB_DATA_STALLED	0x8001C350	157
HW_DIGCTL_L1_AHB_ACTIVE_CYCLES	0x8001C370	158
HW_DIGCTL_L1_AHB_DATA_CYCLES	0x8001C390	159
HW_DIGCTL_L1_AHB_DATA_STALLED	0x8001C380	159
HW_DIGCTL_L2_AHB_ACTIVE_CYCLES	0x8001C3A0	160
HW_DIGCTL_L2_AHB_DATA_CYCLES	0x8001C3C0	161
HW_DIGCTL_L2_AHB_DATA_STALLED	0x8001C3B0	160
HW_DIGCTL_L3_AHB_ACTIVE_CYCLES	0x8001C3D0	162
HW_DIGCTL_L3_AHB_DATA_CYCLES	0x8001C3F0	163
HW_DIGCTL_L3_AHB_DATA_STALLED	0x8001C3E0	162
HW_DIGCTL_MICROSECONDS	0x8001C0C0	142
HW_DIGCTL_MICROSECONDS_CLR	0x8001C0C8	142
HW_DIGCTL_MICROSECONDS_SET	0x8001C0C4	142
HW_DIGCTL_MICROSECONDS_TOG	0x8001C0CC	142
HW_DIGCTL_MPTE0_LOC	0x8001C400	163
HW_DIGCTL_MPTE1_LOC	0x8001C410	164
HW_DIGCTL_MPTE2_LOC	0x8001C420	164
HW_DIGCTL_MPTE3_LOC	0x8001C430	165
HW_DIGCTL_MPTE4_LOC	0x8001C440	166
HW_DIGCTL_MPTE5_LOC	0x8001C450	166
HW_DIGCTL_MPTE6_LOC	0x8001C460	167
HW_DIGCTL_MPTE7_LOC	0x8001C470	167
HW_DIGCTL_OCRAM_BIST_CSR	0x8001C0F0	143
HW_DIGCTL_OCRAM_BIST_CSR_CLR	0x8001C0F8	143
HW_DIGCTL_OCRAM_BIST_CSR_SET	0x8001C0F4	143
HW_DIGCTL_OCRAM_BIST_CSR_TOG	0x8001C0FC	143
HW_DIGCTL_OCRAM_STATUS0	0x8001C110	144
HW_DIGCTL_OCRAM_STATUS0_CLR	0x8001C118	144
HW_DIGCTL_OCRAM_STATUS0_SET	0x8001C114	144
HW_DIGCTL_OCRAM_STATUS0_TOG	0x8001C11C	144
HW_DIGCTL_OCRAM_STATUS1	0x8001C120	145
HW_DIGCTL_OCRAM_STATUS1_CLR	0x8001C128	145
HW_DIGCTL_OCRAM_STATUS1_SET	0x8001C124	145
HW_DIGCTL_OCRAM_STATUS1_TOG	0x8001C12C	145
HW_DIGCTL_OCRAM_STATUS10	0x8001C1B0	150
HW_DIGCTL_OCRAM_STATUS10_CLR	0x8001C1B8	150
HW_DIGCTL_OCRAM_STATUS10_SET	0x8001C1B4	150
HW_DIGCTL_OCRAM_STATUS10_TOG	0x8001C1BC	150
HW_DIGCTL_OCRAM_STATUS11	0x8001C1C0	150
HW_DIGCTL_OCRAM_STATUS11_CLR	0x8001C1C8	150
HW_DIGCTL_OCRAM_STATUS11_SET	0x8001C1C4	150
HW_DIGCTL_OCRAM_STATUS11_TOG	0x8001C1CC	150
HW_DIGCTL_OCRAM_STATUS12	0x8001C1D0	151
HW_DIGCTL_OCRAM_STATUS12_CLR	0x8001C1D8	151
HW_DIGCTL_OCRAM_STATUS12_SET	0x8001C1D4	151
HW_DIGCTL_OCRAM_STATUS12_TOG	0x8001C1DC	151
HW_DIGCTL_OCRAM_STATUS13	0x8001C1E0	152
HW_DIGCTL_OCRAM_STATUS13_CLR	0x8001C1E8	152
HW_DIGCTL_OCRAM_STATUS13_SET	0x8001C1E4	152
HW_DIGCTL_OCRAM_STATUS13_TOG	0x8001C1EC	152
HW_DIGCTL_OCRAM_STATUS2	0x8001C130	145
HW_DIGCTL_OCRAM_STATUS2_CLR	0x8001C138	145
HW_DIGCTL_OCRAM_STATUS2_SET	0x8001C134	145
HW_DIGCTL_OCRAM_STATUS2_TOG	0x8001C13C	145
HW_DIGCTL_OCRAM_STATUS3	0x8001C140	146
HW_DIGCTL_OCRAM_STATUS3_CLR	0x8001C148	146
HW_DIGCTL_OCRAM_STATUS3_SET	0x8001C144	146
HW_DIGCTL_OCRAM_STATUS3_TOG	0x8001C14C	146

STMP3770

HW_DIGCTL_OCRAM_STATUS4	0x8001C150	146
HW_DIGCTL_OCRAM_STATUS4_CLR	0x8001C158	146
HW_DIGCTL_OCRAM_STATUS4_SET	0x8001C154	146
HW_DIGCTL_OCRAM_STATUS4_TOG	0x8001C15C	146
HW_DIGCTL_OCRAM_STATUS5	0x8001C160	147
HW_DIGCTL_OCRAM_STATUS5_CLR	0x8001C168	147
HW_DIGCTL_OCRAM_STATUS5_SET	0x8001C164	147
HW_DIGCTL_OCRAM_STATUS5_TOG	0x8001C16C	147
HW_DIGCTL_OCRAM_STATUS6	0x8001C170	147
HW_DIGCTL_OCRAM_STATUS6_CLR	0x8001C178	147
HW_DIGCTL_OCRAM_STATUS6_SET	0x8001C174	147
HW_DIGCTL_OCRAM_STATUS6_TOG	0x8001C17C	147
HW_DIGCTL_OCRAM_STATUS7	0x8001C180	148
HW_DIGCTL_OCRAM_STATUS7_CLR	0x8001C188	148
HW_DIGCTL_OCRAM_STATUS7_SET	0x8001C184	148
HW_DIGCTL_OCRAM_STATUS7_TOG	0x8001C18C	148
HW_DIGCTL_OCRAM_STATUS8	0x8001C190	148
HW_DIGCTL_OCRAM_STATUS8_CLR	0x8001C198	148
HW_DIGCTL_OCRAM_STATUS8_SET	0x8001C194	148
HW_DIGCTL_OCRAM_STATUS8_TOG	0x8001C19C	148
HW_DIGCTL_OCRAM_STATUS9	0x8001C1A0	149
HW_DIGCTL_OCRAM_STATUS9_CLR	0x8001C1A8	149
HW_DIGCTL_OCRAM_STATUS9_SET	0x8001C1A4	149
HW_DIGCTL_OCRAM_STATUS9_TOG	0x8001C1AC	149
HW_DIGCTL_RAMCTRL	0x8001C030	137
HW_DIGCTL_RAMCTRL_CLR	0x8001C038	137
HW_DIGCTL_RAMCTRL_SET	0x8001C034	137
HW_DIGCTL_RAMCTRL_TOG	0x8001C03C	137
HW_DIGCTL_RAMREPAIR	0x8001C040	138
HW_DIGCTL_RAMREPAIR_CLR	0x8001C048	138
HW_DIGCTL_RAMREPAIR_SET	0x8001C044	138
HW_DIGCTL_RAMREPAIR_TOG	0x8001C04C	138
HW_DIGCTL_ROMCTRL	0x8001C050	138
HW_DIGCTL_ROMCTRL_CLR	0x8001C058	138
HW_DIGCTL_ROMCTRL_SET	0x8001C054	138
HW_DIGCTL_ROMCTRL_TOG	0x8001C05C	139
HW_DIGCTL_SCRATCH0	0x8001C290	153
HW_DIGCTL_SCRATCH1	0x8001C2A0	153
HW_DIGCTL_SJTAGDBG	0x8001C0B0	140
HW_DIGCTL_SJTAGDBG_CLR	0x8001C0B8	140
HW_DIGCTL_SJTAGDBG_SET	0x8001C0B4	140
HW_DIGCTL_SJTAGDBG_TOG	0x8001C0BC	140
HW_DIGCTL_STATUS	0x8001C010	135
HW_DIGCTL_STATUS_CLR	0x8001C018	135
HW_DIGCTL_STATUS_SET	0x8001C014	135
HW_DIGCTL_STATUS_TOG	0x8001C01C	135
HW_DIGCTL_WRITEONCE	0x8001C060	139
HW_DRI_CTRL	0x80074000	792
HW_DRI_CTRL_CLR	0x80074008	792
HW_DRI_CTRL_SET	0x80074004	792
HW_DRI_CTRL_TOG	0x8007400C	792
HW_DRI_DATA	0x80074030	797
HW_DRI_DEBUG0	0x80074040	797
HW_DRI_DEBUG0_CLR	0x80074048	797
HW_DRI_DEBUG0_SET	0x80074044	797
HW_DRI_DEBUG0_TOG	0x8007404C	797
HW_DRI_DEBUG1	0x80074050	799
HW_DRI_DEBUG1_CLR	0x80074058	799
HW_DRI_DEBUG1_SET	0x80074054	799
HW_DRI_DEBUG1_TOG	0x8007405C	799
HW_DRI_STAT	0x80074020	796
HW_DRI_TIMING	0x80074010	795
HW_DRI_VERSION	0x80074060	800

HW_ECC8_BLOCKNAME	0x80008080	454
HW_ECC8_CTRL	0x80008000	445
HW_ECC8_CTRL_CLR	0x80008008	445
HW_ECC8_CTRL_SET	0x80008004	445
HW_ECC8_CTRL_TOG	0x8000800C	445
HW_ECC8_DBGAHBMREAD	0x80008070	454
HW_ECC8_DBGCSFEREAD	0x80008050	453
HW_ECC8_DBGKESREAD	0x80008040	453
HW_ECC8_DBGSYNDGENREAD	0x80008060	453
HW_ECC8_DEBUG0	0x80008030	450
HW_ECC8_DEBUG0_CLR	0x80008038	450
HW_ECC8_DEBUG0_SET	0x80008034	450
HW_ECC8_DEBUG0_TOG	0x8000803C	450
HW_ECC8_STATUS0	0x80008010	447
HW_ECC8_STATUS1	0x80008020	448
HW_ECC8_VERSION	0x800080a0	455
HW_GPMI_AUXILIARY	0x8000C050	408
HW_GPMI_COMPARE	0x8000C010	405
HW_GPMI_CTRL0	0x8000C000	403
HW_GPMI_CTRL0_CLR	0x8000C008	403
HW_GPMI_CTRL0_SET	0x8000C004	403
HW_GPMI_CTRL0_TOG	0x8000C00C	403
HW_GPMI_CTRL1	0x8000C060	409
HW_GPMI_CTRL1_CLR	0x8000C068	409
HW_GPMI_CTRL1_SET	0x8000C064	409
HW_GPMI_CTRL1_TOG	0x8000C06C	409
HW_GPMI_DATA	0x8000C0A0	413
HW_GPMI_DEBUG	0x8000C0C0	415
HW_GPMI_ECCCOUNT	0x8000C030	407
HW_GPMI_ECCCTRL	0x8000C020	405
HW_GPMI_ECCCTRL_CLR	0x8000C028	405
HW_GPMI_ECCCTRL_SET	0x8000C024	405
HW_GPMI_ECCCTRL_TOG	0x8000C02C	405
HW_GPMI_PAYLOAD	0x8000C040	408
HW_GPMI_STAT	0x8000C0B0	413
HW_GPMI_TIMING0	0x8000C070	410
HW_GPMI_TIMING1	0x8000C080	411
HW_GPMI_TIMING2	0x8000C090	412
HW_GPMI_VERSION	0x8000C0D0	416
HW_I2C_CTRL0	0x80058000	669
HW_I2C_CTRL0_CLR	0x80058008	669
HW_I2C_CTRL0_SET	0x80058004	669
HW_I2C_CTRL0_TOG	0x8005800C	669
HW_I2C_CTRL1	0x80058040	674
HW_I2C_CTRL1_CLR	0x80058048	674
HW_I2C_CTRL1_SET	0x80058044	674
HW_I2C_CTRL1_TOG	0x8005804C	674
HW_I2C_DATA	0x80058060	681
HW_I2C_DEBUG0	0x80058070	681
HW_I2C_DEBUG0_CLR	0x80058078	681
HW_I2C_DEBUG0_SET	0x80058074	681
HW_I2C_DEBUG0_TOG	0x8005807C	681
HW_I2C_DEBUG1	0x80058080	682
HW_I2C_DEBUG1_CLR	0x80058088	682
HW_I2C_DEBUG1_SET	0x80058084	682
HW_I2C_DEBUG1_TOG	0x8005808C	682
HW_I2C_STAT	0x80058050	677
HW_I2C_TIMING0	0x80058010	672
HW_I2C_TIMING0_CLR	0x80058018	672
HW_I2C_TIMING0_SET	0x80058014	672
HW_I2C_TIMING0_TOG	0x8005801C	672
HW_I2C_TIMING1	0x80058020	672
HW_I2C_TIMING1_CLR	0x80058028	672

STMP3770

HW_I2C_TIMING1_SET	0x80058024	672
HW_I2C_TIMING1_TOG	0x8005802C	672
HW_I2C_TIMING2	0x80058030	673
HW_I2C_TIMING2_CLR	0x80058038	673
HW_I2C_TIMING2_SET	0x80058034	673
HW_I2C_TIMING2_TOG	0x8005803C	673
HW_I2C_VERSION	0x80058090	684
HW_ICOLL_CTRL	0x80000020	85
HW_ICOLL_CTRL_CLR	0x80000028	85
HW_ICOLL_CTRL_SET	0x80000024	85
HW_ICOLL_CTRL_TOG	0x8000002C	85
HW_ICOLL_DBGFLAG	0x800001A0	121
HW_ICOLL_DBGFLAG_CLR	0x800001A8	121
HW_ICOLL_DBGFLAG_SET	0x800001A4	121
HW_ICOLL_DBGFLAG_TOG	0x800001AC	121
HW_ICOLL_DBGREAD0	0x80000180	120
HW_ICOLL_DBGREAD0_CLR	0x80000188	120
HW_ICOLL_DBGREAD0_SET	0x80000184	120
HW_ICOLL_DBGREAD0_TOG	0x8000018C	120
HW_ICOLL_DBGREAD1	0x80000190	120
HW_ICOLL_DBGREAD1_CLR	0x80000198	120
HW_ICOLL_DBGREAD1_SET	0x80000194	120
HW_ICOLL_DBGREAD1_TOG	0x8000019C	120
HW_ICOLL_DBGREQUEST0	0x800001B0	121
HW_ICOLL_DBGREQUEST0_CLR	0x800001B8	121
HW_ICOLL_DBGREQUEST0_SET	0x800001B4	121
HW_ICOLL_DBGREQUEST0_TOG	0x800001BC	121
HW_ICOLL_DBGREQUEST1	0x800001C0	122
HW_ICOLL_DBGREQUEST1_CLR	0x800001C8	122
HW_ICOLL_DBGREQUEST1_SET	0x800001C4	122
HW_ICOLL_DBGREQUEST1_TOG	0x800001CC	122
HW_ICOLL_DEBUG	0x80000170	118
HW_ICOLL_DEBUG_CLR	0x80000178	118
HW_ICOLL_DEBUG_SET	0x80000174	118
HW_ICOLL_DEBUG_TOG	0x8000017C	118
HW_ICOLL_LEVELACK	0x80000010	84
HW_ICOLL_PRIORITY0	0x80000060	90
HW_ICOLL_PRIORITY0_CLR	0x80000068	90
HW_ICOLL_PRIORITY0_SET	0x80000064	90
HW_ICOLL_PRIORITY0_TOG	0x8000006C	90
HW_ICOLL_PRIORITY1	0x80000070	92
HW_ICOLL_PRIORITY1_CLR	0x80000078	92
HW_ICOLL_PRIORITY1_SET	0x80000074	92
HW_ICOLL_PRIORITY1_TOG	0x8000007C	92
HW_ICOLL_PRIORITY10	0x80000100	107
HW_ICOLL_PRIORITY10_CLR	0x80000108	107
HW_ICOLL_PRIORITY10_SET	0x80000104	107
HW_ICOLL_PRIORITY10_TOG	0x8000010C	107
HW_ICOLL_PRIORITY11	0x80000110	109
HW_ICOLL_PRIORITY11_CLR	0x80000118	109
HW_ICOLL_PRIORITY11_SET	0x80000114	109
HW_ICOLL_PRIORITY11_TOG	0x8000011C	109
HW_ICOLL_PRIORITY12	0x80000120	111
HW_ICOLL_PRIORITY12_CLR	0x80000128	111
HW_ICOLL_PRIORITY12_SET	0x80000124	111
HW_ICOLL_PRIORITY12_TOG	0x8000012C	111
HW_ICOLL_PRIORITY13	0x80000130	112
HW_ICOLL_PRIORITY13_CLR	0x80000138	112
HW_ICOLL_PRIORITY13_SET	0x80000134	112
HW_ICOLL_PRIORITY13_TOG	0x8000013C	112
HW_ICOLL_PRIORITY14	0x80000140	114
HW_ICOLL_PRIORITY14_CLR	0x80000148	114
HW_ICOLL_PRIORITY14_SET	0x80000144	114

HW_ICOLL_PRIORITY14_TOG	0x8000014C	114
HW_ICOLL_PRIORITY15	0x80000150	116
HW_ICOLL_PRIORITY15_CLR	0x80000158	116
HW_ICOLL_PRIORITY15_SET	0x80000154	116
HW_ICOLL_PRIORITY15_TOG	0x8000015C	116
HW_ICOLL_PRIORITY2	0x80000080	93
HW_ICOLL_PRIORITY2_CLR	0x80000088	93
HW_ICOLL_PRIORITY2_SET	0x80000084	93
HW_ICOLL_PRIORITY2_TOG	0x8000008C	93
HW_ICOLL_PRIORITY3	0x80000090	95
HW_ICOLL_PRIORITY3_CLR	0x80000098	95
HW_ICOLL_PRIORITY3_SET	0x80000094	95
HW_ICOLL_PRIORITY3_TOG	0x8000009C	95
HW_ICOLL_PRIORITY4	0x800000A0	97
HW_ICOLL_PRIORITY4_CLR	0x800000A8	97
HW_ICOLL_PRIORITY4_SET	0x800000A4	97
HW_ICOLL_PRIORITY4_TOG	0x800000AC	97
HW_ICOLL_PRIORITY5	0x800000B0	98
HW_ICOLL_PRIORITY5_CLR	0x800000B8	98
HW_ICOLL_PRIORITY5_SET	0x800000B4	98
HW_ICOLL_PRIORITY5_TOG	0x800000BC	98
HW_ICOLL_PRIORITY6	0x800000C0	101
HW_ICOLL_PRIORITY6_CLR	0x800000C8	101
HW_ICOLL_PRIORITY6_SET	0x800000C4	101
HW_ICOLL_PRIORITY6_TOG	0x800000CC	101
HW_ICOLL_PRIORITY7	0x800000D0	102
HW_ICOLL_PRIORITY7_CLR	0x800000D8	102
HW_ICOLL_PRIORITY7_SET	0x800000D4	102
HW_ICOLL_PRIORITY7_TOG	0x800000DC	102
HW_ICOLL_PRIORITY8	0x800000E0	104
HW_ICOLL_PRIORITY8_CLR	0x800000E8	104
HW_ICOLL_PRIORITY8_SET	0x800000E4	104
HW_ICOLL_PRIORITY8_TOG	0x800000EC	104
HW_ICOLL_PRIORITY9	0x800000F0	106
HW_ICOLL_PRIORITY9_CLR	0x800000F8	106
HW_ICOLL_PRIORITY9_SET	0x800000F4	106
HW_ICOLL_PRIORITY9_TOG	0x800000FC	106
HW_ICOLL_RAW0	0x80000040	89
HW_ICOLL_RAW0_CLR	0x80000048	89
HW_ICOLL_RAW0_SET	0x80000044	89
HW_ICOLL_RAW0_TOG	0x8000004C	89
HW_ICOLL_RAW1	0x80000050	89
HW_ICOLL_RAW1_CLR	0x80000058	89
HW_ICOLL_RAW1_SET	0x80000054	89
HW_ICOLL_RAW1_TOG	0x8000005C	89
HW_ICOLL_STAT	0x80000030	88
HW_ICOLL_VBASE	0x80000160	118
HW_ICOLL_VBASE_CLR	0x80000168	118
HW_ICOLL_VBASE_SET	0x80000164	118
HW_ICOLL_VBASE_TOG	0x8000016C	118
HW_ICOLL_VECTOR	0x80000000	84
HW_ICOLL_VECTOR_CLR	0x80000008	84
HW_ICOLL_VECTOR_SET	0x80000004	84
HW_ICOLL_VECTOR_TOG	0x8000000C	84
HW_ICOLL_VERSION	0x800001D0	122
HW_LCDIF_CTRL	0x80030000	576
HW_LCDIF_CTRL_CLR	0x80030008	576
HW_LCDIF_CTRL_SET	0x80030004	576
HW_LCDIF_CTRL_TOG	0x8003000C	576
HW_LCDIF_CTRL1	0x80030010	578
HW_LCDIF_CTRL1_CLR	0x80030018	578
HW_LCDIF_CTRL1_SET	0x80030014	578
HW_LCDIF_CTRL1_TOG	0x8003001C	578

STMP3770

HW_LCDIF_DATA	0x800300b0	591
HW_LCDIF_DEBUG0	0x800300e0	593
HW_LCDIF_DVICTRL0	0x80030070	587
HW_LCDIF_DVICTRL1	0x80030080	588
HW_LCDIF_DVICTRL2	0x80030090	589
HW_LCDIF_DVICTRL3	0x800300a0	590
HW_LCDIF_STAT	0x800300c0	592
HW_LCDIF_TIMING	0x80030020	581
HW_LCDIF_VDCTRL0	0x80030030	582
HW_LCDIF_VDCTRL0_CLR	0x80030038	582
HW_LCDIF_VDCTRL0_SET	0x80030034	582
HW_LCDIF_VDCTRL0_TOG	0x8003003C	582
HW_LCDIF_VDCTRL1	0x80030040	584
HW_LCDIF_VDCTRL2	0x80030050	585
HW_LCDIF_VDCTRL3	0x80030060	586
HW_LCDIF_VERSION	0x800300d0	593
HW_LRADC_CH0	0x80050050	818
HW_LRADC_CH0_CLR	0x80050058	818
HW_LRADC_CH0_SET	0x80050054	818
HW_LRADC_CH0_TOG	0x8005005C	818
HW_LRADC_CH1	0x80050060	819
HW_LRADC_CH1_CLR	0x80050068	819
HW_LRADC_CH1_SET	0x80050064	819
HW_LRADC_CH1_TOG	0x8005006C	819
HW_LRADC_CH2	0x80050070	821
HW_LRADC_CH2_CLR	0x80050078	821
HW_LRADC_CH2_SET	0x80050074	821
HW_LRADC_CH2_TOG	0x8005007C	821
HW_LRADC_CH3	0x80050080	822
HW_LRADC_CH3_CLR	0x80050088	822
HW_LRADC_CH3_SET	0x80050084	822
HW_LRADC_CH3_TOG	0x8005008C	822
HW_LRADC_CH4	0x80050090	823
HW_LRADC_CH4_CLR	0x80050098	823
HW_LRADC_CH4_SET	0x80050094	823
HW_LRADC_CH4_TOG	0x8005009C	823
HW_LRADC_CH5	0x800500A0	824
HW_LRADC_CH5_CLR	0x800500A8	824
HW_LRADC_CH5_SET	0x800500A4	824
HW_LRADC_CH5_TOG	0x800500AC	824
HW_LRADC_CH6	0x800500B0	826
HW_LRADC_CH6_CLR	0x800500B8	826
HW_LRADC_CH6_SET	0x800500B4	826
HW_LRADC_CH6_TOG	0x800500BC	826
HW_LRADC_CH7	0x800500C0	827
HW_LRADC_CH7_CLR	0x800500C8	827
HW_LRADC_CH7_SET	0x800500C4	827
HW_LRADC_CH7_TOG	0x800500CC	827
HW_LRADC_CONVERSION	0x80050130	836
HW_LRADC_CONVERSION_CLR	0x80050138	836
HW_LRADC_CONVERSION_SET	0x80050134	836
HW_LRADC_CONVERSION_TOG	0x8005013C	836
HW_LRADC_CTRL0	0x80050000	807
HW_LRADC_CTRL0_CLR	0x80050008	807
HW_LRADC_CTRL0_SET	0x80050004	807
HW_LRADC_CTRL0_TOG	0x8005000C	807
HW_LRADC_CTRL1	0x80050010	809
HW_LRADC_CTRL1_CLR	0x80050018	809
HW_LRADC_CTRL1_SET	0x80050014	809
HW_LRADC_CTRL1_TOG	0x8005001C	809
HW_LRADC_CTRL2	0x80050020	812
HW_LRADC_CTRL2_CLR	0x80050028	812
HW_LRADC_CTRL2_SET	0x80050024	812

HW_LRADC_CTRL2_TOG	0x8005002C	812
HW_LRADC_CTRL3	0x80050030	815
HW_LRADC_CTRL3_CLR	0x80050038	815
HW_LRADC_CTRL3_SET	0x80050034	815
HW_LRADC_CTRL3_TOG	0x8005003C	815
HW_LRADC_CTRL4	0x80050140	838
HW_LRADC_CTRL4_CLR	0x80050148	838
HW_LRADC_CTRL4_SET	0x80050144	838
HW_LRADC_CTRL4_TOG	0x8005014C	838
HW_LRADC_DEBUG0	0x80050110	834
HW_LRADC_DEBUG0_CLR	0x80050118	834
HW_LRADC_DEBUG0_SET	0x80050114	834
HW_LRADC_DEBUG0_TOG	0x8005011C	834
HW_LRADC_DEBUG1	0x80050120	835
HW_LRADC_DEBUG1_CLR	0x80050128	835
HW_LRADC_DEBUG1_SET	0x80050124	835
HW_LRADC_DEBUG1_TOG	0x8005012C	835
HW_LRADC_DELAY0	0x800500D0	828
HW_LRADC_DELAY0_CLR	0x800500D8	828
HW_LRADC_DELAY0_SET	0x800500D4	828
HW_LRADC_DELAY0_TOG	0x800500DC	828
HW_LRADC_DELAY1	0x800500E0	830
HW_LRADC_DELAY1_CLR	0x800500E8	830
HW_LRADC_DELAY1_SET	0x800500E4	830
HW_LRADC_DELAY1_TOG	0x800500EC	830
HW_LRADC_DELAY2	0x800500F0	831
HW_LRADC_DELAY2_CLR	0x800500F8	831
HW_LRADC_DELAY2_SET	0x800500F4	831
HW_LRADC_DELAY2_TOG	0x800500FC	831
HW_LRADC_DELAY3	0x80050100	833
HW_LRADC_DELAY3_CLR	0x80050108	833
HW_LRADC_DELAY3_SET	0x80050104	833
HW_LRADC_DELAY3_TOG	0x8005010C	833
HW_LRADC_STATUS	0x80050040	817
HW_LRADC_STATUS_CLR	0x80050048	817
HW_LRADC_STATUS_SET	0x80050044	817
HW_LRADC_STATUS_TOG	0x8005004C	817
HW_LRADC_VERSION	0x80050150	841
HW_OCOTP_CRYPT00	0x8002C060	178
HW_OCOTP_CRYPT01	0x8002C070	178
HW_OCOTP_CRYPT02	0x8002C080	179
HW_OCOTP_CRYPT03	0x8002C090	179
HW_OCOTP_CTRL	0x8002C000	174
HW_OCOTP_CTRL_CLR	0x8002C008	174
HW_OCOTP_CTRL_SET	0x8002C004	174
HW_OCOTP_CTRL_TOG	0x8002C00C	174
HW_OCOTP_CUST0	0x8002C020	176
HW_OCOTP_CUST1	0x8002C030	177
HW_OCOTP_CUST2	0x8002C040	177
HW_OCOTP_CUST3	0x8002C050	177
HW_OCOTP_CUSTCAP	0x8002C110	180
HW_OCOTP_DATA	0x8002C010	176
HW_OCOTP_LOCK	0x8002C120	180
HW_OCOTP_ROM0	0x8002C1A0	181
HW_OCOTP_ROM1	0x8002C1B0	182
HW_OCOTP_ROM2	0x8002C1C0	185
HW_OCOTP_VERSION	0x8002C220	185
HW_PINCTRL_CTRL	0x80018000	938
HW_PINCTRL_CTRL_CLR	0x80018008	938
HW_PINCTRL_CTRL_SET	0x80018004	938
HW_PINCTRL_CTRL_TOG	0x8001800C	938
HW_PINCTRL_DIN0	0x80018500	993
HW_PINCTRL_DIN0_CLR	0x80018508	993

STMP3770

HW_PINCTRL_DIN0_SET	0x80018504	993
HW_PINCTRL_DIN0_TOG	0x8001850C	993
HW_PINCTRL_DIN1	0x80018510	994
HW_PINCTRL_DIN1_CLR	0x80018518	994
HW_PINCTRL_DIN1_SET	0x80018514	994
HW_PINCTRL_DIN1_TOG	0x8001851C	994
HW_PINCTRL_DIN2	0x80018520	994
HW_PINCTRL_DIN2_CLR	0x80018528	994
HW_PINCTRL_DIN2_SET	0x80018524	994
HW_PINCTRL_DIN2_TOG	0x8001852C	994
HW_PINCTRL_DOE0	0x80018600	995
HW_PINCTRL_DOE0_CLR	0x80018608	995
HW_PINCTRL_DOE0_SET	0x80018604	995
HW_PINCTRL_DOE0_TOG	0x8001860C	995
HW_PINCTRL_DOE1	0x80018610	995
HW_PINCTRL_DOE1_CLR	0x80018618	995
HW_PINCTRL_DOE1_SET	0x80018614	995
HW_PINCTRL_DOE1_TOG	0x8001861C	995
HW_PINCTRL_DOE2	0x80018620	996
HW_PINCTRL_DOE2_CLR	0x80018628	996
HW_PINCTRL_DOE2_SET	0x80018624	996
HW_PINCTRL_DOE2_TOG	0x8001862C	996
HW_PINCTRL_DOUT0	0x80018400	991
HW_PINCTRL_DOUT0_CLR	0x80018408	991
HW_PINCTRL_DOUT0_SET	0x80018404	991
HW_PINCTRL_DOUT0_TOG	0x8001840C	991
HW_PINCTRL_DOUT1	0x80018410	992
HW_PINCTRL_DOUT1_CLR	0x80018418	992
HW_PINCTRL_DOUT1_SET	0x80018414	992
HW_PINCTRL_DOUT1_TOG	0x8001841C	992
HW_PINCTRL_DOUT2	0x80018420	992
HW_PINCTRL_DOUT2_CLR	0x80018428	992
HW_PINCTRL_DOUT2_SET	0x80018424	992
HW_PINCTRL_DOUT2_TOG	0x8001842C	992
HW_PINCTRL_DRIVE0	0x80018200	955
HW_PINCTRL_DRIVE0_CLR	0x80018208	955
HW_PINCTRL_DRIVE0_SET	0x80018204	955
HW_PINCTRL_DRIVE0_TOG	0x8001820C	955
HW_PINCTRL_DRIVE1	0x80018210	958
HW_PINCTRL_DRIVE1_CLR	0x80018218	958
HW_PINCTRL_DRIVE1_SET	0x80018214	958
HW_PINCTRL_DRIVE1_TOG	0x8001821C	958
HW_PINCTRL_DRIVE10	0x800182A0	976
HW_PINCTRL_DRIVE10_CLR	0x800182A8	976
HW_PINCTRL_DRIVE10_SET	0x800182A4	976
HW_PINCTRL_DRIVE10_TOG	0x800182AC	976
HW_PINCTRL_DRIVE11	0x800182B0	979
HW_PINCTRL_DRIVE11_CLR	0x800182B8	979
HW_PINCTRL_DRIVE11_SET	0x800182B4	979
HW_PINCTRL_DRIVE11_TOG	0x800182BC	979
HW_PINCTRL_DRIVE12	0x800182C0	981
HW_PINCTRL_DRIVE12_CLR	0x800182C8	981
HW_PINCTRL_DRIVE12_SET	0x800182C4	981
HW_PINCTRL_DRIVE12_TOG	0x800182CC	981
HW_PINCTRL_DRIVE13	0x800182D0	983
HW_PINCTRL_DRIVE13_CLR	0x800182D8	983
HW_PINCTRL_DRIVE13_SET	0x800182D4	983
HW_PINCTRL_DRIVE13_TOG	0x800182DC	983
HW_PINCTRL_DRIVE14	0x800182E0	985
HW_PINCTRL_DRIVE14_CLR	0x800182E8	985
HW_PINCTRL_DRIVE14_SET	0x800182E4	985
HW_PINCTRL_DRIVE14_TOG	0x800182EC	985
HW_PINCTRL_DRIVE2	0x80018220	960

HW_PINCTRL_DRIVE2_CLR	0x80018228	960
HW_PINCTRL_DRIVE2_SET	0x80018224	960
HW_PINCTRL_DRIVE2_TOG	0x8001822C	960
HW_PINCTRL_DRIVE3	0x80018230	962
HW_PINCTRL_DRIVE3_CLR	0x80018238	962
HW_PINCTRL_DRIVE3_SET	0x80018234	962
HW_PINCTRL_DRIVE3_TOG	0x8001823C	962
HW_PINCTRL_DRIVE4	0x80018240	964
HW_PINCTRL_DRIVE4_CLR	0x80018248	964
HW_PINCTRL_DRIVE4_SET	0x80018244	964
HW_PINCTRL_DRIVE4_TOG	0x8001824C	964
HW_PINCTRL_DRIVE5	0x80018250	966
HW_PINCTRL_DRIVE5_CLR	0x80018258	966
HW_PINCTRL_DRIVE5_SET	0x80018254	966
HW_PINCTRL_DRIVE5_TOG	0x8001825C	966
HW_PINCTRL_DRIVE6	0x80018260	968
HW_PINCTRL_DRIVE6_CLR	0x80018268	968
HW_PINCTRL_DRIVE6_SET	0x80018264	968
HW_PINCTRL_DRIVE6_TOG	0x8001826C	968
HW_PINCTRL_DRIVE7	0x80018270	970
HW_PINCTRL_DRIVE7_CLR	0x80018278	970
HW_PINCTRL_DRIVE7_SET	0x80018274	970
HW_PINCTRL_DRIVE7_TOG	0x8001827C	970
HW_PINCTRL_DRIVE8	0x80018280	972
HW_PINCTRL_DRIVE8_CLR	0x80018288	972
HW_PINCTRL_DRIVE8_SET	0x80018284	972
HW_PINCTRL_DRIVE8_TOG	0x8001828C	972
HW_PINCTRL_DRIVE9	0x80018290	974
HW_PINCTRL_DRIVE9_CLR	0x80018298	974
HW_PINCTRL_DRIVE9_SET	0x80018294	974
HW_PINCTRL_DRIVE9_TOG	0x8001829C	974
HW_PINCTRL_IRQEN0	0x80018800	999
HW_PINCTRL_IRQEN0_CLR	0x80018808	999
HW_PINCTRL_IRQEN0_SET	0x80018804	999
HW_PINCTRL_IRQEN0_TOG	0x8001880C	999
HW_PINCTRL_IRQEN1	0x80018810	1000
HW_PINCTRL_IRQEN1_CLR	0x80018818	1000
HW_PINCTRL_IRQEN1_SET	0x80018814	1000
HW_PINCTRL_IRQEN1_TOG	0x8001881C	1000
HW_PINCTRL_IRQEN2	0x80018820	1000
HW_PINCTRL_IRQEN2_CLR	0x80018828	1001
HW_PINCTRL_IRQEN2_SET	0x80018824	1001
HW_PINCTRL_IRQEN2_TOG	0x8001882C	1001
HW_PINCTRL_IRQLEVEL0	0x80018900	1001
HW_PINCTRL_IRQLEVEL0_CLR	0x80018908	1001
HW_PINCTRL_IRQLEVEL0_SET	0x80018904	1001
HW_PINCTRL_IRQLEVEL0_TOG	0x8001890C	1001
HW_PINCTRL_IRQLEVEL1	0x80018910	1002
HW_PINCTRL_IRQLEVEL1_CLR	0x80018918	1002
HW_PINCTRL_IRQLEVEL1_SET	0x80018914	1002
HW_PINCTRL_IRQLEVEL1_TOG	0x8001891C	1002
HW_PINCTRL_IRQLEVEL2	0x80018920	1003
HW_PINCTRL_IRQLEVEL2_CLR	0x80018928	1003
HW_PINCTRL_IRQLEVEL2_SET	0x80018924	1003
HW_PINCTRL_IRQLEVEL2_TOG	0x8001892C	1003
HW_PINCTRL_IRQPOL0	0x80018a00	1003
HW_PINCTRL_IRQPOL0_CLR	0x80018a08	1003
HW_PINCTRL_IRQPOL0_SET	0x80018a04	1003
HW_PINCTRL_IRQPOL0_TOG	0x80018a0C	1003
HW_PINCTRL_IRQPOL1	0x80018a10	1004
HW_PINCTRL_IRQPOL1_CLR	0x80018a18	1004
HW_PINCTRL_IRQPOL1_SET	0x80018a14	1004
HW_PINCTRL_IRQPOL1_TOG	0x80018a1C	1004

STMP3770

HW_PINCTRL_IRQPOL2	0x80018A20.....	1005
HW_PINCTRL_IRQPOL2_CLR	0x80018A28.....	1005
HW_PINCTRL_IRQPOL2_SET	0x80018A24.....	1005
HW_PINCTRL_IRQPOL2_TOG	0x80018A2C.....	1005
HW_PINCTRL_IRQSTAT0	0x80018B00.....	1005
HW_PINCTRL_IRQSTAT0_CLR	0x80018B08.....	1005
HW_PINCTRL_IRQSTAT0_SET	0x80018B04.....	1005
HW_PINCTRL_IRQSTAT0_TOG	0x80018B0C.....	1005
HW_PINCTRL_IRQSTAT1	0x80018b10.....	1006
HW_PINCTRL_IRQSTAT1_CLR	0x80018b18.....	1006
HW_PINCTRL_IRQSTAT1_SET	0x80018b14.....	1006
HW_PINCTRL_IRQSTAT1_TOG	0x80018b1C.....	1006
HW_PINCTRL_IRQSTAT2	0x80018b20.....	1007
HW_PINCTRL_IRQSTAT2_CLR	0x80018b28.....	1007
HW_PINCTRL_IRQSTAT2_SET	0x80018b24.....	1007
HW_PINCTRL_IRQSTAT2_TOG	0x80018b2C.....	1007
HW_PINCTRL_MUXSEL0	0x80018100.....	939
HW_PINCTRL_MUXSEL0_CLR	0x80018108.....	939
HW_PINCTRL_MUXSEL0_SET	0x80018104.....	939
HW_PINCTRL_MUXSEL0_TOG	0x8001810C.....	939
HW_PINCTRL_MUXSEL1	0x80018110.....	941
HW_PINCTRL_MUXSEL1_CLR	0x80018118.....	941
HW_PINCTRL_MUXSEL1_SET	0x80018114.....	941
HW_PINCTRL_MUXSEL1_TOG	0x8001811C.....	941
HW_PINCTRL_MUXSEL2	0x80018120.....	943
HW_PINCTRL_MUXSEL2_CLR	0x80018128.....	943
HW_PINCTRL_MUXSEL2_SET	0x80018124.....	943
HW_PINCTRL_MUXSEL2_TOG	0x8001812C.....	943
HW_PINCTRL_MUXSEL3	0x80018130.....	945
HW_PINCTRL_MUXSEL3_CLR	0x80018138.....	945
HW_PINCTRL_MUXSEL3_SET	0x80018134.....	945
HW_PINCTRL_MUXSEL3_TOG	0x8001813C.....	945
HW_PINCTRL_MUXSEL4	0x80018140.....	947
HW_PINCTRL_MUXSEL4_CLR	0x80018148.....	947
HW_PINCTRL_MUXSEL4_SET	0x80018144.....	947
HW_PINCTRL_MUXSEL4_TOG	0x8001814C.....	947
HW_PINCTRL_MUXSEL5	0x80018150.....	950
HW_PINCTRL_MUXSEL5_CLR	0x80018158.....	950
HW_PINCTRL_MUXSEL5_SET	0x80018154.....	950
HW_PINCTRL_MUXSEL5_TOG	0x8001815C.....	950
HW_PINCTRL_MUXSEL6	0x80018160.....	952
HW_PINCTRL_MUXSEL6_CLR	0x80018168.....	952
HW_PINCTRL_MUXSEL6_SET	0x80018164.....	952
HW_PINCTRL_MUXSEL6_TOG	0x8001816C.....	952
HW_PINCTRL_MUXSEL7	0x80018170.....	954
HW_PINCTRL_MUXSEL7_CLR	0x80018178.....	954
HW_PINCTRL_MUXSEL7_SET	0x80018174.....	954
HW_PINCTRL_MUXSEL7_TOG	0x8001817C.....	954
HW_PINCTRL_PIN2IRQ0	0x80018700.....	997
HW_PINCTRL_PIN2IRQ0_CLR	0x80018708.....	997
HW_PINCTRL_PIN2IRQ0_SET	0x80018704.....	997
HW_PINCTRL_PIN2IRQ0_TOG	0x8001870C.....	997
HW_PINCTRL_PIN2IRQ1	0x80018710.....	997
HW_PINCTRL_PIN2IRQ1_CLR	0x80018718.....	997
HW_PINCTRL_PIN2IRQ1_SET	0x80018714.....	997
HW_PINCTRL_PIN2IRQ1_TOG	0x8001871C.....	997
HW_PINCTRL_PIN2IRQ2	0x80018720.....	998
HW_PINCTRL_PIN2IRQ2_CLR	0x80018728.....	998
HW_PINCTRL_PIN2IRQ2_SET	0x80018724.....	998
HW_PINCTRL_PIN2IRQ2_TOG	0x8001872C.....	998
HW_PINCTRL_PULL0	0x80018300.....	987
HW_PINCTRL_PULL0_CLR	0x80018308.....	987
HW_PINCTRL_PULL0_SET	0x80018304.....	987

HW_PINCTRL_PULL0_TOG	0x8001830C	987
HW_PINCTRL_PULL1	0x80018310	988
HW_PINCTRL_PULL1_CLR	0x80018318	988
HW_PINCTRL_PULL1_SET	0x80018314	988
HW_PINCTRL_PULL1_TOG	0x8001831C	988
HW_PINCTRL_PULL2	0x80018320	989
HW_PINCTRL_PULL2_CLR	0x80018328	989
HW_PINCTRL_PULL2_SET	0x80018324	989
HW_PINCTRL_PULL2_TOG	0x8001832C	989
HW_PINCTRL_PULL3	0x80018330	990
HW_PINCTRL_PULL3_CLR	0x80018338	990
HW_PINCTRL_PULL3_SET	0x80018334	990
HW_PINCTRL_PULL3_TOG	0x8001833C	990
HW_POWER_5VCTRL	0x80044010	861
HW_POWER_5VCTRL_CLR	0x80044018	861
HW_POWER_5VCTRL_SET	0x80044014	861
HW_POWER_5VCTRL_TOG	0x8004401C	861
HW_POWER_BATTMONITOR	0x800440D0	878
HW_POWER_CHARGE	0x80044030	864
HW_POWER_CHARGE_CLR	0x80044038	864
HW_POWER_CHARGE_SET	0x80044034	864
HW_POWER_CHARGE_TOG	0x8004403C	864
HW_POWER_CTRL	0x80044000	859
HW_POWER_CTRL_CLR	0x80044008	859
HW_POWER_CTRL_SET	0x80044004	859
HW_POWER_CTRL_TOG	0x8004400C	859
HW_POWER_DCFUNCV	0x80044070	871
HW_POWER_DCLIMITS	0x80044090	872
HW_POWER_DEBUG	0x800440F0	880
HW_POWER_DEBUG_CLR	0x800440F8	880
HW_POWER_DEBUG_SET	0x800440F4	880
HW_POWER_DEBUG_TOG	0x800440FC	880
HW_POWER_LOOPCTRL	0x800440A0	873
HW_POWER_LOOPCTRL_CLR	0x800440A8	873
HW_POWER_LOOPCTRL_SET	0x800440A4	873
HW_POWER_LOOPCTRL_TOG	0x800440AC	873
HW_POWER_MINPWR	0x80044020	862
HW_POWER_MINPWR_CLR	0x80044028	862
HW_POWER_MINPWR_SET	0x80044024	862
HW_POWER_MINPWR_TOG	0x8004402C	862
HW_POWER_MISC	0x80044080	872
HW_POWER_RESET	0x800440E0	879
HW_POWER_RESET_CLR	0x800440E8	879
HW_POWER_RESET_SET	0x800440E4	879
HW_POWER_RESET_TOG	0x800440EC	879
HW_POWER_SPECIAL	0x80044100	881
HW_POWER_SPECIAL_CLR	0x80044108	881
HW_POWER_SPECIAL_SET	0x80044104	881
HW_POWER_SPECIAL_TOG	0x8004410C	881
HW_POWER_SPEED	0x800440C0	877
HW_POWER_SPEED_CLR	0x800440C8	877
HW_POWER_SPEED_SET	0x800440C4	877
HW_POWER_SPEED_TOG	0x800440CC	877
HW_POWER_STS	0x800440B0	875
HW_POWER_VDDACTRL	0x80044050	867
HW_POWER_VDDDCTRL	0x80044040	865
HW_POWER_VDDIOCTRL	0x80044060	869
HW_POWER_VERSION	0x80044110	881
HW_PWM_ACTIVE0	0x80064010	642
HW_PWM_ACTIVE0_CLR	0x80064018	642
HW_PWM_ACTIVE0_SET	0x80064014	642
HW_PWM_ACTIVE0_TOG	0x8006401C	642
HW_PWM_ACTIVE1	0x80064030	644

STMP3770

HW_PWM_ACTIVE1_CLR	0x80064038	644
HW_PWM_ACTIVE1_SET	0x80064034	644
HW_PWM_ACTIVE1_TOG	0x8006403C	644
HW_PWM_ACTIVE2	0x80064050	646
HW_PWM_ACTIVE2_CLR	0x80064058	646
HW_PWM_ACTIVE2_SET	0x80064054	646
HW_PWM_ACTIVE2_TOG	0x8006405C	646
HW_PWM_ACTIVE3	0x80064070	647
HW_PWM_ACTIVE3_CLR	0x80064078	647
HW_PWM_ACTIVE3_SET	0x80064074	647
HW_PWM_ACTIVE3_TOG	0x8006407C	647
HW_PWM_ACTIVE4	0x80064090	649
HW_PWM_ACTIVE4_CLR	0x80064098	649
HW_PWM_ACTIVE4_SET	0x80064094	649
HW_PWM_ACTIVE4_TOG	0x8006409C	649
HW_PWM_CTRL	0x80064000	640
HW_PWM_CTRL_CLR	0x80064008	640
HW_PWM_CTRL_SET	0x80064004	640
HW_PWM_CTRL_TOG	0x8006400C	640
HW_PWM_PERIOD0	0x80064020	643
HW_PWM_PERIOD0_CLR	0x80064028	643
HW_PWM_PERIOD0_SET	0x80064024	643
HW_PWM_PERIOD0_TOG	0x8006402C	643
HW_PWM_PERIOD1	0x80064040	644
HW_PWM_PERIOD1_CLR	0x80064048	644
HW_PWM_PERIOD1_SET	0x80064044	644
HW_PWM_PERIOD1_TOG	0x8006404C	644
HW_PWM_PERIOD2	0x80064060	646
HW_PWM_PERIOD2_CLR	0x80064068	646
HW_PWM_PERIOD2_SET	0x80064064	646
HW_PWM_PERIOD2_TOG	0x8006406C	646
HW_PWM_PERIOD3	0x80064080	648
HW_PWM_PERIOD3_CLR	0x80064088	648
HW_PWM_PERIOD3_SET	0x80064084	648
HW_PWM_PERIOD3_TOG	0x8006408C	648
HW_PWM_PERIOD4	0x800640A0	650
HW_PWM_PERIOD4_CLR	0x800640A8	650
HW_PWM_PERIOD4_SET	0x800640A4	650
HW_PWM_PERIOD4_TOG	0x800640AC	650
HW_PWM_VERSION	0x800640b0	651
HW_RTC_ALARM	0x8005C040	627
HW_RTC_ALARM_CLR	0x8005C048	627
HW_RTC_ALARM_SET	0x8005C044	627
HW_RTC_ALARM_TOG	0x8005C04C	627
HW_RTC_CTRL	0x8005C000	622
HW_RTC_CTRL_CLR	0x8005C008	622
HW_RTC_CTRL_SET	0x8005C004	622
HW_RTC_CTRL_TOG	0x8005C00C	622
HW_RTC_DEBUG	0x8005C0C0	633
HW_RTC_DEBUG_CLR	0x8005C0C8	633
HW_RTC_DEBUG_SET	0x8005C0C4	633
HW_RTC_DEBUG_TOG	0x8005C0CC	633
HW_RTC_MILLISECONDS	0x8005C020	625
HW_RTC_MILLISECONDS_CLR	0x8005C028	625
HW_RTC_MILLISECONDS_SET	0x8005C024	625
HW_RTC_MILLISECONDS_TOG	0x8005C02C	625
HW_RTC_PERSISTENT0	0x8005C060	628
HW_RTC_PERSISTENT0_CLR	0x8005C068	628
HW_RTC_PERSISTENT0_SET	0x8005C064	628
HW_RTC_PERSISTENT0_TOG	0x8005C06C	628
HW_RTC_PERSISTENT1	0x8005C070	630
HW_RTC_PERSISTENT1_CLR	0x8005C078	630
HW_RTC_PERSISTENT1_SET	0x8005C074	630

HW_RTC_PERSISTENT1_TOG	0x8005C07C	630
HW_RTC_PERSISTENT2	0x8005C080	631
HW_RTC_PERSISTENT2_CLR	0x8005C088	631
HW_RTC_PERSISTENT2_SET	0x8005C084	631
HW_RTC_PERSISTENT2_TOG	0x8005C08C	631
HW_RTC_PERSISTENT3	0x8005C090	631
HW_RTC_PERSISTENT3_CLR	0x8005C098	631
HW_RTC_PERSISTENT3_SET	0x8005C094	631
HW_RTC_PERSISTENT3_TOG	0x8005C09C	632
HW_RTC_PERSISTENT4	0x8005C0A0	632
HW_RTC_PERSISTENT4_CLR	0x8005C0A8	632
HW_RTC_PERSISTENT4_SET	0x8005C0A4	632
HW_RTC_PERSISTENT4_TOG	0x8005C0AC	632
HW_RTC_PERSISTENT5	0x8005C0B0	633
HW_RTC_PERSISTENT5_CLR	0x8005C0B8	633
HW_RTC_PERSISTENT5_SET	0x8005C0B4	633
HW_RTC_PERSISTENT5_TOG	0x8005C0BC	633
HW_RTC_SECONDS	0x8005C030	626
HW_RTC_SECONDS_CLR	0x8005C038	626
HW_RTC_SECONDS_SET	0x8005C034	626
HW_RTC_SECONDS_TOG	0x8005C03C	626
HW_RTC_STAT	0x8005C010	624
HW_RTC_STAT_CLR	0x8005C018	624
HW_RTC_STAT_SET	0x8005C014	624
HW_RTC_STAT_TOG	0x8005C01C	624
HW_RTC_VERSION	0x8005C0D0	634
HW_RTC_WATCHDOG	0x8005C050	627
HW_RTC_WATCHDOG_CLR	0x8005C058	627
HW_RTC_WATCHDOG_SET	0x8005C054	627
HW_RTC_WATCHDOG_TOG	0x8005C05C	627
HW_SPDIF_CTRL	0x80054000	781
HW_SPDIF_CTRL_CLR	0x80054008	781
HW_SPDIF_CTRL_SET	0x80054004	781
HW_SPDIF_CTRL_TOG	0x8005400C	781
HW_SPDIF_DATA	0x80054050	786
HW_SPDIF_DATA_CLR	0x80054058	786
HW_SPDIF_DATA_SET	0x80054054	786
HW_SPDIF_DATA_TOG	0x8005405C	786
HW_SPDIF_DEBUG	0x80054040	786
HW_SPDIF_DEBUG_CLR	0x80054048	786
HW_SPDIF_DEBUG_SET	0x80054044	786
HW_SPDIF_DEBUG_TOG	0x8005404C	786
HW_SPDIF_FRAMECTRL	0x80054020	784
HW_SPDIF_FRAMECTRL_CLR	0x80054028	784
HW_SPDIF_FRAMECTRL_SET	0x80054024	784
HW_SPDIF_FRAMECTRL_TOG	0x8005402C	784
HW_SPDIF_SRR	0x80054030	785
HW_SPDIF_SRR_CLR	0x80054038	785
HW_SPDIF_SRR_SET	0x80054034	785
HW_SPDIF_SRR_TOG	0x8005403C	785
HW_SPDIF_STAT	0x80054010	783
HW_SPDIF_STAT_CLR	0x80054018	783
HW_SPDIF_STAT_SET	0x80054014	783
HW_SPDIF_STAT_TOG	0x8005401C	783
HW_SPDIF_VERSION	0x80054060	787
HW_SSP_CMD0	0x80010010	546
HW_SSP_CMD0_CLR	0x80010018	546
HW_SSP_CMD0_SET	0x80010014	546
HW_SSP_CMD0_TOG	0x8001001C	546
HW_SSP_CMD1	0x80010020	550
HW_SSP_COMPMASK	0x80010040	550
HW_SSP_COMPREF	0x80010030	550
HW_SSP_CTRL0	0x80010000	544

STMP3770

HW_SSP_CTRL0_CLR	0x80010008	544
HW_SSP_CTRL0_SET	0x80010004	544
HW_SSP_CTRL0_TOG	0x8001000C	544
HW_SSP_CTRL1	0x80010060	551
HW_SSP_CTRL1_CLR	0x80010068	551
HW_SSP_CTRL1_SET	0x80010064	551
HW_SSP_CTRL1_TOG	0x8001006C	551
HW_SSP_DATA	0x80010070	554
HW_SSP_DEBUG	0x80010100	558
HW_SSP_SDRESP0	0x80010080	555
HW_SSP_SDRESP1	0x80010090	555
HW_SSP_SDRESP2	0x800100A0	556
HW_SSP_SDRESP3	0x800100B0	556
HW_SSP_STATUS	0x800100C0	556
HW_SSP_TIMING	0x80010050	551
HW_SSP_VERSION	0x80010110	559
HW_TIMROT_ROT_COUNT	0x80068010	604
HW_TIMROT_ROT_CTRL	0x80068000	603
HW_TIMROT_ROT_CTRL_CLR	0x80068008	603
HW_TIMROT_ROT_CTRL_SET	0x80068004	603
HW_TIMROT_ROT_CTRL_TOG	0x8006800C	603
HW_TIMROT_TIMCOUNT0	0x80068030	606
HW_TIMROT_TIMCOUNT1	0x80068050	609
HW_TIMROT_TIMCOUNT2	0x80068070	611
HW_TIMROT_TIMCOUNT3	0x80068090	614
HW_TIMROT_TIMCTRL0	0x80068020	605
HW_TIMROT_TIMCTRL0_CLR	0x80068028	605
HW_TIMROT_TIMCTRL0_SET	0x80068024	605
HW_TIMROT_TIMCTRL0_TOG	0x8006802C	605
HW_TIMROT_TIMCTRL1	0x80068040	607
HW_TIMROT_TIMCTRL1_CLR	0x80068048	607
HW_TIMROT_TIMCTRL1_SET	0x80068044	607
HW_TIMROT_TIMCTRL1_TOG	0x8006804C	607
HW_TIMROT_TIMCTRL2	0x80068060	609
HW_TIMROT_TIMCTRL2_CLR	0x80068068	609
HW_TIMROT_TIMCTRL2_SET	0x80068064	609
HW_TIMROT_TIMCTRL2_TOG	0x8006806C	609
HW_TIMROT_TIMCTRL3	0x80068080	612
HW_TIMROT_TIMCTRL3_CLR	0x80068088	612
HW_TIMROT_TIMCTRL3_SET	0x80068084	612
HW_TIMROT_TIMCTRL3_TOG	0x8006808C	612
HW_TIMROT_VERSION	0x800680a0	615
HW_UARTAPP_CTRL0	0x8006C000	689
HW_UARTAPP_CTRL0_CLR	0x8006C008	689
HW_UARTAPP_CTRL0_SET	0x8006C004	689
HW_UARTAPP_CTRL0_TOG	0x8006C00C	689
HW_UARTAPP_CTRL1	0x8006C010	690
HW_UARTAPP_CTRL1_CLR	0x8006C018	690
HW_UARTAPP_CTRL1_SET	0x8006C014	690
HW_UARTAPP_CTRL1_TOG	0x8006C01C	690
HW_UARTAPP_CTRL2	0x8006C020	691
HW_UARTAPP_CTRL2_CLR	0x8006C028	691
HW_UARTAPP_CTRL2_SET	0x8006C024	691
HW_UARTAPP_CTRL2_TOG	0x8006C02C	691
HW_UARTAPP_DATA	0x8006C060	697
HW_UARTAPP_DEBUG	0x8006C080	700
HW_UARTAPP_INTR	0x8006C050	696
HW_UARTAPP_INTR_CLR	0x8006C058	696
HW_UARTAPP_INTR_SET	0x8006C054	696
HW_UARTAPP_INTR_TOG	0x8006C05C	696
HW_UARTAPP_LINECTRL	0x8006C030	693
HW_UARTAPP_LINECTRL_CLR	0x8006C038	693
HW_UARTAPP_LINECTRL_SET	0x8006C034	693

HW_UARTAPP_LINECTRL_TOG	0x8006C03C	693
HW_UARTAPP_LINECTRL2	0x8006C040	695
HW_UARTAPP_LINECTRL2_CLR	0x8006C048	695
HW_UARTAPP_LINECTRL2_SET	0x8006C044	695
HW_UARTAPP_LINECTRL2_TOG	0x8006C04C	695
HW_UARTAPP_STAT	0x8006C070	698
HW_UARTAPP_VERSION	0x8006C090	701
HW_UARTDBGCR	0x80070030	712
HW_UARTDBGDMACR	0x80070048	718
HW_UARTDBGDR	0x80070000	706
HW_UARTDBGFBRD	0x80070028	710
HW_UARTDBGFR	0x80070018	709
HW_UARTDBGIBRD	0x80070024	709
HW_UARTDBGICR	0x80070044	717
HW_UARTDBGIFLS	0x80070034	713
HW_UARTDBGIMSC	0x80070038	714
HW_UARTDBGLCR_H	0x8007002C	710
HW_UARTDBGMIS	0x80070040	716
HW_UARTDBGRIIS	0x8007003C	715
HW_UARTDBGRRSR_ECR	0x80070004	708
HW_USBCTRL_BURSTSIZE	0x80080160	219
HW_USBCTRL_CAPLENGTH	0x80080100	198
HW_USBCTRL_DCCPARAMS	0x80080124	203
HW_USBCTRL_DCIVERSION	0x80080120	202
HW_USBCTRL_DEVICEADDR	0x80080154	215
HW_USBCTRL_ENDPOINTLISTADDR	0x80080158	217
HW_USBCTRL_ENDPTCOMPLETE	0x800801BC	241
HW_USBCTRL_ENDPTCTRL0	0x800801C0	242
HW_USBCTRL_ENDPTCTRL1	0x800801C4	244
HW_USBCTRL_ENDPTCTRL2	0x800801C8	244
HW_USBCTRL_ENDPTCTRL3	0x800801CC	244
HW_USBCTRL_ENDPTCTRL4	0x800801D0	244
HW_USBCTRL_ENDPTFLUSH	0x800801B4	239
HW_USBCTRL_ENDPTNAK	0x80080178	223
HW_USBCTRL_ENDPTNAKEN	0x8008017C	224
HW_USBCTRL_ENDPTPRIME	0x800801B0	238
HW_USBCTRL_ENDPTSETUPSTAT	0x800801AC	238
HW_USBCTRL_ENDPTSTAT	0x800801B8	240
HW_USBCTRL_FRINDEX	0x8008014C	214
HW_USBCTRL_GPTIMER0CTRL	0x80080084	197
HW_USBCTRL_GPTIMER0LD	0x80080080	196
HW_USBCTRL_GPTIMER1CTRL	0x8008008C	198
HW_USBCTRL_GPTIMER1LD	0x80080088	198
HW_USBCTRL_HCCPARAMS	0x80080108	201
HW_USBCTRL_HCIVERSION	0x80080102	199
HW_USBCTRL_HCSPARAMS	0x80080104	199
HW_USBCTRL_HWDEVICE	0x8008000C	195
HW_USBCTRL_HWHOST	0x80080008	194
HW_USBCTRL_HWRXBUF	0x80080014	196
HW_USBCTRL_HWTXBUF	0x80080010	195
HW_USBCTRL_ID	0x80080000	193
HW_USBCTRL_OTGSC	0x800801A4	233
HW_USBCTRL_PORTSC1	0x80080184	225
HW_USBCTRL_TTCTRL	0x8008015C	218
HW_USBCTRL_TXFILLTUNING	0x80080164	220
HW_USBCTRL_ULPI	0x80080170	222
HW_USBCTRL_USBCMD	0x80080140	203
HW_USBCTRL_USBINTR	0x80080148	212
HW_USBCTRL_USBMODE	0x800801A8	235
HW_USBCTRL_USBSTS	0x80080144	208
HW_USBPHY_CTRL	0x8007C030	259
HW_USBPHY_CTRL_CLR	0x8007C038	259
HW_USBPHY_CTRL_SET	0x8007C034	259

STMP3770

HW_USBPHY_CTRL_TOG	0x8007C03C	259
HW_USBPHY_DEBUG	0x8007C050	262
HW_USBPHY_DEBUG_CLR	0x8007C058	262
HW_USBPHY_DEBUG_SET	0x8007C054	262
HW_USBPHY_DEBUG_TOG	0x8007C05C	262
HW_USBPHY_DEBUG0_STATUS	0x8007C060	264
HW_USBPHY_DEBUG1	0x8007C070	264
HW_USBPHY_DEBUG1_CLR	0x8007C078	264
HW_USBPHY_DEBUG1_SET	0x8007C074	264
HW_USBPHY_DEBUG1_TOG	0x8007C07C	265
HW_USBPHY_PWD	0x8007C000	256
HW_USBPHY_PWD_CLR	0x8007C008	256
HW_USBPHY_PWD_SET	0x8007C004	256
HW_USBPHY_PWD_TOG	0x8007C00C	256
HW_USBPHY_RX	0x8007C020	258
HW_USBPHY_RX_CLR	0x8007C028	258
HW_USBPHY_RX_SET	0x8007C024	258
HW_USBPHY_RX_TOG	0x8007C02C	258
HW_USBPHY_STATUS	0x8007C040	261
HW_USBPHY_TX	0x8007C010	257
HW_USBPHY_TX_CLR	0x8007C018	257
HW_USBPHY_TX_SET	0x8007C014	257
HW_USBPHY_TX_TOG	0x8007C01C	257
HW_USBPHY_VERSION	0x8007C080	265